יסודות מדעי המחשב 1

בשפת #C

תמר בניה וד״ר מיכל ארמוני – ראשי צוות הכתיבה

יעל בילצ׳יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי התבניות)

הילה קדמן (נספח)

ייעוץ: ד״ר דוד גינת

עריכה: לירון ברגר

תשס״ח 2007

אוניברסיטת תל-אביב החוג להוראת המדעים

מטה מל״מ המרכז הישראלי להוראת המדעים ע״ש עמוס דה-שליט

משרד החינוך האגף לתכנון ולפיתוח תכניות לימודים

יסודות מדעי המחשב 1 בשפת #C

תמר בניה וד"ר מיכל ארמוני – ראשי צוות הכתיבה

יעל בילצ'יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי תבניות)

הילה קדמן (נספח)

ייעוץ: ד"ר דוד גינת

עריכה: לירון ברגר

כל הזכויות שמורות © 2007

השראה הוצאה לאור, ת"ד 19022, חיפה 31190

טל': 04-8254752, פקס: 1534-8254752

E-Mail: books@hashraa.co.il

www.hashraa.co.il

השראה הוצאה לאור

מהדורה שנייה 2007

עיצוב העטיפה: טל גרין

אין לשכפל, להעתיק, לצלם, לתרגם, להקליט, לאחסן במאגר מידע כלשהו, לשדר או לקלוט בכל דרך או בכל אמצעי אלקטרוני, אופטי או מכני (לרבות צילום, הקלטה, אינטרנט, מחשב ודואר אלקטרוני), כל חלק שהוא מהחומר שבספר זה. שימוש מסחרי מכל סוג בחומר הכלול בספר זה אסור בהחלט, אלא ברשות מפורשת בכתב מהמו"ל ומהגורמים המפורטים להלן.

©

כל הזכויות שמורות

משרד החינוך

מסת"ב ISBN 965-90844-5-5

פתח דבר

יחידות הלימוד "יסודות מדעי המחשב 1 ו-2" מיועדות להקניית מושגי יסוד ועקרונות שעליהם מושתת תחום מדעי המחשב. פרקי יחידות הלימוד משלבים שני ערוצים – ערוץ תיאורטי וערוץ יישומי. הערוץ התיאורטי מתמקד בחשיבה אלגוריתמית ובפיתוח וניתוח של אלגוריתמים וכוללת התייחסות למושג עצמים. הערוץ היישומי כולל יישום של האלגוריתמים בשפת התכנות C#, שפה מונחית עצמים.

ספר זה כולל את היחידה "יסודות מדעי המחשב 1". היחידה מציגה בעיות ראשונות ואת פתרונותיהן המיועדים לביצוע למחשב. הבעיות נקראות בעיות אלגוריתמיות, ופתרונותיהן – אלגוריתמים. האלגוריתמים מיושמים בתוכניות מחשב. במהלך הלימוד מוצגים המרכיבים הבסיסיים של אלגוריתמים ושל תוכניות מחשב. ההצגה משלבת פיתוח וניתוח של אלגוריתמים, וכוללת התייחסות ראשונית לַמושג עצמים. פיתוח האלגוריתמים נעשה בשלבים, תוך שימת דגש על ניתוח הבעיה ועל התייחסות להיבטים של נכונות ושל יעילות. כמו כן, מושם דגש על מבנים תבניתיים בפתרונות אלגוריתמיים, והם נקראים תבניות. פירוט מלא של התבניות מופיע באתר הספר: www.tau.ac.il/~csedu/yesodot.html.

ספר זה פותח על בסיס ספר הלימוד "יסודות מדעי המחשב 1" שפותח במכון ויצמן למדע בסוף שנות ה-90. בספר הקודם נעשה היישום של האלגוריתמים בשפת התכנות Pascal, שפה פרוצדורלית. בספר זה נעשה היישום בשפת C#, שפה מונחית עצמים. העקרונות האלגוריתמיים והנושאים בשמונת הפרקים הראשונים בספר זה זהים לאלה שפותחו בידי מכון ויצמן. בספר "יסודות מדעי המחשב 2" מורחב המבט על עצמים, על אלגוריתמים ועל תבניות. לספר זה מצורף מדריך מעבדה מקוון, המופיע באתר הספר המצוין לעיל.

תודות. ספר זה פותח בתמיכת מפמ"ר מדעי המחשב במשרד החינוך ד"ר אבי כהן וחברי שתי ועדות המקצוע האחרונות להוראת מדעי המחשב – הועדה בראשות פרופ' עמיהוד אמיר והועדה (הנוכחית) בראשות פרופ' יהודית גל-עזר. תודתנו נתונה להם על תמיכתם ועל הערותיהם. בנוסף, לאורך הספר משולבת התייחסות מפורשת לתבניות בפיתוח ובניתוח של אלגוריתמים. ההתייחסות מבוססת על הספר "תבניות במדעי המחשב" שפיתחו חברי הקבוצה להוראת מדעי-המחשב בחוג להוראת-המדעים באוניברסיטת תל-אביב בשנת 2001. ארנה מילר, אחת מחברות הקבוצה, אף חקרה את הנושא של הוראה מוכוונת תבניות, ושיתפה את חברות צוות הכתיבה בניסיונה. תודתנו נתונה לה על כך.

תוכן עניינים

פרק 1 – מבוא 1

1.1 מהו מחשב? 1

1.2 חומרה 2

1.3 תוכנה 5

1.4 התפתחות המחשבים ומדעי המחשב 8

התפתחות הנדסית וטכנולוגית – חומרה 8

התפתחות הנדסית וטכנולוגית – תוכנה 9

סיכום 11

שאלות נוספות 11

פרק 2 – פתרון בעיות אלגוריתמיות 13

2.1 אלגוריתמים 13

2.2 תבניות 21

סיכום 22

שאלות נוספות 22

פרק 3 – מודל חישוב בסיסי 25

3.1 צעדים ראשונים: הוראת פלט, הוראת קלט ומשתנים 25

3.2 הוראת הֲשָׂמָה 33

3.3 טבלת מעקב 39

3.4 החלפה בין ערכי משתנים 44

3.5 טיפוסים 46

3.6 קבועים 51

סיכום 52

סיכום מרכיבי שפת C# שנלמדו בפרק 3 54

שאלות נוספות 57

תבניות – פרק 3 61

החלפת ערכים בין שני משתנים, היפוך סדר האיברים בסדרה, ממוצע של סדרת מספרים, הזזה מעגלית בסדרה

פרק 4 – הרחבה בפיתוח אלגוריתמים 63

4.1 מבט נוסף אל התהליך של פיתוח אלגוריתם ויישומו 63

המחלקה המתמטית 67

4.2 פעולות חלוקה בשלמים 68

עוד על פעולת השארית 73

המרת ערך שלם לממשי 75

פירוק מספר דו-ספרתי לספרותיו 77

4.3 הטיפוס התווי 80
המרה מתו המייצג ספרה לערך מספרי מתאים 84
4.4 בחירה אקראית 85
סיכום 88
סיכום מרכיבי שפת C# שנלמדו בפרק 4 89
שאלות נוספות 90
תבניות – פרק 4 92
חלוקת כמות פריטים לקבוצות בגודל נתון, פירוק מספר חיובי לספרותיו, בניית מספר
פרק 5 – ביצוע מותנה 95
5.1 הוראה לביצוע-בתנאי 95
הוראה לביצוע-בתנאי במבנה *אם... אחרת...* 95
הוראה לביצוע-בתנאי במבנה *אם...* 101
התניית ביצוע של שתי הוראות או יותר 106
ביטויים בוליאניים הכוללים תווים 108
5.2 תנאי מורכב 111
הקֶשֶר *וגם* 112
הקֶשֶר *או* 116
תנאים מורכבים מעורבים 122
5.3 קינון של הוראה לביצוע-בתנאי 122
5.4 הוראת שרשרת לביצוע-בתנאי 129
5.5 הוראת בחירה 133
סיכום 138
סיכום מרכיבי שפת C# שנלמדו בפרק 5 139
שאלות נוספות 140
תבניות – פרק 5 143
מציאת מקסימום ומינימום בסדרה, סידור ערכים בסדרה, ערכים עוקבים, זוגיות מספר, מחלק של מספר
פרק 6 – נכונות אלגוריתמים 147
סיכום 154
פרק 7 – ביצוע-חוזר 155
7.1 ביצוע-חוזר מספר פעמים ידוע מראש 155
7.2 מציאת מקסימום או מינימום 169
7.3 מציאת ערך נלווה למקסימום או למינימום 172
7.4 ביצוע-חוזר-בתנאי 174
ביצוע-חוזר בשימוש בזקיף 174
ביצוע-חוזר עם תנאי כניסה כלשהו 180

ביצוע-חוזר אינסופי 187
7.5 משתנים מטיפוס בוליאני... 191
7.6 הקשר הלוגי לא (not).. 196
7.7 קינון הוראות לביצוע-חוזר.. 198
סיכום 202
סיכום מרכיבי שפת #C שנלמדו בפרק 7.. 204
תבניות – פרק 7 205
מנייה וצבירה, ממוצע של סדרת מספרים, מציאת מקסימום או מינימום בסדרה, מציאת ערך נלווה למקסימום או למינימום בסדרה, איסוף בקיזוז, פירוק מספר חיובי לספרותיו, בניית מספר, האם כל הערכים בסדרה מקיימים תנאי?, האם קיים ערך בסדרה המקיים תנאי?, מציאת כל הערכים בסדרה המקיימים תנאי, מעבר על זוגות סמוכים בסדרה
פרק 8 – יעילות של אלגוריתמים 213
סיכום 222
אינדקס.. 223

תוכן יסודות 2

פרק 9 – המחלקה מחרוזת (String)

פרק 10 – מערכים

פרק 11 – מחלקות ועצמים: הרחבה והעמקה

פרק 12 – תבניות אלגוריתמיות (מערך דו-ממדי, מיון, חיפוש ומיזוג)

פרק 13 – פתרון בעיות

פרק 1 – מבוא

בפרק מבוא קצר זה נסביר מעט על המחשב, על תפקידיו, על מבנהו ועל אופן השימוש בו לצורך פתרון בעיות מסוגים שונים. ייתכן כי לאלה מביניכם הרגילים בשימוש במחשב, ואולי אף בתכנות, חלק מהמושגים יהיו מוכרים. בכל זאת, סביר שקריאת הפרק תאיר כמה מהמושגים האלה ואת הקשרים ביניהם באור מעט שונה, וסביר שכמה מהמושגים המוצגים בפרק יהיו חדשים גם עבור התלמידים שרגילים בשימוש במחשב.

מחשבים מצויים במקומות רבים: הם מנחים מטוסים, אוניות וספינות חלל; הם מפקחים על מיליוני טלפונים המחוברים ברשת ופזורים על פני מדינות שונות; הם משמשים לחיזוי מזג האוויר, לכתיבה ולהדפסה של מסמכים, לבקרה על מערכות ייצור אוטומטיות, להלחנת מוסיקה, למשחקים ולדברים רבים נוספים.

ללא המחשב לא היה מתאפשר מבצע הנחתת אדם על הירח. המבצע דרש פתרון בעיות חישוביות קשות ומסובכות לקביעת מסלול חללית בהשפעת הירח, כדור הארץ והשמש. ביצוע החישובים האלה ללא מחשב היה מצריך עבודת צוות גדולה במשך עשרות שנים. בנוסף, כל שינוי במועד ההמראה או במסלול הטיסה חייב פתרון מחדש של בעיות חישוביות אלו. המבצע לא היה מתאפשר ללא מחשב.

1.1 מהו מחשב?

מחשב (computer) הוא מכונה אלקטרונית הקולטת נתונים, מעבדת אותם ופולטת מידע הנוצר בתהליך העיבוד.
הנתונים שקולט המחשב נקראים **קלט** (input); המידע שפולט המחשב נקרא **פלט** (output); העיבוד המבוצע במחשב מונחה על ידי אוסף הוראות הנקרא **תוכנית מחשב** (computer program).

הנה כמה דוגמאות להמחשת ההגדרות האלו:

- כשאנו מגיעים לקנות כרטיסים לסרט, הקופאי מקיש בלוח המקשים של המחשב שעל שולחנו את מספר הכרטיסים שאנו מבקשים לקנות. המחשב מעבד נתון זה בבדיקת המקומות הפנויים, בהקצאת מקומות כמבוקש ובסימון המקומות שהוקצו כתפוסים. המדפסת מדפיסה כרטיסים שעליהם מסומנים המקומות שהוקצו. במקרה זה הקלט הוא מספר הכרטיסים המבוקש, והפלט הוא הכרטיסים שעליהם מקומות מסומנים.
- אמן אנימציה מציין כקלט על צג מחשב שתי נקודות לתנועה של דמות – נקודת התחלה ונקודת סיום. המחשב מעבד נתונים אלה יחד עם נתונים נוספים על הדמות, ומציג על הצג כפלט את תנועת הדמות מנקודת ההתחלה לנקודת הסיום.
- מנהל חשבונות מקיש כקלט שם של עובד ואת מספר הימים שעבד בחודש האחרון. המחשב מעבד נתונים אלה יחד עם נתונים אחרים השמורים בו (כמו דרגת העובד), ומדפיס כפלט את המשכורת שמגיעה לעובד עבור החודש האחרון.
- מדען, המשתמש במחשב לצורך חישוב מסלול התעופה של טיל, נותן כקלט למחשב את נקודת שיגור הטיל, את זמן השיגור, את מהירות הטיל ונתונים על הרוחות במסלול מעופו הצפוי של הטיל. המחשב מעבד נתונים אלה בעזרת נוסחאות לחישוב תעופת טיל ובעזרת מידע השמור בו על כדור הארץ, ונותן כפלט את המקום שאמור הטיל לנחות בו.

שאלה 1.1

הביאו דוגמאות מסביבת הבית ובית הספר לשימוש במחשב, וציינו עבור כל דוגמה את הקלט ואת הפלט.

ייחודה של המכונה "מחשב" לעומת מכונות אחרות הוא במגוון השימושים הרחב שלה. בעוד שבמכונת כביסה אנו משתמשים כדי לכבס, במקרר כדי לשמור על מזון ובמכונית כדי להגיע ממקום למקום, הרי שבמחשב משתמשים בקשת רחבה של משימות. זה נובע מכך שהמחשב מבצע פעילויות שונות של עיבוד ואחסון מידע, פעילויות המונחות כולן על ידי תוכניות מחשב מתאימות.

אמנם, קיימים כיום מחשבים אשר מסוגלים לבצע בצורה מוגבלת תפקודים אנושיים, כמו ראייה, שמיעה, דיבור ותנועה, אבל המחשב אינו כל יכול. הוא בסך הכול מכונה. הוא איננו יכול למשל לדמיין, לכאוב או לשמוח. למעשה, גם אם נגביל את הדיון למשימות שאינן מערבות רגשות, ישנן משימות רבות שאינן ניתנות לביצוע על ידי מחשב.

יתרון בולט של מחשב הוא בכך שהוא מבצע פעולות חישוב ועיבוד מידע במהירות עצומה ובדיוק רב. למשל, מחשב יכול לקלוט אלף מספרים בני חמש ספרות כל אחד, ולחשב תוך שבריר שנייה את סכומם. מחשב יכול לקלוט טקסט בן אלף מילים, ולחשב תוך שבריר שנייה את האות השכיחה ביותר בטקסט.

יתרון חשוב נוסף של מחשב הוא שניתן לאחסן בזיכרונו מיליוני נתונים. למשל, במחשב של משרד הפנים מאוחסנים נתונים על כל אחד מתושבי המדינה (שם, מספר זהות, תאריך לידה, מצב משפחתי ועוד). במחשב של בנק מאוחסנים פרטים אישיים על כל לקוח של הבנק, והנתונים על התנועות בכל חשבונות הבנק.

אמנם המחשב מבצע פעולות חישוב ועיבוד בדיוק רב, אך מאחר שהוא לא יותר ממכונה, המבצעת הוראות, לא מובטח שתמיד ייתן המחשב כפלט תוצאת עיבוד נכונה. פלט לא נכון יכול להתקבל כתוצאה מקלט שגוי או מתוכנית מחשב שגויה.

? מהי תוכנית מחשב שגויה?

תוכנית מחשב נכתבת על ידי בני אדם. ייתכן שההוראות הכלולות בה אינן מורות על העיבוד הדרוש. במקרה כזה התוכנית שגויה, וייתכן כי כאשר המחשב פועל על פיה הוא ייתן כפלט תוצאת עיבוד לא נכונה. למשל, ייתכן כי מתכנת יכתוב תוכנית לחישוב ממוצע של אלף מספרים, ובה יורה על סיכום המספרים אך יטעה וישכח להורות על חלוקת הסכום המחושב ב-1000. ברור כי עיבוד שיתבצע על פי תוכנית זו לא יבצע חישוב ממוצע כנדרש וייתן פלט שגוי.

אנו מבחינים בין שני צדדים של מחשב: חומרה ותוכנה.

חומרה (hardware) היא הרכיבים הפיסיים שמרכיבים את המחשב.

תוכנה (software) היא אוסף תוכניות המחשב.

1.2 חומרה

בסעיף זה נתאר על קצה המזלג את הרכיבים הפיסיים של המחשב. במהלך לימוד היחידה "יסודות מדעי המחשב" תעבדו בוודאי על מחשבים אישיים (personal computer – pc). מחשבים אישיים הם קטנים יחסית בממדיהם ומיועדים לשרת בו זמנית משתמש אחד בלבד.

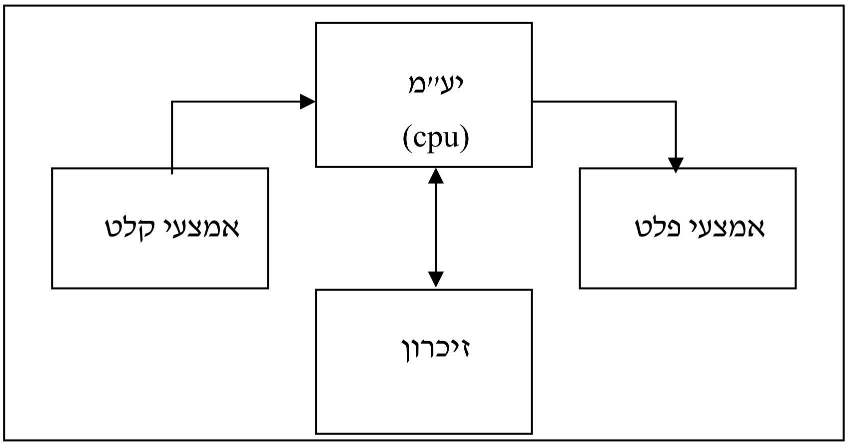
קיימים גם מחשבים גדולים יותר, שיכולים לשרת בו זמנית משתמשים רבים. עם זאת, המבנה הכללי של מחשב אינו תלוי בגודלו.

למחשב כמה יחידות בסיסיות :

- ♦ **יחידת עיבוד מרכזית** (central processing unit), ובקיצור **יע"מ** (CPU) היא היחידה שאחראית על עיבוד מידע, מבצעת חישובים ומנהלת את כל התהליכים המתבצעים במחשב.
- ♦ ב**זיכרון** (memory) נשמרים תוכניות המחשב, המידע שבו משתמש המחשב ותוצאות ביניים של תהליכי עיבוד.
- ♦ דרך **אמצעי קלט** (input devices) מתבצעת קליטת נתונים.
- ♦ דרך **אמצעי פלט** (output devices) ניתן הפלט של תוצאות העיבוד.

איור 1.1 מתאר את היחידות הבסיסיות של מחשב :

מחשב



איור 1.1 – יחידותיו הבסיסיות של מחשב

נרחיב מעט על היחידות הבסיסיות :

יחידת העיבוד המרכזית היא ה"מוח של המחשב". היא מבצעת את ההוראות הכתובות בתוכניות המחשב, ביניהן הוראות לביצוע פעולות חישוב, פעולות השוואה ועוד. במהלך ביצוע פעולותיה פונה יחידת העיבוד המרכזית אל הזיכרון ואל אמצעי הקלט והפלט.

הזיכרון מורכב מאוסף גדול מאוד של תאים. לכל תא בזיכרון מותאם מספר סידורי, הנקרא מען (כתובת, address). יחידת העיבוד המרכזית משתמשת במידע השמור בזיכרון ולעתים משנה אותו, תוך פנייה לתאי הזיכרון על ידי המען שלהם. יש שני סוגי פניות לזיכרון : כתיבת מידע בזיכרון וקריאת מידע מהזיכרון. תהליך הכתיבה בזיכרון גורם למחיקת מידע ולשמירת מידע חדש במקומו. לעומתו, תהליך הקריאה מהזיכרון גורם להעברת המידע מהזיכרון אל יחידת העיבוד המרכזית, אך איננו גורם למחיקתו. הוא מזכיר את תהליך הקריאה שאנו מבצעים : כאשר אנו קוראים ספר, השורות אינן נעלמות מדפי הספר, אלא רק "מועתקות" לזיכרוננו.

המידע השמור בזיכרון מיוצג על ידי סיביות. **סיבית** (bit) – קיצור של **ס**פרה **בי**נארית (**b**inary dig**it**) – היא אחת מן הספרות 0 או 1, כמו שספרה עשרונית היא אחת מן הספרות 0, ..., 9.

תא בזיכרון המחשב מכיל סדרה של סיביות, בדרך כלל 16, 32 או 64 סיביות, על פי תכנון החומרה של המחשב. מפתיע ומרשים שכל המשימות המורכבות המתבצעות על ידי מחשב הן בסופו של דבר אוסף של פעולות על סדרות המורכבות מהספרות 0 ו-1! המחשב מפרש סדרות של סיביות כמספרים או כאותיות, לעתים סדרות של סיביות מפורשות כהוראות לביצוע או כטיפוסי מידע אחרים.

למעשה, יחידת העיבוד המרכזית, ה"מוח" של המחשב, מבצעת בסך הכול הוראות פשוטות כגון "קרא את סדרות הסיביות שנמצאות בתאי הזיכרון שמעניהם הם 13 ו-37, התייחס לכל סדרת סיביות כאל מספר, חבר אותם וכתוב את התוצאה בתא שמענו 116".

זיכרון המחשב מורכב למעשה משני חלקים נפרדים:

זיכרון ראשי (main memory) משמש לשמירת תוכניות בזמן ביצוען, ולשמירת נתונים ותוצאות ביניים של תוכניות שמתבצעות.

זיכרון משני (secondary memory) משמש לאחסון לזמן בלתי מוגבל של מידע ושל תוכניות מחשב.

הזיכרון הראשי קטן משמעותית מן הזיכרון המשני. מהירות הקריאה ממנו והכתיבה אליו גדולה הרבה יותר ממהירותן של פעולות אלו בזיכרון המשני. הזיכרון הראשי ממוקם צמוד ליחידת העיבוד המרכזית, ואילו הזיכרון המשני נמצא באמצעי אחסון חיצוניים כמו דיסק ודיסק קשיח. במהלך ביצועה של תוכנית מחשב, יחידת העיבוד המרכזית מעתיקה את התוכנית מהזיכרון המשני אל הזיכרון הראשי. במובנים רבים, ניתן להתייחס לזיכרון הראשי כאל זיכרון לטווח קצר, ואל הזיכרון המשני כאל זיכרון לטווח ארוך.

שאלה 1.2

הביאו דוגמה מכיתת בית הספר לחפץ שניתן לדמות את השימוש בו לשימוש בזיכרון ראשי, ולעומתה דוגמה לחפץ שניתן לדמות את השימוש בו לשימוש בזיכרון משני.

אמצעי קלט משמשים להעברת נתונים אל המחשב מן המשתמשים בו. למשל, בדרך כלל מחוברים למחשב אישי עכבר ולוח מקשים. שניהם אמצעי קלט המשמשים להעברת נתונים. גם סורק דיגיטלי, מצלמה דיגיטלית או מיקרופון המחוברים למחשב הם אמצעי קלט.

אמצעי פלט משמשים להעברת מידע מן המחשב אל המשתמשים בו. למשל, בדרך כלל מחוברים למחשב אישי מסך ומדפסת. גם מקרן או רמקול המחוברים למחשב הם אמצעי פלט.

שאלה 1.3

מחשבון נועד לבצע פעולות חשבון. מהו אמצעי הקלט למחשבון? מהו אמצעי הפלט למחשבון?

שאלה 1.4

המחשב קולט מידע, מעבד אותו ונותן כפלט את תוצאת העיבוד. גם מוח האדם קולט מידע, מעבד אותו ופולט את תוצאת העיבוד.

א. הביאו דוגמאות לאיברים קולטי מידע ולאיברים פולטי מידע בגוף האדם.

ב. הביאו דוגמאות למידע השמור בזיכרון האדם.

1.3 תוכנה

תוכנה היא אוסף תוכניות מחשב. תוכניות מחשב מנחות את העיבוד המתבצע על ידי החומרה. קיימות תוכניות לביצוע חישובים מתמטיים, לניהול מאגרי מידע, לבקרה על תהליכים, להדמיית מערכות, לעיבוד תמלילים, למשחקים ולישומים שונים ורבים נוספים.

בנוסף לתוכניות המיועדות ליישומים שונים, יש בכל מחשב תוכנית מיוחדת הנקראת מערכת הפעלה, והיא מהווה את הקשר בין התוכנה לחומרה:

מערכת הפעלה היא תוכנית המנהלת את שאר התוכניות ומקצה לשימושן את משאבי החומרה השונים (יע"מ, זיכרון, אמצעי קלט ואמצעי פלט).

כל תוכנית מחשב (או בקיצור, תוכנית) נכתבת על ידי מתכנת בשפת תכנות. נסביר את שני המושגים האלה:

שפת תכנות (programming language) היא למעשה אוסף של כל הכללים הקובעים כיצד נכתבות ההוראות בתוכנית מחשב, ומה המשמעות של כל הוראה.

מתכנת (programmer) הוא אדם הכותב תוכניות בשפת מחשב. עבודתו של מתכנת – תהליך התכנות – כולל ניתוח של משימות המיועדות לביצוע במחשב, כתיבת מתכון לביצוע המשימה, ויישומו של המתכון בשפת מחשב.

כזכור, הפעולות המתבצעות ביחידת העיבוד המרכזית הן פעולות על סיביות. לכן, בעצם, השפה שבאמצעותה ניתן לתקשר עם מחשב, או השפה שבה ניתן לתת לו הוראות שיוכל "להבין", צריכה להיות שפה מאוד פשוטה, הכוללת הוראות לביצוע פעולות על סיביות. שפה כזאת נקראת שפת מכונה:

שפת מכונה (machine language) היא שפת תכנות הכוללת הוראות לביצוע פעולות פשוטות מאוד. כל הוראה בשפת מכונה מורה על ביצוע פעולות על סדרות של סיביות (למשל, חיבור שתי סדרות). למעשה, גם ההוראות של שפת מכונה נכתבות בקוד המבוסס על סיביות, ולכן תוכנית בשפת מכונה היא בעצם סדרה ארוכה של סיביות, כלומר, רצף ארוך של 0 ו-1.

לכל סוג מחשב שפת מכונה משלו.

התוכניות הראשונות שנכתבו עבור מחשבים (בסוף שנות ה-40 של המאה העשרים) נכתבו בשפת מכונה. תהליך הכתיבה של תוכניות אלו היה מסורבל מאוד ולא נוח, בגלל החסרונות הרבים שיש לכתיבה בשפת מכונה.

? מהם החסרונות של כתיבה בשפת מכונה?

- מאחר שתוכנית בשפת מכונה היא רצף ארוך של 0 ו-1, קשה מאוד לכתוב אותה וקשה עוד יותר לקרוא אותה, לעקוב אחר מהלך ביצועה ולהבין את מטרתה. חשוב להבין שתהליך התכנות לא מסתיים בדרך כלל עם כתיבת התוכנית: לעתים מתגלות שגיאות בתוכנית וצריך לתקנה, לפעמים צריך לעדכן אותה כדי להתאימה לדרישות חדשות של המשימה שהיא

מבצעת. לא תמיד התיקונים והעדכונים מתבצעים על ידי הכותב המקורי, ולכן לנוחות הקריאה וההבנה של תוכנית נתונה יש חשיבות רבה.

- לכל סוג מחשב יש שפת מכונה שמתאימה בדרך כלל רק לו. לכן לא ניתן לקחת תוכנית שנכתבה בשפת מכונה של מחשב מסוג אחד, ולהריץ אותה כמו שהיא במחשב מסוג אחר. מעבר בין סוגי מחשבים דורש כתיבה מחודשת של התוכנית.

חסרונות אלה הביאו לפיתוחן של שפות נוחות יותר לכתיבה, לקריאה ולשימוש. שפות כאלו פותחו החל מאמצע שנות ה-50 של המאה העשרים, והן נקראות שפות עיליות. ביניהן למשל השפות פסקל (pascal), ג'אווה (Java), ו-C.

שפה עילית (high level language) היא שפת תכנות, אשר ההוראות בה דומות למשפטים בשפה טבעית (כמו אנגלית) או לנוסחאות מתמטיות. למרות הדמיון לשפה טבעית, ההוראות אינן נכתבות בכתיבה חופשית, אלא על פי כללים מוגדרים, שנקראים **כללי התחביר** (syntax) של השפה.

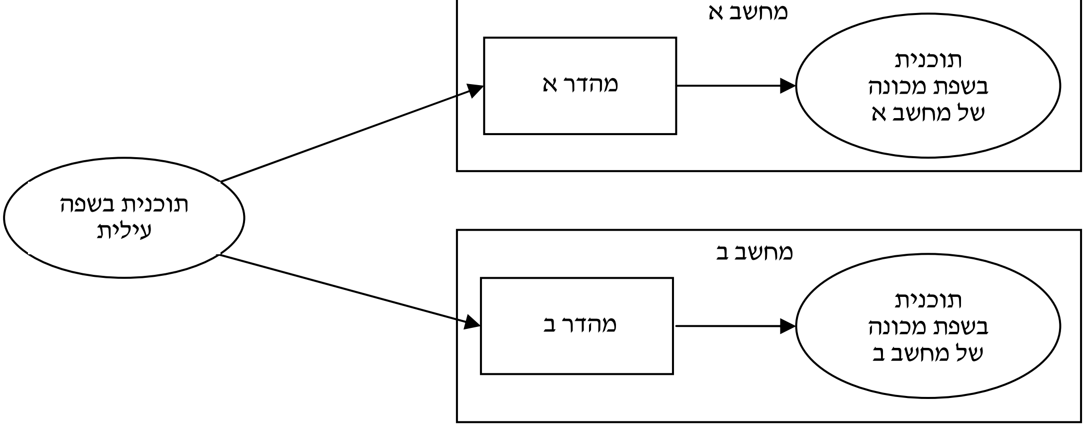
? אם המחשב "מבין" רק שפת מכונה, כיצד ניתן לגרום לו "להבין" תוכנית הכתובה בשפה עילית?

תוכנית הכתובה בשפה עילית עוברת תהליך של תרגום לשפת מכונה. התרגום נעשה על ידי תוכנית מחשב מיוחדת, הנקראת מהדר:

מהדר (compiler) היא תוכנית המתרגמת משפה עילית לשפת מכונה. תהליך התרגום נקרא **הידור** או **קומפילציה** (compilation). הקלט של המהדר הוא תוכנית בשפה עילית והפלט שלו הוא תוכנית בשפת מכונה, שהיא התרגום של תוכנית הקלט.

אם כך, כדי לבצע תוכנית מחשב הכתובה בשפה עילית צריכים להתבצע שני שלבים. קודם כל מתבצע שלב ההידור, במהלכו התוכנית בשפה העילית עוברת תרגום לתוכנית בשפת מכונה. רק אחר כך יכול להתבצע שלב ההרצה, במהלכו מתבצעת המשימה עצמה, כלומר, המחשב מבצע את ההוראות הכתובות בשפת מכונה, ונותן את תוצאת העיבוד כפלט.

לכל זוג של שפה עילית ושפת מכונה דרוש מהדר נפרד שיבצע את התרגום ביניהן. אם בכוונתנו להריץ במחשב מסוים תוכניות הכתובות בכמה שפות עיליות, עלינו לדאוג שבמחשב יהיה מותקן מהדר מתאים לכל אחת מהשפות העיליות האלו. גם ההיפך נכון: אם בכוונתנו להריץ תוכניות בשפה עילית מסוימת בכמה מחשבים מסוגים שונים, עלינו לדאוג שיהיו ברשותנו מהדר עבור כל סוג מחשב. איור 1.2 מדגים את הקשרים האלה.



איור 1.2 – הידור של תוכנית בשפה עילית במחשבים מסוגים שונים

אם כך, שפה עילית איננה רק יותר נוחה לקריאה ולכתיבה, אלא היא גם מגשרת על פני ההבדלים בין סוגים שונים של מחשבים. בעזרת מהדר מתאים ניתן לתרגם כל תוכנית בשפה עילית לתוכנית בשפת מכונה של מחשב זה או אחר.

תהליך ההידור מורכב משני שלבים:

1. בדיקת תחביר התוכנית בשפה העילית
2. תרגום התוכנית בשפה העילית לתוכנית בשפת מכונה.

בשלב 1, נערכת בדיקה כי התוכנית בשפה העילית עומדת בכללי התחביר של השפה שבה נכתבה. אם כתיבת התוכנית לא נעשתה בהתאם לכללי התחביר, יש בה **שגיאות תחביר** (syntax errors). הפלט של המהדר אחרי שלב 1 הוא פירוט שגיאות התחביר שמצא. לפני שניתן יהיה לתרגם את התוכנית יש לתקן את כל שגיאות התחביר שבה, כלומר, לעבור בהצלחה את שלב 1. למשל, תוכנית בשפת C# חייבת להכיל בתוכה לפחות פעם אחת את המילה `class`. אם ננסה לתרגם תוכנית בשפת C# שאינה מכילה את המילה `class` המהדר יודיע על שגיאת תחביר.

כאשר שלב 1 מסתיים בהצלחה, ואין בתוכנית שגיאות תחביר, מתבצע השלב השני ובו התוכנית מיתרגמת לשפת מכונה, ומתקבלת תוכנית שיכולה לרוץ במחשב.

גם במהלך הריצה של התוכנית עלולות להתגלות שגיאות שיגרמו לעצירת הריצה לפני סיומה המיועד, או להודעות שגיאה. אלו הן **שגיאות ריצה** (run-time errors), שאינן יכולות להתגלות בזמן ההידור. למשל, הניסיון לחלק ערך השמור בזיכרון המחשב ב-0 יגרום לשגיאת ריצה ולהדפסת הודעה מתאימה.

תהליך איתור שגיאות ריצה ותיקונן נקרא **ניפוי** (debugging). מקורו של המונח באנגלית בשלבים המוקדמים של שימוש במחשבים, כאשר מחשבים היו כה גדולים עד כי מחשב אחד מילא אולם שלם. מחשב מסוים חדל לפעול, ולאחר זמן התגלה בין רכיביו חרק גדול שנתקע שם והפריע לפעולתו התקינה של המחשב. מאז נוהגים לקרוא לשגיאה בתוכנית באג (bug – חרק באנגלית).

מאז תחילת פיתוח השפות העיליות, באמצע שנות ה-50 של המאה העשרים, פותח מספר גדול מאוד של שפות. עובדה זו מעוררת את השאלות הבאות:

- מדוע יש צורך בכל כך הרבה שפות?
- האם לא עדיפה שפה אחת אחידה שבה ייכתבו כל התוכניות, ואשר אותה יוכל כל אחד ללמוד בקלות?
- מה מבדיל בין השפות השונות?
- מי משתמש באילו שפות ולאילו מטרות?

לריבוי השפות שתי סיבות עיקריות. סיבה אחת קשורה להתפתחות שפות תכנות כתגובה לצרכים המתעוררים בשטחים חדשים ושונים של יישומים. לכל שפה מאפיינים ייחודיים משלה: יש שפות המיועדות בעיקר לחישובים מדעיים, יש אחרות המתאימות יותר לעיבוד נתונים מנהלי (הפקת משכורות, הנהלת חשבונות וכו'). הסיבה השנייה היא התקדמות המחקר המדעי העוסק בשפות תכנות ומסייע בשיפור השפות.

את שפות התכנות ניתן לחלק לקבוצות על פי העקרונות המנחים את הכתיבה בשפות אלו. בספר זה ללימוד היחידה "יסודות מדעי המחשב" נשתמש בשפת C#, השייכת לקבוצת השפות הקרויות **מונחות עצמים** (object oriented). קבוצות אחרות, שאליהן לא נתייחס ביחידה זו, הן קבוצת השפות ה**פרוצדורליות** (כמו פסקל או C), קבוצת השפות ה**פונקציונליות** (כמו scheme) וקבוצת השפות ה**לוגיות** (כמו פרולוג).

1.4 התפתחות המחשבים ומדעי המחשב

ההתפתחות הקשורה למחשבים נעשתה בשני מסלולים: ההתפתחות ההנדסית והטכנולוגית שאפשרה בניית מחשבים משוכללים יותר ויותר, וההתפתחות המדעית שניסתה להתמודד בצורה מדויקת, אפילו פורמלית לעתים, עם שאלות הקשורות לפתרון בעיות באמצעות מחשב. בסעיף זה ננסה לתת סקירה קצרה של שני מסלולי ההתפתחות, שכמובן אינם מנותקים זה מזה.

התפתחות הנדסית וטכנולוגית – חומרה

ציון דרך חשוב בהתפתחות ההנדסית הוא באמצע המאה ה-17. אז פיתח המתמטיקאי הצרפתי בלייז פסקל (Pascal) – על שמו נקראת שפת התכנות פסקל – מכונת חיבור וחיסור. את המכונה בנה עבור אביו, כדי לסכם סכומי כסף לצורך גביית מסים. פסקל בנה את המכונה כדי לעזור לאנשים שביצעו בדרך כלל את החישובים הדרושים, ובמיוחד כדי להקטין את מספר הטעויות שנגרמו על ידי החישובים האנושיים. המכונה של פסקל מהווה ציון דרך חשוב משום שזו הייתה הפעם הראשונה ששולב מרכיב אוטומטי, באמצעות מכונה, בפעולת חישוב.

המכונה של פסקל זכתה לשיפורים כמה עשרות שנים מאוחר יותר. המדען הגרמני וילהלם לייבניץ (Leibnitz) בנה אף הוא מכונת חישוב. הוא העתיק את מנגנוני החיבור והחיסור של מכונתו מהמכונה של פסקל, אך הוסיף גם חלק שמבצע כפל וחילוק. לייבניץ בנה את המכונה שלו משום שלטענתו המדע אמנם לא יכול להתקיים ללא חישוב, אך חישוב הוא פעולה חוזרת על עצמה, משעממת ולא יצירתית, וצריך להעביר את ביצועה למכונות.

בתחילת המאה ה-19, ב-1801, פיתח גם הצרפתי ז'וזף ז'אקאר (Jacquard) מכונה לביצוע אוטומטי של משימות. בניגוד למכונות של פסקל ולייבניץ אלו לא היו משימות חישוב מספריות, אלא משימות אריגה. מכונתו של ז'אקאר היתה למעשה נול אריגה מתוחכם, שיכול היה לארוג במגוון של דוגמאות. הדוגמאות השונות תוארו על ידי כרטיסים מנוקבים, לכל דוגמת אריגה תבנית ניקובים משלה. מנגנון בקרה מיוחד בתוך המכונה חש את הנקבים בכרטיס, ובהתאם לכך פיקח על פעולות המכונה, כגון בחירת חוטים.

התכנון של מה שנחשב היום המחשב הראשון הגיע כשלושים שנה מאוחר יותר. ב-1833, תוכננה לראשונה מכונה כללית יותר, שיכולה לבצע משימות מסוגים שונים. המתמטיקאי האנגלי צ'רלס בבג' (Babbage) תכנן את "המכונה האנליטית" שלו, מכונה שהיתה אמורה לבצע תוכניות שונות מסוגים שונים למטרות שונות. התוכניות היו אמורות להיות מקודדות, בדומה למכונה של ז'אקאר, בעזרת כרטיסים מנוקבים. התכנון של בבג', שכלל צירים, ידיות, גלגלי שיניים ורכיבים מכניים אחרים, לא מומש אף פעם. בניית חלקי המכונה דרשה דיוק טכני רב מדי, שלא ניתן היה להשגה באותו זמן. אבל, אף על פי שלא נבנתה, המכונה האנליטית היא בעלת חשיבות גדולה. למעשה, הרעיונות הגלומים בה הם הבסיס למבנה המחשבים של ימינו ולאופן פעולתם. אפילו בעצם נכתבה אז תוכנית עבור המכונה הלא-בנויה, תוכנית הראויה בהחלט להיחשב כתוכנית המחשב הראשונה בהיסטוריה! את התוכנית כתבה עדה לאבלייס (Lovelace) – על שמה נקראת שפת Ada.

לאחר מותו של בבג' מרכז הפעילות בבניית מכונות חישוב נדד לארצות הברית. העיסוק המוגבר בכך בארצות הברית נבע בין השאר מהצורך שהתעורר מהשטח: ב-1880 נערך בארצות הברית מפקד אוכלוסין, המפקד הבא נועד ל-1890, וב-1886, ארבע שנים לפני המפקד הבא, ושש שנים אחרי המפקד הקודם, עדיין לא סיימו לסכם את תוצאותיו, ומועד סיום הסיכום לא נראה באופק... הרמן הולרית (Hollerith), מהנדס שעבד כפקיד בלשכת מפקד התושבים, גילה יוזמה וב-

1886 הוא הציע כמה מכשירים שפיתח כדי לפתור את הבעיה. סיכום המפקד של 1890 כבר נעשה בעזרת פיתוחו של הולריק וארך לא יותר מחודש! גם פיתוחו של הולריק היה מבוסס על כרטיסים מנוקבים. בעקבות הצלחתו ראה הולריק כי טוב והקים חברה למכונות חישוב. מאוחר יותר, ב-1928, הרחיבה אותה חברה את פעילותה והפכה לחברה בינלאומית למכונות עסקיות, ושינתה את שמה בהתאם ל-International Business Machines, או בקיצור, IBM, המוכרת לנו היטב גם היום.

בשנת 1937 החל המדען האמריקני הווארד אייקן (Aiken), יחד עם חברת IBM, לבנות את המחשב האלקטרו-מכני הראשון. למעשה, אייקן הגשים את חלומו של בבג׳: התכנון שלו התבסס על רעיונותיו של בבג׳, ונעזר במכשירים החשמליים והאלקטרו-מכניים אשר בתקופתו של אייקן כבר היו זמינים. בניית המחשב הושלמה ב-1944. שמו היה MARK I וגודלו היה כגודל אולם התעמלות! למעשה, במקביל לבניית MARK I, במהלך מלחמת העולם השנייה, בנו גם הבריטים מחשב, בשם אניגמה (Enigma), שבעזרתו פיצחו צפנים של הגרמנים. אלא שעקב תפקידו הרגיש נשמר קיומו של Enigma בסוד במשך זמן רב. זמן קצר אחר-כך, ב-1946, כבר הושלמה בנייתו של מחשב מהיר בהרבה מ-MARK I. הוא נקרא ENIAC, ותוכנן על ידי האמריקנים אקרט (Eckert) ומוצ׳לי (Mauchly). הוא היה הרבה יותר מהיר משום שלא התבסס בכלל על תנועות מכניות, ולכן נחשב המחשב האלקטרוני הראשון. אורכו היה שלושים מטרים, רוחבו מטר אחד, גובהו שלושה מטרים, והוא הכיל 18,000 שפופרות ואקום. צריכת החשמל של ה-ENIAC, שהוצב בעיר פילדלפיה, הייתה כה גבוהה, עד שנהגו אז לומר שבכל פעם שהופעל התעממו האורות בעיר כולה!

אותם מחשבים ראשונים השתמשו בסרטי נייר מנוקבים. עבור כל עיבוד קודדו על סרט כזה גם התוכנית לביצוע וגם נתוני הקלט עבורה. כלומר, אם ביצעו אותה תוכנית כמה פעמים, היה צריך להזין למחשב את סרט הנייר שעליו קודדה בכל פעם ופעם. ב-1946 הציע המדען ההונגרי-אמריקני ג׳ון פון-נוימן (von Neuman) לשמור את התוכניות בזיכרון המחשב. כתוצאה מכך, מהירות תהליכי העיבוד השתפרה משמעותית. המחשב התעשייתי הראשון שפעל לפי עקרון זה נבנה ב-1951, ונקרא UNIVAC. עקרון זה של פון-נוימן משמש למעשה גם במחשבים של ימינו.

מכאן החלה האצה בקצב ההתפתחות הטכנולוגית. בשנות ה-50 וה-60 של המאה העשרים הוחלפו שפופרות הריק בטרנזיסטורים, ושינוי זה הביא להקטנה בממדי המחשבים, להקטנה בצריכת ההספק החשמלי שלהם ולהגדלה במהירות פעולתם. בנוסף למחשבים הגדולים (mainframes) שהיו עד אז, נבנו גם מחשבים בינוניים (שגודלם כגודל כוננית ספרים) שנקראו מחשבי מידי ומחשבי מיני (midi/mini computers). זמן לא רב אחר-כך קטנו המחשבים אף יותר: בסוף שנות ה-60 ובתחילת שנות ה-70 פותחה טכנולוגיית המעגל המשולב (integrated circuit) ובעקבות כך ירד מחיר המחשבים, מהירותם עלתה ונבנו אפילו מחשבים זעירים, שגודלם לא עלה על גודל קופסת גפרורים – המיקרו-מחשבים (micro computers). משום שהיו כה קטנים וזולים יחסית, החלו להשתמש בהם הרבה לצרכים מגוונים למשל כבקרים במערכות אלקטרוניות שונות. המחשב האישי, המוכר לנו היום, נבנה לראשונה בסוף שנות ה-70 והוא היה מבוסס על מיקרו-מחשב. קפיצת הדרך הזאת, שנעשתה במשך תקופה קצרה יחסית, היא כמעט בלתי נתפסת: היום יש מיקרו מחשבים שכושר העיבוד שלהם עולה בהרבה על זה של אותם מחשבי הענק הראשונים.

התפתחות הנדסית וטכנולוגית – תוכנה

במקביל להתפתחות הטכנולוגית של החומרה, המכונות עצמן, חלה התפתחות גם בתחום התוכנה. המחשבים הראשונים תוכנתו בשפת מכונה. כפי שהזכרנו, כתיבת תוכניות כאלו הייתה כרוכה באי-נוחות רבה. אי-נוחות זו הביאה באמצע שנות ה-50 לפיתוחה של שפת התכנות העילית

הראשונה, פורטרן (Fortran). משום שבאותה תקופה הייתה עלייה בביקוש לתוכניות מחשב המבצעות חישובים מתמטיים, פורטרן הותאמה לכתיבה של חישובים כאלה. אבל היא לא הייתה נוחה לכתיבת תוכניות לניהול מאגרי מידע (ניהול כוח אדם, ניהול מלאי וכו'), ומשום כך פותחה בעקבותיה, בתחילת שנות ה-60, שפת קובול (Cobol). באותה תקופה פותחה שפה נוספת, ליספ (Lisp) שהתאימה לצרכים אחרים. ממנה נגזרה מאוחר יותר שפת לוגו (LOGO) המשמשת בדרך כלל כשפה לימודית לפיתוח הרגלי חשיבה בפתרון בעיות.

מאז פותחו שפות עיליות רבות לצרכים שונים ומגוונים. למשל, שפת בייסיק (Basic) פותחה באמצע שנות ה-60, למטרות לימודיות. פסקל (Pascal) פותחה אף היא כשפה לימודית, בתחילת שנות ה-70, ובאותה תקופה פותחה גם שפת C, שנועדה לכתיבת מערכות הפעלה. ב-1970 פותחה שפת פרולוג (PROLOG) שגישת התכנות בה מבוססת על כללים לוגיים. עם עליית הצורך בכתיבת תוכניות גדולות מאוד ומורכבות מאוד, פותחו בשנות ה-80 שפות שנועדו במיוחד לפיתוח תוכניות גדולות, כגון Modula ו-Ada. בעקבותיהן פותחו שפות נוספות שהתמקדו בפיתוח תוכניות גדולות, והתבססו על מה שקרוי תכנות מונחה-עצמים. בין אלו ניתן למצוא את ++C, את SmallTalk, את Ada95, את Eiffel ואת Java.

התפתחות מדעית

כאמור, במקביל להתפתחות ההנדסית והטכנולוגית, הן בתחום החומרה והן בתחום התוכנה, חלה גם התפתחות מדעית. החל מאמצע שנות ה-30 של המאה העשרים (עוד לפני שנבנה המחשב הראשון!), נעשתה עבודה תיאורטית חשובה, שהניחה את הבסיס לתחום המדעי הקרוי היום מדעי המחשב. עבודה זו נעשתה על ידי מתמטיקאים, שניסו להגדיר בצורה מתמטית מהו תהליך של חישוב, ולנתח בצורה מתמטית, פורמלית ומדוייקת, את מגבלותיהם של תהליכי חישוב, ובפרט את מגבלותיהן של מכונות המבצעות תהליכי חישוב. בין המתמטיקאים האלה ניתן למנות את אלן טיורינג (Turing) האנגלי, שהיה מעורב מאוחר יותר בפרויקט האניגמה, את קורט גדל (Gödel) הגרמני, את אנדריי מרקוב (Markov) הרוסי, ואת האמריקנים אלונזו צ'רץ' (Church), אמיל פוסט (Post) וסטיבן קלין (Kleene). חוקרים אלה ידעו להצביע כבר אז על כך שיש בעיות חישוביות שלא יצליחו לעולם להיפתר על ידי מכונה חישובית, וזאת כאמור עוד לפני שנבנה המחשב הראשון.

מאז חלה התפתחות מדעית עצומה, כאשר לעתים ההתפתחויות הטכנולוגיות הניעו וזירזו התפתחויות מדעיות ולעתים דווקא ההתפתחויות המדעיות גרמו להתפתחות טכנולוגית. כיום קיימת מחלקה למדעי המחשב כמעט בכל מוסד אקדמי, והפעילות המחקרית במדעי המחשב היא רבה ומגוונת: תורת החישוביות העוסקת באפיון של בעיות שניתנות או לא ניתנות לפתרון; תורת הסיבוכיות העוסקת באפיון של בעיות על פי כמות המשאבים (זמן וזיכרון) הנדרשים לפתרונן; קריפטוגרפיה העוסקת בהצפנות מסוגים שונים; חישוב מקבילי ומבוזר, העוסק בפתרון בעיות שנועדו להתבצע במערכות שבהן כמה מחשבים עובדים ביחד לפתרון משימה אחת; תורת התקשורת העוסקת באפיונים של רשתות תקשורת (כמו האינטרנט) ופתרון בעיות הקשורות לרשתות תקשורת; בינה מלאכותית העוסקת במערכות שנועדו לדמות פעילות אנושית, ועוד תחומים רבים נוספים.

במסגרת לימודי מדעי המחשב בבית הספר התיכון ניתן כמובן להציג רק מקצת מתחומי הפעילות השונים במדעי המחשב. בכל זאת תוכנית הלימודים התיכונית במדעי המחשב נוגעת במגוון רחב למדי של נושאים, גם בחלק מאלה שהוזכרו לעיל.

סיכום

בפרק זה תיארנו בקצרה מהו מחשב, מהן היחידות הבסיסיות שמהן הוא בנוי, וכיצד נכתבות ומתבצעות תוכניות מחשב. תיארנו גם את ההתפתחות ההנדסית והטכנולוגית של המחשבים ואת ההתפתחות המדעית של תחום מדעי המחשב.

מחשב הוא מכונה אלקטרונית הקולטת נתונים, מעבדת אותם ופולטת מידע שנוצר בתהליך העיבוד.

הנתונים שקולט המחשב נקראים **קלט**.

המידע שפולט המחשב נקרא **פלט**.

העיבוד המבוצע במחשב מונחה על ידי קבוצת הוראות הנקראת **תוכנית מחשב**.

הרכיבים הפיסיים של המחשב נקראים **חומרה** ואוסף תוכניות המחשב נקרא **תוכנה**. החומרה מחולקת לכמה רכיבים בסיסיים: יחידת עיבוד מרכזי, זיכרון, אמצעי קלט ואמצעי פלט. תוכנה של מחשב כוללת בין השאר את מערכת ההפעלה, את המהדרים ותוכניות ליישומים שונים.

יחידת העיבוד המרכזית מנהלת את כל התהליכים המתבצעים במחשב.

הזיכרון שומר מידע, תוכניות, ותוצאות ביניים של תהליכי עיבוד. הזיכרון מתחלק ל**זיכרון משני** ול**זיכרון ראשי**. המידע השמור בזיכרון מיוצג באמצעות סיביות (**סיבית** – אחת מן הספרות 0 או 1).

אמצעי הקלט אחראים על קליטת נתונים ו**אמצעי הפלט** אחראים על פליטת מידע.

תוכנית מחשב נכתבת על ידי **מתכנת** בשפת תכנות. ל**שפת תכנות** כללים המכתיבים את אופן כתיבת ההוראות בתוכנית. כיום נכתבות תוכניות מחשב בשפה עילית.

ב**שפה עילית** המשפטים דומים למשפטים בשפה טבעית, כמו אנגלית. **שפת מכונה** היא השפה אותה "מבין" המחשב, וההוראות בה מקודדות על ידי סיביות.

לפני ביצוע תוכנית בשפה עילית עליה לעבור שני שלבים: הידור והרצה. **הידור** (קומפילציה) הוא התהליך של תרגום תוכנית בשפה עילית לשפת מכונה. תהליך זה מתבצע על ידי **מהדר** (קומפיילר).

שגיאות תחביר מתגלות בשלב ההידור. **שגיאות ריצה** מתגלות בזמן ההרצה. תהליך איתור שגיאות ריצה ותיקונן נקרא **ניפוי שגיאות**.

שאלות נוספות

1. מדוע אי-אפשר לכתוב תוכנית למחשב בשפה טבעית, כמו אנגלית או עברית?
2. מדוע שפת תכנות עילית נקראת בשם זה?
3. למה מתכוונים כאשר אומרים כי מחשב מבין שפת מכונה?
4. כמה מהדרים דרושים להידור תוכנית בשפת C, תוכנית בשפת פסקל, ותוכנית בשפת בייסיק בשלושה מחשבים מסוגים שונים?

פרק 2 – פתרון בעיות אלגוריתמיות

פרק זה עורך הכרה עם הנושא שבו נעסוק למעשה במהלך כל לימוד היחידה: פתרון בעיות אלגוריתמיות. נכיר את המושג אלגוריתם, מושג מרכזי וחשוב במדעי המחשב, שמשמעותו למעשה פתרון לבעיה, פתרון שאפשר אחר כך לתרגמו לתוכנית מחשב. לאחר מכן נערוך היכרות ראשונית עם מושג התבנית. גם תבנית היא אלגוריתם, כלומר פתרון לבעיה, אך היא יכולה גם לשמש כתת-פתרון לבעיות רבות בעלות מאפיינים משותפים.

2.1 אלגוריתמים

המושג **אלגוריתם** שנתמקד בו בפרק זה, הוא מושג מרכזי במדעי המחשב. בפרק זה נכיר ונפתח אלגוריתמים ראשונים, אשר אינם מיועדים לביצוע במחשב. בפרק הבא ובפרקים הבאים אחריו נפתח אלגוריתמים המיועדים ליישום בתוכניות מחשב, לביצוע במחשב.

קבוצת ההוראות שבדוגמה הבאה היא אלגוריתם:

1. *הרתח עשר כוסות מים*
2. *הוסף קורט מלח*
3. *הוסף חצי קילו פתיתים למים הרותחים*
4. *הבא את המים לרתיחה נוספת*
5. *בשל את הפתיתים למשך 20 דקות על אש קטנה*
6. *סנן את הפתיתים*

האלגוריתם שבדוגמה זו הוא מתכון לבישול חצי קילו פתיתים.

הנה דוגמה נוספת לאלגוריתם:

1. *בחר מספר שלם חיובי*
2. *חבר את ספרות המספר*
3. *חלק את התוצאה ב-3*
4. *כתוב את שארית החלוקה*

שאלה 2.1

מהי תוצאת ביצוע האלגוריתם שלעיל עבור המספר 1977?

באופן כללי ניתן לומר כי אלגוריתם הוא מתכון אך לאו דווקא לבישול:

אלגוריתם הוא מתכון לביצוע משימה. אלגוריתם מורכב תמיד מקבוצת הוראות **חד-משמעיות** ו**אפשריות לביצוע**, אשר **סדר ביצוען** מוגדר היטב.

המילים המודגשות בהגדרה שלעיל הן שלושת המאפיינים החשובים לאלגוריתם: חד-משמעיות, אפשריות לביצוע וסדר ביצוען מוגדר היטב.

חד-משמעיות: הוראה המופיעה באלגוריתם חייבת להיות חד-משמעית. כלומר, כל ביצוע שלה צריך להסתיים תמיד באותה תוצאה. כך, למשל, ההוראה השנייה בדוגמה האחרונה, *חבר את ספרות המספר*, היא חד-משמעית. לעומתה, ההוראה *צוז קצת הצידה* אינה חד-משמעית: אנשים שונים יזוזו לפיה למקומות שונים, וייתכן גם כי אותו אדם יזוז לפיה אחרת בפעמים שונות.

אפשר לבצע: הוראה באלגוריתם צריכה להיות אפשרית לביצוע, ובפרט, עליה להתאים למבצֵעַ המיועד שלה. למשל טבח יכול לבצע את ההוראה *הרתח 10 כוסות מים*, אך אינו יכול לבצע את ההוראה *הנחת את המטוס*.

סדר ביצוע: סדר ביצוע הוראות האלגוריתם הוא לפי סדר הופעתן, אם לא נאמר אחרת. הביצוע של האלגוריתם מסתיים כאשר אין יותר הוראות. כאשר ביצוע של הוראה מסתיים, ממשיכים להוראה הבאה.

צורת הכתיבה שאנו כותבים בה אלגוריתמים נקראת כתיבה בפסאודו-קוד. **פסאודו-קוד** (קוד מדומה, pseudo-code) של אלגוריתם הוא ייצוג או כתיבה של האלגוריתם בדרך דמוית שפת תכנות, כלומר, במילים ובמשפטים בשפה חופשית אבל ברורה וחד-משמעית.

המונח "אלגוריתם" נגזר ככל הנראה משמו של המתמטיקאי מוחמד אל-חואריזמי, שהשתבש לאל-גואריזמי. אל-חואריזמי חי במאה ה-9 לספירה, באזור חואריזם, אשר נמצא היום באוזבקיסטן. אל-חואריזמי היה הראשון שניסח את הכללים המשמשים אותנו עד היום לביצוע ארבע פעולות החשבון הבסיסיות. במאה ה-14 החל המונח "אלגוריתם" להיות שגור בפי המתמטיקאים, ככינוי ל"מתכון מתמטי". מתכונים מתמטיים כאלו הם למשל אלגוריתם להכפלת שני מספרים ("כפל ארוך"), אלגוריתם להעלאה בחזקה של מספר אחד באחר, אלגוריתם למציאת המחלק המשותף הגדול ביותר של שני מספרים שלמים חיוביים (האלגוריתם של אוקלידס).

אנו נתמקד באלגוריתמים כפתרון ל**בעיות אלגוריתמיות**:

בעיה אלגוריתמית היא בעיה אשר נתונות בה נקודת מוצא ומטרה ונדרש אלגוריתם שלאחר ביצועו מגיעים מנקודת המוצא אל המטרה.

כלומר, בעיה אלגוריתמית מגדירה למעשה משימה: הגעה מנקודת המוצא הנתונה אל המטרה הנתונה. אלגוריתם הפותר את הבעיה הוא מתכון לביצוע המשימה הזאת. הנה דוגמה לבעיה אלגוריתמית:

בעיה 1

מטרת הבעיה ופתרונה: הצגת בעיה אלגוריתמית ראשונה, והצגת אלגוריתם לפתרונה.

שייט נמצא על גדת נהר ועמו כרוב, כבש וזאב. הוא רוצה לעבור לגדה השנייה בסירה קטנה ולהעביר את השלושה. הסירה יכולה להכיל בו-זמנית רק את השייט ועוד אחד מבין שלושת הפריטים שאיתו. השייט אינו יכול להשאיר את הזאב ואת הכבש ביחד או את הכבש ואת הכרוב ביחד ללא השגחתו.

פתחו אלגוריתם שינחה את השייט כיצד להעביר את הכבש, את הזאב ואת הכרוב מהגדה האחת אל האחרת.

בעיה זו מגדירה משימה של חציית נהר באילוצים שונים. נקודת המוצא היא המצב שהשייט, הזאב, הכבש והכרוב נמצאים על גדה אחת (גדה א) של הנהר. המטרה היא המצב שהשייט, הזאב, הכבש והכרוב נמצאים על הגדה האחרת (גדה ב).

נציג אלגוריתם אשר מורה כיצד לבצע את המשימה המתוארת בבעיה. בצד הוראות האלגוריתם מוצג מעקב אחר מהלך ביצועו.

אלגוריתם לפתרון בעיה 1

	גדה א	גדה ב
(נקודת המוצא) ←	שייט, כרוב, כבש, זאב	
1. הפלג מגדה א לגדה ב עם הכבש	כרוב, זאב	שייט, כבש
2. הפלג מגדה ב לגדה א לבד	שייט, כרוב, זאב	כבש
3. הפלג מגדה א לגדה ב עם הזאב	כרוב	שייט, כבש, זאב
4. הפלג מגדה ב לגדה א עם הכבש	שייט, כרוב, כבש	זאב
5. הפלג מגדה א לגדה ב עם הכרוב	כבש	שייט, כרוב, זאב
6. הפלג מגדה ב לגדה א לבד	שייט, כבש	כרוב, זאב
7. הפלג מגדה א לגדה ב עם הכבש		שייט, כרוב, כבש, זאב

? האם זהו האלגוריתם היחיד הפותר את בעיה 1?

לא. למשל גם האלגוריתם הבא הוא פתרון לבעיה 1:

1. *הפלג מגדה א לגדה ב עם הכבש*
2. *הפלג מגדה ב לגדה א לבד*
3. *הפלג מגדה א לגדה ב עם הכרוב*
4. *הפלג מגדה ב לגדה א עם הכבש*
5. *הפלג מגדה א לגדה ב עם הזאב*
6. *הפלג מגדה ב לגדה א לבד*
7. *הפלג מגדה א לגדה ב עם הכבש*

שימו ♥: במקרים רבים קיים יותר מאלגוריתם אחד הפותר בעיה אלגוריתמית נתונה.

שאלה 2.2

עקבו באמצעות טבלה מתאימה אחר מהלך ביצוע האלגוריתם הנוסף לפתרון בעיה 1.

סוף פתרון בעיה 1

בעיה 1 היא בעיה ספציפית העוסקת בפריטים מסוימים (כרוב, כבש, זאב). אפשר להרחיב אותה לבעיה כללית שנתונים בה שלושה פריטים כלשהם, הנמצאים בגדה א, ויש להעבירם לגדה ב, בשמירה על האילוצים שבבעיה 1. את הבעיה הכללית אפשר לפתור באמצעות אותם אלגוריתמים שפותרים את בעיה 1, אך צריך יהיה להחליף בהם כל התייחסות ספציפית לכרוב, לכבש ולזאב ב"פריט 1", ב"פריט 2", וב"פריט 3" בהתאמה. כך אפשר יהיה להשתמש באותו אלגוריתם כדי לפתור את הבעיה עבור כל שלושה פריטים המתאימים לאילוצים. למשל, עבור כלב, חתול ועכבר, או עבור אריה, פרה וחציר. גם במקרה של כלב, חתול ועכבר, לא ניתן להשאיר את פריט 1 (כלב) ואת פריט 2 (חתול) יחד ללא השגחה, וגם לא את פריט 2 (חתול) ואת פריט 3 (עכבר). כך גם לגבי המקרה שפריט 1 הוא אריה, פריט 2 הוא פרה ופריט 3 הוא חציר.

במדעי המחשב עוסקים בדרך כלל בבעיות אלגוריתמיות, אשר לכל אחת מהן אפשרויות רבות לנקודת מוצא. אלגוריתם הפותר בעיה אלגוריתמית שעבורה כמה נקודות מוצא, צריך להיות כללי ולפתור את הבעיה עבור כולן.

שאלה 2.3

על לוח בן חמש משבצות מונחות שתי אבני משחק מצבע אחד ושתי אבני משחק מצבע אחר בנקודת המוצא הבאה:

○	○		●	●

המטרה היא להביא את אבני המשחק למצב:

●	●		○	○

הפעולות המותרות הן: העברת אבן משחק למשבצת סמוכה פנויה, והקפצת אבן משחק מצבע אחד מעל לאבן משחק מצבע אחר אל משבצת פנויה.

פתחו אלגוריתם להעברת אבני המשחק מנקודת המוצא אל מצב המטרה.

שימו ♥: בבעיה זו מספר רב של אפשרויות לנקודת המוצא, כיוון שכל זוג צבעים שונים של אבני המשחק מכתיב למעשה נקודות מוצא שונות. האלגוריתם שתפתחו יתאים לכל זוג צבעים, ולכן יהיה כללי.

בפתרון בעיה 1 פיתחנו אלגוריתם, אשר בכל ביצוע שלו מבוצעות כל הוראותיו, זו אחר זו, לפי סדר הופעתן. לעתים יש צורך באלגוריתמים אשר ביצוע חלק מההוראות בהם הוא מותנה. נראה זאת בבעיה הבאה.

בעיה 2

מטרת הבעיה ופתרונה: הצגת אלגוריתם שיש בו הוראה לביצוע-בתנאי.

שני מְכָלים, מכל א ומכל ב, מכילים מספר שונה של תפוזים. סכום מספרי התפוזים בשני המכלים הוא זוגי.

פתחו אלגוריתם להעברת תפוזים בין המכלים, כך שתתבצע רק העברה אחת של מספר תפוזים, ולאחר ההעברה יכילו המכלים מספר שווה של תפוזים.

שימו ♥: יש אפשרויות רבות לנקודת המוצא, כי יש אפשרויות רבות לזוגות מספרי תפוזים שונים זה מזה שסכומם זוגי.

שאלה 2.4

באלגוריתם לפתרון יהיה צריך לציין כמה תפוזים יש להעביר בין המכלים כדי שמספרי התפוזים בשני המכלים יהיו שווים. לפניכם זוגות של מספרי תפוזים. בכל זוג המספר השמאלי מציין את מספר התפוזים במכל א, והמספר הימני מציין את מספר התפוזים במכל ב. עבור כל זוג, חשבו כמה תפוזים יש להעביר מהמכל המלא יותר למכל הפחות מלא.

א. 160 100

ב. 935 971

ג. 1001 1

בעזרת התשובה לשאלה 2.4 ניתן להיווכח שמספר התפוזים להעברה הוא חצי מההפרש בין מספרי התפוזים שבמכלים. צריך לחשב מספר זה לפני ההעברה, ולכן ההוראה הראשונה

באלגוריתם לפתרון הבעיה תהיה:

חשב את מספר התפוזים להעברה: חצי מההפרש בין מספרי התפוזים שבמכלים

אם במכל א יש יותר תפוזים, צריך לבצע את ההוראה הבאה:

העבר תפוזים ממכל א למכל ב על פי המספר המחושב

ואם במכל ב יש יותר תפוזים, צריך לבצע את ההוראה הבאה:

העבר תפוזים ממכל ב למכל א על פי המספר המחושב

? כיצד נבחר איזו משתי ההוראות לבצע?

בחירת ההוראה המתאימה לביצוע תיקבע על פי תנאי. התנאי המתאים הוא:

במכל א יש יותר תפוזים מאשר במכל ב

אם התנאי יתקיים, תתבצע ההוראה הראשונה. אחרת, תתבצע ההוראה השנייה. באלגוריתם לפתרון הבעיה ננסח את ההתניה הזאת על ידי הוראה לביצוע-בתנאי באופן הבא:

אלגוריתם לפתרון בעיה 2

1. *חשב את מספר התפוזים להעברה: חצי מההפרש בין מספרי התפוזים שבמכלים*
2. ***אם*** *במכל א יש יותר תפוזים מאשר במכל ב*
   2.1. *העבר תפוזים ממכל א למכל ב על פי המספר המחושב*
3. ***אחרת***
   3.1. *העבר תפוזים ממכל ב למכל א על פי המספר המחושב*

סוף פתרון בעיה 2

ההוראה **אם ...**
.
.
.
אחרת
.
.
.

היא **הוראה לביצוע-בתנאי**, המורה על ביצוע קבוצת הוראות אחת או קבוצת הוראות אחרת על פי תנאי.

הוראה לביצוע-בתנאי היא הוראת בקרה. **הוראת בקרה** היא הוראה שמשפיעה על מהלך ביצוע ההוראות באלגוריתם.

שאלה 2.5

על השולחן מונחות שלוש מעטפות בשורה, בכל מעטפה יש פתק ועליו רשום מספר. במעטפה אחת יש פתק שעליו רשום המספר 0, ובשתי המעטפות האחרות יש פתקים שרשומים עליהם מספרים שונים מ-0. המעטפה שבה נמצא הפתק שעליו רשום 0 איננה המעטפה האמצעית בשורה.

א. יש אינסוף אפשרויות לנקודת המוצא, כי על שניים מהפתקים רשומים מספרים כלשהם שונים מ-0. תארו חמש אפשרויות שונות של נקודת המוצא.

ב. פתחו אלגוריתם שמטרתו היא לשים באמצע, בין שתי המעטפות האחרות, את המעטפה עם הפתק שעליו רשום 0. הפעולות שבהן יש להשתמש לביצוע המשימה הן: קריאת המספר הרשום על המעטפה, והחלפת מקומות בין מעטפות שכנות.

בפתרון בעיה 2 ראינו אלגוריתם שכלל ביצוע הוראות בתנאי. לעתים יש צורך באלגוריתמים אשר ביצוע חלק מההוראות בהם חוזר כמה פעמים. נראה זאת בבעיה הבאה.

בעיה 3

מטרת הבעיה ופתרונה: הצגת אלגוריתם שבו קיימת הוראה לביצוע-חוזר.

על השולחן יש קלפים המסודרים בשורה. מספר הקלפים בשורה הוא אי-זוגי, והקלף האמצעי הוא לבן. כל הקלפים שמשמאל לקלף הלבן הם שחורים וכל הקלפים שמימינו אדומים. נתונות גם שתי סימניות: סימניה-1 המוצבת על הקלף שבקצה השמאלי וסימניה-2 המוצבת על הקלף שבקצה הימני.

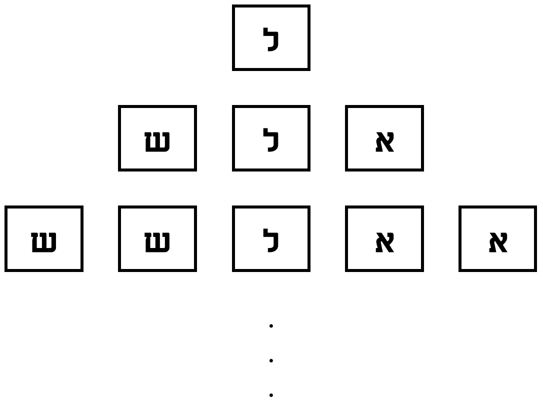
פתחו אלגוריתם אשר מסדר את שורת הקלפים מחדש כך שכל הקלפים האדומים יהיו משמאל לקלף הלבן וכל השחורים מימינו.

הפעולות המותרות לביצוע הן:

- ♦ הצבת סימניה מימין או משמאל לקלף שעליו היא מוצבת. הצבת סימניה כוללת קריאת צבע הקלף שעליו היא מוצבת.
- ♦ החלפה זה בזה של מקומות הקלפים שעליהם מוצבות הסימניות.

נקודת המוצא של הבעיה היא המצב שמונחות על השולחן שורת הקלפים ושתי הסימניות. המטרה היא הסידור החדש, שמוחלפים בו מקומותיהם של הקלפים האדומים ושל הקלפים השחורים.

שימו ♥: יש אינסוף אפשרויות לנקודת המוצא כי מספר הקלפים בשורה יכול להיות כל מספר אי-זוגי, למשל (**א** מסמן קלף אדום, **ל** מסמן קלף לבן ו-**ש** מסמן קלף שחור):



האלגוריתם המבוקש אינו צריך להיות תלוי באורך שורת הקלפים הנתונה, כלומר ביצועו צריך להשיג את המטרה לכל נקודת מוצא אפשרית.

כדי לפתח אלגוריתם לפתרון הבעיה, נחשוב תחילה על מקרה פרטי של הבעיה. נניח כי בשורה שבעה קלפים. כלומר, נקודת המוצא היא:

ש	ש	ש	ל	א	א	א

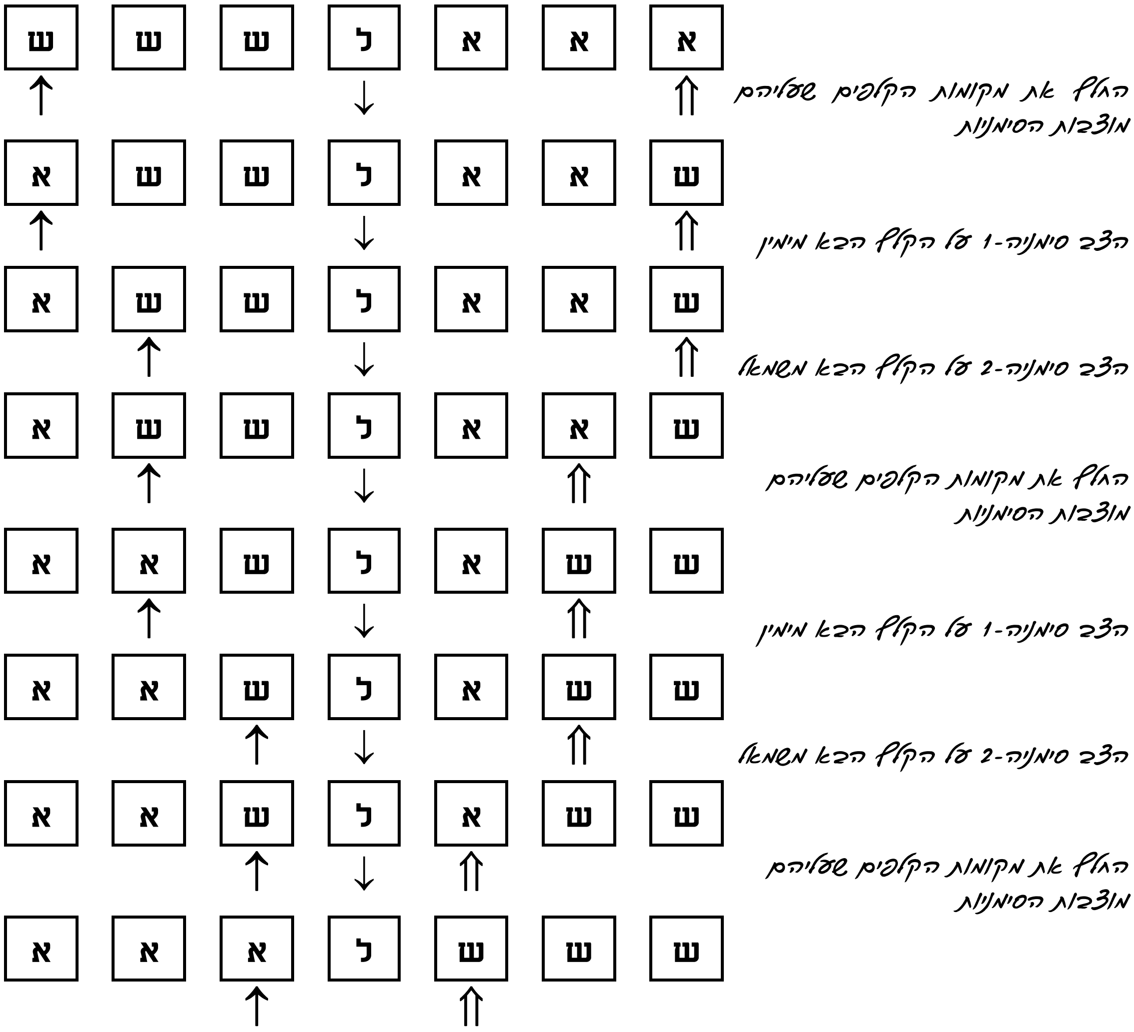
ונקודת הסיום היא:

א	א	א	ל	ש	ש	ש

את המקרה הפרטי הזה ניתן לפתור באופן הבא:

ראשית צריך להחליף את זוג הקלפים שבקצות השורה, ולקדם את הסימניות פנימה (כלומר, לקדם את הסימניה השמאלית מקום אחד ימינה ואת הימנית מקום אחד שמאלה). אחר כך צריך להחליף את זוג הקלפים שעליהם מצביעות הסימניות כעת, ושוב לקדם את הסימניות פנימה. לבסוף נותר להחליף את זוג הקלפים שעדיין לא הוחלפו.

הנה תיאור מפורט של אלגוריתם הפותר את המקרה הפרטי, יחד עם מעקב אחר ביצועו (סימניה-1 מסומנת בחץ דק וסימניה-2 מסומנת בחץ עבה):



פתרון המקרה הפרטי של הבעיה מלמד כי כדי להשיג את המטרה יש לבצע כמה פעמים את התת-משימה הבאה:

החלפת מקומות שהקלפים עליהם מוצבות הסימניות וקידום הסימניות פנימה

תת-משימה זו תבוצע על ידי קבוצת ההוראות הבאה:

1. החלף את מקומות הקלפים שעליהם מוצבות הסימניות
2. הצב סימניה-1 על הקלף הבא מימין
3. הצב סימניה-2 על הקלף הבא משמאל

האלגוריתם הדרוש צריך להורות על ביצוע-חוזר של התת-משימה (כלומר, שלוש ההוראות). מספר החזרות תלוי במספר הקלפים בשורה. מאחר שיש לכתוב אלגוריתם כללי, שיתאים לשורת קלפים בכל אורך אי-זוגי, לא יהיה זה מספיק לכתוב אלגוריתם המטפל באורך מסוים. בנוסף, אפילו אם היה ידוע מראש אורך מסוים, למשל 401, אין זה סביר שנכתוב אלגוריתם ובו כתובות שלוש ההוראות שוב ושוב 401 פעמים.

? כיצד ננסח אלגוריתם שיתאים לכל שורת קלפים באורך אי-זוגי?

כדי לנסח את האלגוריתם המבוקש, נשתמש ב**הוראה לביצוע-חוזר-בתנאי**, שתורה לחזור על ביצוע קבוצת שלוש ההוראות שתיארנו כל עוד מתקיים התנאי:

הקלף שעליו מוצבות הסימניות אינו לבן

האלגוריתם הבא פותר את הבעיה, והוא כולל הוראה לביצוע-חוזר-בתנאי:

אלגוריתם לפתרון בעיה 3

1. **כל עוד** הקלף שעליו מוצבות הסימניות אינו לבן **בצע:**
   1.1. החלף את מקומות הקלפים שעליהם מוצבות הסימניות
   1.2. הצב סימניה-1 על הקלף הבא מימין
   1.3. הצב סימניה-2 על הקלף הבא משמאל

סוף פתרון בעיה 3

ההוראה **כל עוד ... בצע:**

.

.

.

היא **הוראה לביצוע-חוזר-בתנאי**, המורה לחזור על ביצוע של קבוצת הוראות כל עוד מתקיים תנאי מסוים.

בדומה להוראה לביצוע-בתנאי גם הוראה לביצוע-חוזר-בתנאי היא הוראת בקרה.

שאלה 2.6

תארו את מצבי ביצוע האלגוריתם לפתרון בעיה 3 עבור כל אחת מן האפשרויות הבאות של נקודת המוצא.

א. שורת קלפים באורך 9.

ב. שורת קלפים באורך 3.

שאלה 2.7

שנו את האלגוריתם שבפתרון בעיה 3 כך שבתום הביצוע תושג מטרה אחרת: הקלפים שמשני צידי הקלף הלבן יהיו מסודרים לסירוגין לפי צבעים (כלומר, **ש**, **א**, **ש**, ..., **ל**, ... **א**, **ש**, **א**).

2.2 תבניות

בפתרון בעיה 3 בסעיף הקודם נכלל ביצוע-חוזר של הפעולה "החלף את מקומות הקלפים שעליהם מוצבות הסימניות". פעולת ההחלפה משולבת בבדיקה חוזרת ונשנית של מקום הסימניות – "האם הן מוצבות על קלף שאינו לבן". **פעולת ההחלפה**, ו**פעולת הבדיקה החוזרת של ערך** (במקרה זה, צבע) הן פעולות שימושיות בפתרונות של בעיות אלגוריתמיות נוספות רבות. למשל, הבעיה של מיון שורת מספרים היא בעיה אלגוריתמית, ובפתרונה ניתן להשתמש שוב ושוב ב**החלפה** בין מספרים סמוכים בשורה, עד אשר השורה תהיה ממוינת. גם הבעיה של חיפוש מספר מסוים בשורת מספרים היא בעיה אלגוריתמית. ניתן לפתור אותה במעבר שיטתי על פני המספרים שבשורה משמאל לימין, תוך **בדיקה חוזרת של הערך** של המספר הבא בשורה (עד מציאת המספר הרצוי או עד ההגעה לסוף השורה).

לעתים קרובות, פתרונות לבעיות אלגוריתמיות כוללים פעולות, אשר חוזרות שוב ושוב בפתרונות שונים. לכל פעולה כזו יש מבנה אלגוריתמי בסיסי המתאר אופן ביצוע של משימה מנקודת מוצא למטרה. כיוון שפעולה זו משמשת שוב ושוב בפתרונות שונים נוח לקרוא לה בשם, ולהתייחס אל המבנה האלגוריתמי שלה כאל **תבנית** אלגוריתמית לביצוע משימה. **תבנית** מורכבת מהמרכיבים הבאים:

- **שם התבנית**, המבטא בצורה מאוד תמציתית משימה לביצוע או את דרך ביצועה (למשל החלפת ערכים).
- **נקודת מוצא**, המציינת את המצב ההתחלתי הנתון של המשימה לביצוע.
- **מטרה**, המתארת את המצב הסופי, את הפלט הדרוש, או את הערך שיש להחזיר בתום הביצוע.
- **אלגוריתם**, המתאר מתכון לביצוע המשימה. האלגוריתם הוא לב התבנית.

מחקרים שונים בתחום הוראת מדעי המחשב מצביעים על כך שפותרים מנוסים של בעיות אלגוריתמיות שומרים בזיכרונם פתרונות קודמים על פי התבנית שבבסיסם. בבואם לפתור בעיה אלגוריתמית הם מסוגלים לזהות קשר בינה ובין בעיות אחרות שכבר פתרו, על פי התבניות שמשמשות בפתרון הבעיות. משום כך, השימוש בתבניות מסייע לתהליך הפיתוח של אלגוריתמים. לכן בספר הלימוד אנו נתייחס גם לתבניות, נציג אותן ונדון בהן.

ספר הלימוד משלב תבניות בצורה מודרגת. כבר בפרק הבא מוצגות תבניות ראשונות, ובפרקים שאחריו מוצגות עוד ועוד תבניות בהתאם לבעיות האלגוריתמיות שבהם. כל פרק שנוספות בו תבניות חדשות, הן מוצגות בסוף הפרק, מודגמות, מתורגלות, ומקושרות לבעיות אלגוריתמיות שהוצגו בפרק. לפעמים תבנית חדשה אף מוצגת כסעיף של פרק. לעתים מורחב בפרק חדש המבט על תבנית שכבר הוצגה בפרק קודם.

הרחבה בנושא התבניות ושאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

סיכום

בפרק זה הכרנו את המושגים **אלגוריתם** ו**בעיה אלגוריתמית**.

אלגוריתם הוא מתכון לביצוע משימה, המורכב מקבוצת הוראות חד-משמעיות ואפשריות לביצוע אשר סדר ביצוען מוגדר היטב.

בעיה אלגוריתמית היא בעיה שבה מתוארות נקודת מוצא ומטרה המגדירות משימה, ונדרש אלגוריתם הפותר את המשימה, כלומר מגיע מנקודת המוצא אל המטרה.

בבעיות אלגוריתמיות ייתכנו אפשרויות רבות, לפעמים אינסוף, לנקודת המוצא.

בפרק זה ובפרקים הבאים אנו משתמשים **בפסאודו-קוד** לכתיבת אלגוריתמים, כלומר, במילים ובמשפטים בשפה חופשית אך ברורה וחד-משמעית.

סדר ביצוע ההוראות של אלגוריתם הוא על פי סדר הופעתן, אם לא נאמר אחרת.

כאשר יש צורך להתנות ביצוע של הוראות בקיום תנאי מסוים, האלגוריתם יכלול **הוראה לביצוע-בתנאי** (*אם... אחרת...*).

כאשר יש צורך בביצוע-חוזר של הוראות, האלגוריתם יכלול **הוראה לביצוע-חוזר** (*כל עוד... בצע...*).

הוראה לביצוע-בתנאי והוראה לביצוע-חוזר הן **הוראות בקרה** המנחות את אופן הביצוע של הוראות האלגוריתם.

שאלות נוספות

1. בצעו את האלגוריתם הבא ותארו את מהלך ביצועו. מהי התוצאה המוכרזת בסוף הביצוע?

1. הכפל ב-2 את שנת הולדתך
2. הוסף 5 לתוצאה
3. הכפל את התוצאה ב-50
4. הוסף את גילך (כמספר שלם) לתוצאה
5. החסר 250 מן התוצאה
6. חלק ב-100 את התוצאה
7. הכרז על התוצאה

2. על השולחן מונחים שלושה קלפים בשורה. על כל קלף רשום מספר. נתון האלגוריתם הבא:

1. השווה את המספר שעל הקלף השמאלי למספר שעל הקלף האמצעי
2. אם המספר על הקלף השמאלי גדול מהמספר על הקלף האמצעי
2.1. החלף את מקומותיהם
3. השווה את המספר על הקלף האמצעי הנוכחי למספר שעל הקלף הימני
4. אם המספר על הקלף האמצעי הנוכחי גדול מהמספר על הקלף הימני
4.1. החלף את מקומותיהם

א. תארו את מהלך ביצוע האלגוריתם עבור נקודת המוצא הבאה:

24	2	15

ב. תארו את מהלך ביצוע האלגוריתם עבור נקודת המוצא הבאה:

13	24	2

ג. תארו את מהלך ביצוע האלגוריתם עבור נקודת המוצא הבאה:

13	2	15

ד. מהי הבעיה האלגוריתמית שהאלגוריתם פותר?

ה. שנו את האלגוריתם כך שישיג את המטרה הבאה: הקלף שרשום עליו המספר הקטן ביותר יהיה בקצה הימני של שורת הקלפים.

3. מפעל מסמן כל מוצר שלו בסימן המורכב מאות גדולה באלף-בית האנגלי ומספרה. כלומר סדרת הסימנים של המפעל היא הסדרה A0, A1, …, A9, B0, B1, …, Z0, …, Z9. A0 הוא הסימן הראשון בסדרה ו-Z9 הוא הסימן האחרון בסדרה.

א. מהם הסימנים שעוקבים לסימנים A1, B9 ו-Z0?

ב. פתחו אלגוריתם לקריאת סימן שאינו הסימן האחרון בסדרה ולכתיבת הסימן הבא אחריו. הפעולות המותרות הן: קריאת סימן (המורכב מאות ומספרה), כתיבת אות, כתיבת ספרה.

4. בכיתת תלמידים יש ילד אחד ששערו ג'ינג'י וילדים רבים שצבע שערם אינו ג'ינג'י. התלמידים מסודרים בטור. הילד הג'ינג'י אינו עומד בראש הטור.
פתחו אלגוריתם אשר מטרתו היא שהילד הג'ינג'י יעמוד בראש הטור.
הפעולות המותרות הן: עמידה מול התלמיד שבראש הטור, התקדמות לתלמיד הבא בטור, החלפת מקומות בין התלמיד שאתה עומד מולו ובין התלמיד שבראש הטור.

5. על השולחן מונחים קלפים שחורים ואדומים בשורה. בשורה שלושה קלפים לפחות. הקלפים השחורים נמצאים משמאל לקלפים האדומים, וידוע כי מספר הקלפים האדומים קטן ממספר הקלפים השחורים.
נתונות שתי סימניות: סימניה-1 המוצבת על הקלף שבקצה השמאלי, וסימניה-2 המוצבת על הקלף שבקצה הימני.
הפעולות המותרות הן: הצבת סימניה מימין או משמאל לקלף שעליו היא מוצבת, והחלפה של מקומות הקלפים שעליהם מוצבות הסימניות זה בזה.

א. ציינו שלוש אפשרויות שונות לשורת הקלפים.

ב. נתון האלגוריתם הבא אשר מטרתו היא שכל הקלפים האדומים יהיו משמאל לכל הקלפים השחורים. השלימו את האלגוריתם:

1. כל עוד ____________________ בצע:

1.1. החלף את מקומות הקלפים שעליהם מוצבות הסימניות

1.2. הצב סימניה-1 על הקלף הבא מימין

1.3. הצב סימניה-2 על הקלף הבא משמאל

פרק 3 – מודל חישוב בסיסי

בפרק 1 הכרנו את המכונה מחשב, ובפרק 2 ראינו אלגוריתמים ראשונים, אשר לא נועדו לביצוע במחשב. בפרק זה נכיר אלגוריתמים המיועדים לביצוע במחשב, ונראה כיצד נכתוב אותם כתוכניות בשפת התכנות #C.

באמצעות בעיות אלגוריתמיות שונות נציג את מרכיביהם הבסיסיים של אלגוריתמים ואת יישומם בשפת התכנות #C. בכל בעיה יתואר פלט דרוש עבור קלט נתון. הקלט הוא נקודת המוצא של הבעיה, והפלט הוא המטרה. כל אלגוריתם שיפותח יתואר בצורה מילולית, בדומה לתיאורים שהוצגו בפרק 2, ולאחר מכן יוצג יישומו באמצעות תוכנית בשפת התכנות #C.

בפרק זה נכיר את האמצעי לשמירת נתונים במחשב, את ההוראות לביצוע קלט ופלט וכיצד נבצע חישובים ונשמור את תוצאותיהם. לאחר מכן נעקוב אחר מהלך ביצוע של אלגוריתם.

3.1 צעדים ראשונים: הוראת פלט, הוראת קלט ומשתנים

המחשב מציג הודעות באמצעות אמצעי פלט. בפרק 1 הזכרנו את שני אמצעי הפלט הנפוצים: מדפסת וצג. בפתרון הבעיה הבאה נראה אלגוריתם להצגת מילים כפלט. האלגוריתם מיושם בתוכנית בשפת #C, התוכנית הראשונה בפרק.

בעיה 1

מטרת הבעיה ופתרונה: הצגת משפט כפלט

> פתחו אלגוריתם שהפלט שלו הוא המילים Hello World, וישמו את האלגוריתם בתוכנית מחשב בשפת #C.

האלגוריתם לפתרון הבעיה יהיה האלגוריתם הפשוט הבא, הכולל הוראת פלט אחת:

1. *הצג כפלט את המילים* Hello World

היישום בשפת #C של הוראת הפלט (הצגת הפלט על המסך) ייראה כך:

```
System.Console.WriteLine("Hello World");
```

נבחן ממה מורכבת הוראה זו:

- נכתוב בסוגריים את ההודעה שברצוננו להציג על מסך. הודעה זו תמוסגר ב "..."
- הפעולה להצגת שורה על המסך
- המחלקה האחראית על קלט ופלט
- מרחב השמות המכיל מחלקות האחראיות לפעולות מערכת כלליות

כיוון שבתוכנית מתבצעות פעולות רבות של מחלקות השייכות למרחב השמות System, (כגון פעולות קלט פלט של המחלקה Console), נהוג לקצר את כתיבת הפעולות באמצעות הכרזה בתחילת התוכנית שהיא משתמשת במחלקות הנמצאות במרחב שמות זה. ההכרזה נכתבת כך:

```
using System;
```

וכעת ניתן לכתוב את הוראת הפלט בצורה מקוצרת כך:

```
Console.WriteLine("Hello World");
```

כעת ניצור מהמשפט שכתבנו תוכנית מלאה בשפת C# :

```
using System;
public class MyFirstProgram
{
     public static void Main ()
     {
         Console.WriteLine("Hello World");
     }
}
```

> *הידעת?*
> המשפט Hello World הוא
> המשפט הראשון שנאמר
> מחללית שיצאה מתחום
> האטמוספירה של כדור הארץ

תוצאת ההרצה של התוכנית תהיה הצגת המילים Hello World על המסך.

סוף פתרון בעיה 1

ננסה להבין כמה מהחלקים שהוספנו לתוכנית ונוסיף הערות להבהרה:

```
/*
Hello World  התוכנית מדפיסה את המשפט
*/
using System;
public class MyFirstProgram
{
    public static void Main ()
    {
      Console.WriteLine("Hello World");
    }  // Main סוף
}  // MyFirstProgram סוף המחלקה
```

שורה בתוכנית	הסבר
`/* Hello World התוכנית מדפיסה את המשפט */`	בתחילת כל תוכנית נפרסם ב**הערה** את מטרת התוכנית. מה הקלט ומה הפלט
`using System;`	הכרזה על **שימוש** במרחב השמות System
`public class MyFirstProgram`	הכרזה על **מחלקה** בתוכנית
`{`	פתיחת **תחום** המחלקה
`public static void Main ()`	הכרזה על נקודת **תחילת הביצוע**
`{`	פתיחת **תחום** הפעולה הראשית
`Console.WriteLine("Hello World");`	**גוף** הפעולה הראשית
`}  // Main סוף`	סגירת **תחום** הפעולה הראשית + **הערה**
`}  // MyFirstProgram סוף המחלקה`	סגירת **תחום** המחלקה + **הערה**

מחלקה – class

כל תוכנית בשפה מורכבת ממחלקות שונות. לכל מחלקה תפקיד ואחריות משלה.

תוכנית זו היא תוכנית פשוטה ביותר, ולכן מכילה מחלקה אחת בלבד, שהיא התוכנית כולה.

כל מחלקה מוגדרת באמצעות המילה **`class`**. בתוכנית זו מוכרז כי המחלקה `MyFirstProgram` היא ציבורית (**`public`**), משמע, פתוחה לשימוש לכל המעוניין. בשלב הראשון נכתוב תוכניות המכילות מחלקה אחת בלבד ונצהיר שמחלקה זו היא ציבורית.

בשפת C# מקובל כי שם מחלקה מתחיל תמיד באות גדולה, ואם שם המחלקה מורכב מכמה מילים, הן נכתבות צמודות זו לזו, והאות הראשונה בכל מילה היא גדולה.

Main

לכל תוכנית יש נקודת התחלה אחת בלבד. שורת הכותרת `Main` מציינת נקודה זו.

את המשמעות המדויקת של שורה זו על כל מרכיביה תבינו בהמשך לימודיכם.

המחלקה אשר מכילה את נקודת תחילת התוכנית (מכילה את שורת ה-Main) היא המחלקה הראשית
בתוכנית. שם המחלקה הראשית הוא למעשה שם התוכנית.

בדוגמה זו, שם המחלקה הראשית הוא MyFirstProgram.

גוף הפעולה הראשית

בגוף הפעולה הראשית נכתוב את רצף ההוראות שהוא תרגום המקודד את האלגוריתם הפותר את הבעיה לשפת התכנות.

כל הוראה נכתבת בשורה נפרדת המסתיימת בסימן ;.

בגוף תוכנית זו נכללת הוראה יחידה המציגה על המסך את הכיתוב Hello World.

אנו מבצעים זאת על ידי קריאה לפעולה WriteLine אשר מציגה שורה על המסך. שימו לב כיצד פנינו לפעולה זו: Console.WriteLine. זהו שמה המקוצר של הפעולה, המעיד על כך שהפעולה WriteLine היא פעולת פלט, השייכת למחלקה האחראית לקלט ולפלט. השם המלא הוא System.Console.WriteLine והוא מעיד על כך שהמחלקה Console, שייכת למרחב השמות של System (הכוללת מחלקות שאחראיות לפעולות מערכת כלליות).

תחום

את רצף ההוראות, המהוות את גוף הפעולה הראשית, יש לתחום בין פותח מסולסל לסוגר מסולסל (הסימנים {...}).

בדומה לתחום הפעולה הראשית, יש לתחום בין פותח מסולסל לסוגר מסולסל את כל ההוראות השייכות למחלקה. אם כך, בתוכנית זו הוגדרו שני תחומים של הוראות: האחד למחלקה התחומה בסימנים { } החיצוניים, והשני לפעולה הראשית Main התחומה בסימנים { } הפנימיים.

הערה

פעמים רבות נרצה לכתוב הערות בתוכנית, אשר נועדות לקורא התוכנית (תיעוד). נציג כאן שתי דרכים לכתיבת הערות:

- הערה אשר מתפרשת על פני כמה שורות ניתן לרשום בין הסימנים /* ... תוכן ההערה ... */
למשל שלוש השורות הראשונות בדוגמה זו הן הערה המבהירה לקורא מהי מטרת התוכנית.
- הערה אשר מתפרשת על פני שורה בודדת ניתן לרשום אחרי הסימנים //

למשל ההערות המופיעות בשתי השורות האחרונות בדוגמה זו מסייעות לקורא לשייך את הסוגריים המסולסלים לתחומים השונים.

אין חובה לכלול הערות בתוכנית, אך הן תורמות תרומה משמעותית לקריאות התוכנית ולכן חשוב להוסיפן.

שאלה 3.1

פתחו אלגוריתם המציג על המסך את שמכם, וישמו אותו בשפת C#.

למשל, אם השם הוא יאיר, הפלט יהיה: Yair

שאלה 3.2

פתחו אלגוריתם המציג על המסך את שמכם מוקף במסגרת של כוכביות, וישמו אותו בשפת C#.

למשל אם השם הוא יאיר הפלט יהיה:

```
******
*Yair*
******
```

הדרכה: פעולת `WriteLine` מציגה שורה אחת על המסך. בתוכנית זו עליכם להציג 3 שורות.

כזכור, המחשב קולט נתונים באמצעות אמצעי קלט. אמצעי קלט נפוץ הוא לוח המקשים – המקלדת.

בפתרון הבעיה הבאה, נראה אלגוריתם הקולט נתונים ומציגם כפלט.

בעיה 2

מטרת הבעיה ופתרונה: שימוש במשתנים, הוראות קלט ופלט למשתנים, תוכנית הכוללת כמה משפטים ומעקב ראשון אחרי ביצוע של תוכנית.

> פתחו אלגוריתם שהקלט שלו הוא שני מספרים שלמים (המופרדים ברווח), והפלט שלו הוא ההודעה: "שני המספרים שנקלטו הם:" ומתחתיה שני המספרים. ישמו את האלגוריתם בתוכנית בשפת C#.

שימו ♥: מספר הקלטים האפשריים בבעיה זו הוא רב (ולמעשה אינסופי) כיוון שקלט יכול להיות **כל** זוג מספרים, למשל: 7 ו-5 , או 20 ו-2. האלגוריתם המבוקש צריך לתת את הפלט הנכון עבור כל זוג מספרים שהוא, כלומר, הוא צריך להיות אלגוריתם כללי.

חלוקה לתת-משימות

את המשימה שיש לפתור בבעיה ניתן לחלק לשלוש תת-משימות:

1. קליטת שני מספרים שלמים
2. הצגת ההודעה "שני המספרים שנקלטו הם:"
3. הצגת שני המספרים שנקלטו בשורה חדשה

? היכן ישמור המחשב את הנתונים הנקלטים?

הנתונים הנקלטים נשמרים בתאי הזיכרון המכונים "**תאי משתנים**" או בקיצור "**משתנים**".

> **משתנה** (variable) הוא תא זיכרון אשר במהלך ביצוע אלגוריתם ניתן לשמור בו ערך ולקרוא את הערך השמור בו. למידע השמור בתוך המשתנה קוראים **ערך המשתנה**. פנייה למשתנה נעשית באמצעות שמו, שהוא **שם המשתנה**.

בבעיה זו הקלט הוא שני נתונים (שני מספרים), ולכן נגדיר שני משתנים שנקרא להם: **num1** ו-**num2**. בכל אחד מהמשתנים יישמר **ערך אחד בלבד**. כאשר אנו בוחרים שמות למשתנים כדאי לבחור שמות משמעותיים, שיעידו על תפקידיהם של המשתנים. בחירת שמות משמעותיים מסייעת לקריאות התוכנית ולבהירותה, בדיוק כמו כתיבת הערות בתוכנית.

בשפת C# מקובל להתחיל שם של משתנה באות קטנה. אם שמו מורכב מכמה מילים הן נכתבות צמודות זו לזו ללא רווחים; החל מהמילה השנייה תיכתב כל אחת מהן באות גדולה בתחילתה.

האלגוריתם לפתרון הבעיה ישתמש בשני המשתנים שבחרנו:

1. קלוט שני מספרים שלמים במשתנים num1 ו-num2

2. הצג כפלט את ההודעה: "שני המספרים שנקלטו הם:"
3. הצג כפלט בשורה חדשה את ערך המשתנה num1 ואת ערך המשתנה num2

יישום האלגוריתם

ב-C# יש להצהיר על כל משתנה לפני השימוש בו. הצהרה נעשית בכתיבת **טיפוס** המשתנה ושמו של המשתנה.

> **טיפוס** (type) הוא סוג של ערכים. למשל, כל המספרים השלמים הם מטיפוס שלם וכל המספרים הממשיים הם מטיפוס ממשי.

כיוון שבחרנו במספרים שלמים, נגדיר כי שני המשתנים הם מטיפוס שלם. זאת נעשה בשימוש במילה **int** (קיצורה של המילה האנגלית integer, שמשמעותה מספר שלם), למשל כך:

```
int num1;
```

בהצהרת משתנים בתוכנית C# ניתן להצהיר על כמה משתנים מאותו טיפוס ברשימת שמות אחת שלפניה יופיע שם הטיפוס ואחריה הסימן נקודה פסיק. שמות המשתנים יופרדו בפסיקים. נצהיר על משתנים באותה השורה רק כאשר יש להם תפקיד דומה וניתן להסביר את תפקידם בהערת תיעוד אחת משותפת.

למשל נוכל להצהיר על שני המשתנים כך:

```
int num1, num2;    // שני משתנים לשמירת מספרים שלמים הנקלטים מהמשתמש
```

הצהרה על משתנה יכולה להיות בכל מקום בתוך תחום הפעולה, לפני ההתייחסות הראשונה אליו. עם זאת, כדי שהתוכנית תהיה בהירה וקריאה מקובל לרכז את כל הצהרות המשתנים ביחד בתחילת התחום.

הוראה 1 (קלט)

כדי ליישם את הוראה 1 עלינו ללמוד כיצד לקלוט.

קליטת מספר שלם נעשית באמצעות ההוראה:

```
שם משתנה = int.Parse(Console.ReadLine());
```

הוראה זו קולטת ערך שלם ומכניסה אותו לתוך המשתנה ששמו כתוב בצד שמאל. למעשה, זו הוראה מורכבת למדי, המפעילה כמה פעולות. הפעולה הראשונה שמופעלת היא `ReadLine` של המחלקה `Console`, הקוראת שורה מהמקלדת. השורה הזאת מועברת למחלקה האחראית למספרים השלמים **int**. מחלקה זו בתורה מפעילה על השורה שקיבלה את הפעולה `Parse` והיא מפרשת את רצף הסימנים שנקרא מהמקלדת ומתרגמת אותו לערך שלם.

בעת הרצת התוכנית, כאשר תתבצע פעולה זו, **תעצור** התוכנית ותחכה לקלט מתאים מהמשתמש. נהוג להוסיף הנחיה למשתמש, מעין הדרכה מדוע נעצרה התוכנית ולמה היא מצפה. את ההנחיה אפשר להציג על המסך באמצעות הוראת פלט שנכתבת לפני הוראת הקלט. בהוראת פלט כזו נעדיף שלא לעבור לשורה הבאה בסיום הדפסת ההודעה על המסך. לשם כך לא נשתמש בפעולה `WriteLine` שהשתמשנו עד כה, אלא בפעולה `Write`. גם היא פעולת פלט, בדומה ל-`WriteLine`, אך היא אינה גורמת למעבר לשורה הבאה במסך.

לכן, את הוראה 1 ניישם במשפטי הקלט הבאים:

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
```

הוראה 2 (פלט ערכי המשתנים)

גם הפעם ברצוננו להדפיס הודעת פלט על המסך ונרצה לשלב בה את ערכי המשתנים. לשם כך נשתמש בהוראת הדפסה, ונסמן בגוף ההודעה להדפסה היכן ישתלבו ערכי המשתנים. הסימונים הם רשימה של מספרים בסדר עולה, החל מ-0, עטופים בסוגריים מסולסלים ({ }). לאחר ההודעה להדפסה (התחומה בגרשיים כפולים), נוסיף סימן פסיק (,) ואחריו תופיע רשימת המשתנים שאת ערכם ברצוננו לשלב במשפט הפלט, והם מופרדים בפסיקים. כך:

```
Console.WriteLine("The two numbers are: {0} {1} ",num1, num2);
```

שימו ♥: ניתן לשלב בהודעה ערכים של משתנים רבים, אך מספר המשתנים ברשימה חייב להיות **שווה** למספר הסימונים בהודעת ההדפסה.

אם ברצוננו להציג על המסך רק ערך של משתנה יחיד, ללא שילובו בהודעה כלשהי, ניתן לעשות זאת בצורה פשוטה יותר, באמצעות אחת מהוראות הפלט `Write` או `WriteLine`, למשל כך:

```
Console.WriteLine(num1);
```

נשלים את גוף התוכנית לכדי תוכנית מלאה, בדומה למה שהודגם בפתרון בעיה 1:

שימו ♥ כי בתוכנית שילבנו הערות. בתחילה שילבנו הערה המבהירה מה תפקיד התוכנית כולה, ואחר כך שילבנו הערות המבהירות לקורא מה תפקיד כל חלק בתוכנית.

שילוב ההערות אינו בגדר חובה, אך מומלץ ביותר כדי שהתוכנית תהיה ברורה לכל אדם הקורא אותה.

```
/*
 התוכנית קולטת שני מספרים שלמים
 ומציגה את ערכם כפלט
*/
using System;
public class ReadWrite
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int num1, num2; //משתנים לשמירת מספרים שלמים הנקלטים מהמשתמש
        // הוראות התוכנית
        Console.Write("Enter first number: ");
        num1 = int.Parse(Console.ReadLine());
        Console.Write("Enter second number: ");
        num2 = int.Parse(Console.ReadLine());
        Console.WriteLine("The two numbers are: {0} {1}",num1, num2);
    } // Main
} // class ReadWrite
```

נעקוב אחר הרצת התוכנית `ReadWrite` עבור קלט כלשהו.

כאשר ניתנת למחשב סדרה של נתונים הוא קולט אותם משמאל לימין. למשל, אם הקלט עבור התוכנית `ReadWrite` הוא המספרים: 20 10, אז המספר השמאלי 10 מוקלד ראשון, והמספר הימני 20 מוקלד שני.

הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית ייראה כך (לאחר נתון שהמשתמש מקליד, עליו להקיש על המקש <Enter>):

המחשב יציג כפלט:

```
Enter first number:
```

המשתמש יקליד את הערך 10 שיופיע אף הוא על המסך, בהמשך השורה:

```
Enter first number: 10
```

המחשב יציג כפלט:

```
Enter second number:
```

המשתמש יקליד את הערך 10 שיופיע אף הוא על המסך, בהמשך השורה:

```
Enter second number:  20
```

המחשב יציג כפלט:

```
The two numbers are: 10 20
```

סוף פתרון בעיה 2

כדי להבין היטב את תפקידם של משתנים בתוכנית ואת השפעת הפעולות הקשורות אליהם, נעקוב שוב אחרי ביצוע התוכנית שבפתרון בעיה 2, אך הפעם נתמקד בזיכרון.

המשפט הראשון שמתייחס למשתני התוכנית הוא משפט ההצהרה על המשתנים:

```
int num1, num2;
```

כתוצאה מביצוע המשפט הזה מוקצים עבור `num1` ועבור `num2` תאי זיכרון **שתוכנם אינו ידוע**. אפשר לצייר זאת כך:

num1:	?	num2:	?

המשפטים הבאים בתוכנית הם משפטי הקלט (בצירוף הנחיות):

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
```

תוצאת הביצוע של משפט קלט היא שמירת ערך במשתנה.

למשל, אם בתגובה להודעה `Enter first number` הקליד המשתמש את המספר 10, הרי אחרי ביצוע משפט הקלט הראשון ערכו של `num1` יהיה 10. אם בתגובה להודעה `Enter second number` הקליד המשתמש את המספר 20, אז אחרי ביצוע משפט הקלט השני ערכו של `num2` יהיה 20. נוכל לצייר זאת כך:

num1:	10	num2:	20

לאחר שמירת הנתונים בזיכרון יבצע המחשב את המשפט הבא , האחרון בתוכנית, ויציג כפלט את ההודעה, בצירוף הערכים השמורים בתוך המשתנים:

```
The two numbers are: 10 20
```

אין לפעולת הפלט כל השפעה על ערכם של המשתנים.

בתום ביצוע המשפט האחרון בתוכנית "ישוחררו" תאי הזיכרון שהוקצו בתחילת הרצת התוכנית, כלומר תאי זיכרון אלה כבר לא שייכים לתוכנית ואסור לה להשתמש בהם יותר. אם נבקש להריץ שוב את התוכנית, יוקצו שוב תאי זיכרון, ואלה **אינם** בהכרח אותם תאי הזיכרון שהוקצו בהרצה הקודמת.

שימו ♥: מאחר שהגדרנו את שני המשתנים מטיפוס **int**, הם יכולים להכיל מספרים שלמים בלבד.

שאלה 3.3

א. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית `ReadWrite` (שבפתרון בעיה 2), עבור הקלט 10 5. זכרו שהמספר 5 מוקלד ראשון.

ב. נתייחס להרצת התוכנית `ReadWrite` עבור הקלט 10 5. מה יהיו הערכים השמורים במשתנים `num1` ו-`num2` בתחילת הביצוע? מה יהיו הערכים השמורים בהם לאחר כל משפט קלט? ולאחר ביצוע משפט הפלט האחרון?

ג. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית `ReadWrite` עבור הקלט 5 10.

ד. ענו על אותן שאלות מסעיף ב, אך הפעם התייחסו לביצוע התוכנית `ReadWrite` עבור הקלט 5 10.

שאלה 3.4

בחרו שני משתנים, הראשון לשמירת מספר הבנות בכיתה והשני לשמירת מספר הבנים בכיתה. לכל משתנה בחרו שם מתאים והצהירו עליו ב-#C, תוך ציון טיפוסו, ותוך תיעוד תפקידו בהערה מתאימה.

שימו ♥: פיתוח ויישום אלגוריתם יעשה תמיד על פי השלבים הבאים, כפי שנעשה בפתרון בעיה 2:

1. בחינת דוגמאות קלט שונות והבנת הקשר הדרוש בין הקלט לפלט.
2. חלוקת המשימה לתת-משימות.
3. בחירת משתנים – תפקיד, שם וטיפוס לכל משתנה.
4. כתיבת האלגוריתם.
5. יישום האלגוריתם על ידי תוכנית.

אנו מקפידים על פיתוח ועל יישום של אלגוריתם בשלבים. אמנם בפרק זה האלגוריתמים לפיתוח הם קצרים, אך חשוב כבר עכשיו להבחין בשלבים השונים. ככל שנתקדם יותר בחומר הלימוד נפתח אלגוריתמים מורכבים יותר, וחשיבות הפיתוח בשלבים תתברר יותר ויותר.

לאחר כתיבת התוכנית חשוב לבצע מעקב לבדיקתה, כפי שביצענו בפתרון בעיה 2. מעקב זה מסייע בבדיקת נכונות התוכנית, ולעתים נמצא בעזרתו שגיאות שנצטרך לתקן.

שאלה 3.5

פתחו וישמו בשלבים אלגוריתם שיקבל כקלט שלושה מספרים שלמים, והפלט שלו יהיה המספר השני שנקלט. למשל, עבור הקלט: 3 5 9 הפלט הדרוש הוא 5.

שאלה 3.6

נתונה התוכנית הבאה:

```
using System;
public class InOut
{
    public static void Main ()
    {
        int num1;
        int num2;
```

```
        Console.Write("Enter first number: ");
        num1 = int.Parse(Console.ReadLine());
        Console.Write("Enter second number: ");
        num2 = int.Parse(Console.ReadLine());
        Console.WriteLine(num1);
        Console.WriteLine(num2);
        Console.Write("Enter another one: ");
        num2 = int.Parse(Console.ReadLine());
        Console.WriteLine(num1);
        Console.WriteLine(num2);
    }
}
```

א. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית InOut, כאשר הקלט עבור שני משפטי הקלט הראשונים הוא 3 2, והקלט עבור משפט הקלט השלישי הוא 5.

ב. תארו את הדו-שיח בין המשתמש למחשב בעת ביצוע התוכנית InOut, כאשר הקלט עבור שני משפטי הקלט הראשונים הוא 5 2, והקלט עבור משפט הקלט השלישי הוא 3.

שאלה 3.7

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא שלושה מספרים שלמים, והפלט שלו הוא שתי שורות, ובכל שורה שניים ממספרי הקלט: בשורה הראשונה המספר השני והמספר השלישי מסודרים לפי סדר קליטתם, ובשורה השנייה המספר הראשון והמספר השני מסודרים בסדר **הפוך** לסדר קליטתם.

שאלה 3.8

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא שלושה מספרים שלמים המהווים סדרה. הפלט שלו הוא סדרה של שישה מספרים שבה משוכפל כל אחד מנתוני הקלט כמספר הפעמים המתאים למקומו הסידורי בסדרת הקלט. למשל, עבור הקלט: 6 3 8, הפלט הוא:

8 3 3 6 6 6

הנתון הראשון מופיע פעם אחת, השני פעמיים והשלישי שלוש פעמים.

3.2 הוראת הֲשָׂמָה

בסעיף הקודם הכרנו אלגוריתמים (ותוכניות) למחשב שקולטים נתונים ונותנים פלט. אך קלט ופלט הם רק מרכיב אחד של אלגוריתמים למחשב. אלגוריתמים למחשב מיועדים בדרך כלל לביצוע חישובים ועיבודים שונים, בנוסף לקלט ולפלט. בסעיף זה נראה אלגוריתמים ראשונים המבצעים חישובים.

בעיה 3

מטרת הבעיה ופתרונה: הצגת הוראת השמה.

> פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא שני מספרים שלמים חיוביים, המציינים אורך ורוחב של מלבן, והפלט שלו הוא שטחו והיקפו של המלבן.

בבעיה זו, כמו בבעיות הקודמות שפתרנו, מספר הקלטים הוא רב. אם האורך והרוחב של המלבן יכולים להיות כל זוג מספרים שלמים חיוביים, הרי יש בעצם **אינסוף** קלטים אפשריים.

בדיקת דוגמאות

כזכור, לפני שאנו ניגשים לכתיבת האלגוריתם כדאי לוודא שאנו מבינים את המשימה על ידי כך שנבחן את הפלט עבור דוגמאות קלט מגוונות:

שאלה 3.9

ציינו את הפלט עבור כל אחד מהקלטים הבאים:

א. 10 5

ב. 3 12

חלוקה לתת-משימות

מהן התת-משימות של האלגוריתם?

תחילה יש לקלוט את הנתונים, אחר כך יש לחשב את השטח ואת ההיקף, ולבסוף יש להציג כפלט את תוצאת החישוב. נתאר זאת בחלוקה הבאה לתת-משימות:

1. קליטת שני מספרים שלמים המייצגים אורך ורוחב של מלבן
2. חישוב שטח המלבן
3. חישוב היקף המלבן
4. הצגת תוצאת החישוב

בחירת משתנים

כדי לשמור את אורכו ואת רוחבו של המלבן נשתמש בשני משתנים מטיפוס שלם, שנקרא להם length ו-width בהתאמה (זכרו! אנו מקפידים על בחירת שמות משמעותיים). בנוסף, נבחר את המשתנים area ו-perimeter מטיפוס שלם, לשמירת תוצאות חישובי השטח וההיקף.

בסך הכול בחרנו ארבעה משתנים מטיפוס שלם:

length – ישמור את אורך המלבן.

width – ישמור את רוחב המלבן.

area – ישמור את שטח המלבן.

perimeter – ישמור את היקף המלבן.

האלגוריתם

את תת-משימה 1 נבצע על ידי הוראת קלט מתאימה:

קלוט אורך ורוחב של מלבן ב-length ו-width

? לאחר קליטת הנתונים עלינו לחשב שטח והיקף. כיצד נבצע את החישובים?

חישוב שטח מתקבל באמצעות הביטוי: length * width

חישוב היקף מתקבל באמצעות הביטוי: 2 * (width + length)

שימו ♥: ב-C# מציינים פעולת כפל באמצעות התו *

? היכן נשמור את תוצאות החישובים?

המחשב מסוגל לבצע חישוב של ביטוי חשבוני ולשים (לשמור) את תוצאת החישוב במשתנה. הביטוי החשבוני יכול לכלול ערכים המצוינים במפורש (למשל המספר 2) או ערכים השמורים במשתנים. הדרך להורות למחשב לשמור את תוצאת הביטוי במשתנה היא באמצעות **הוראת השמה**.

אם כך, נכלול באלגוריתם הוראות השמה לשמירת תוצאות החישוב במשתנים area ו-perimeter.

האלגוריתם לפתרון הבעיה יכלול שתי הוראות השמה ויהיה:

1. *קלוט אורך ורוחב של מלבן ב-length וב-width*
2. *חשב את שטח המלבן על ידי* length * width *והשם (שמור) את התוצאה ב-*area
3. *חשב את היקף המלבן על ידי* 2 * (width + length) *והשם (שמור) את התוצאה ב-*perimeter
4. *הצג כפלט את ערך* area *ואת ערך* perimeter

יישום האלגוריתם

כעת ניגש ליישום האלגוריתם בתוכנית #C.

יישום הוראות ההשמה יהיה על ידי שני המשפטים הבאים:

```
area = length * width;
perimeter = (width + length) * 2;
```

אלה הם משפטי השמה. כל משפט מורה על ביצוע חישוב ועל השמת תוצאתו במשתנה.

ביצוע המשפט הראשון, יביא לחישוב מכפלת ערכו של `width` בערכו של `length`, ולהשמת התוצאה במשתנה `area`.

ביצוע המשפט השני, יביא לחישוב הסכום של ערכו של `width` בערכו של `length` ולהכפלתו ב-2, והשמת התוצאה במשתנה `perimeter`.

הנה יישום האלגוריתם כולו בשפת #C:

```
Console.Write("Enter length: ");
length = int.Parse(Console.ReadLine());
Console.Write("Enter width:");
width = int.Parse(Console.ReadLine());
area = length * width;
perimeter = (width + length) * 2;
Console.WriteLine("The area is: {0}", area);
Console.WriteLine("The perimeter is: {0}", perimeter);
```

נשלים את משפטי התוכנית לתוכנית מלאה:

```
/*
התוכנית מחשבת את שטחו ואת היקפו של מלבן */
using System;
public class Rectangle
{
    public static void Main()
    {
        int length, width, area, perimeter;
        Console.Write("Enter length: ");
        length = int.Parse(Console.ReadLine());
        Console.Write("Enter width:");
        width = int.Parse(Console.ReadLine());
        area = length * width;
        perimeter = (width + length) * 2;
        Console.WriteLine("The area is: {0}", area);
        Console.WriteLine("The perimeter is: {0}", perimeter);
    } // Main
} // class Rectangle
```

הוראות קלט

הוראות השמה

הוראות פלט

מעקב

נעקוב עתה אחר מהלך הרצת התוכנית `Rectangle` עבור הקלט 3 5:

בתחילת ההרצה ערכי כל המשתנים אינם ידועים.

בעקבות ביצוע משפט הקלט הראשון תוצג ההודעה "` Enter length:`" ואז ימתין המחשב לקלט מן המשתמש. המשתמש יקיש 5 ו-<Enter>. תוצאת הוראת הקלט הראשונה תהיה שמירת הערך 5 במשתנה `length`.

לאחר מכן תוצג ההודעה "` Enter width:`" ושוב ימתין המחשב לקלט. המשתמש יקיש 3 ו-<Enter>. תוצאת הוראת הקלט השנייה תהיה שמירת הערך 3 במשתנה `width`.

בביצוע משפט ההשמה הראשון יחושב הערך 15, שהוא ערך הביטוי 5*3, המופיע בצד ימין של המשפט. ערך זה יישמר במשתנה `area`, ששמו כתוב בצד שמאל של המשפט.

בביצוע משפט ההשמה השני יחושב הערך 16, שהוא ערך הביטוי 2*(3+5), המופיע בצד ימין של המשפט. ערך זה יישמר במשתנה `perimeter`, ששמו כתוב בצד שמאל של המשפט.

שימו ♥: ערכיהם של `length` ושל `width` לא ישתנו בעקבות ביצוע משפטי ההשמה! משפט השמה משפיע רק על המשתנה שבצד שמאל של המשפט. משתנים המעורבים בביטוי שבצד ימין של המשפט אינם מושפעים ממנו.

בעקבות ביצוע שני משפטי הפלט האחרונים יתקבל הפלט:

```
The area is: 15
The perimeter is: 16
```

סוף פתרון בעיה 3

בפתרון בעיה 3 הכרנו את פעולת ההשמה:

> בפעולת **השמה**, המחשב מבצע חישוב כלשהו **ושָׂם** את התוצאה בתוך משתנה.
>
> פעולת ההשמה מיושמת ב-C# כך: `y = expression;`. משפט זה מורה על השמת ערך הביטוי `expression` בתוך המשתנה `y`.
>
> הביטוי `expression` יכול להיות ביטוי פשוט (למשל מספר או שם של משתנה) או ביטוי המורכב מפעולות חשבוניות שונות.

דוגמאות:

- `a = 7;` פירושו: "השם את הערך 7 במשתנה ששמו `a`". לאחר ביצוע פעולה זו, ערכו של `a` יהיה 7.
- `a = b;` פירושו: "השם את ערכו של המשתנה `b` בתוך המשתנה `a`". למשל, אם ערכו של `b` לפני ביצוע המשפט הוא 3, אז לאחר ביצוע המשפט ערכו של `a` יהיה 3. ערכו של `b` לא ישתנה אלא יישאר 3.
- `a = b * c;` פירושו: "הכפל את ערכיהם של `b` ושל `c` והשם את התוצאה ב-`a`". למשל, אם לפני ביצוע המשפט ערכיהם של `b` ושל `c` הם 9 ו-5 בהתאמה, אז לאחר ביצוע המשפט יהיה ערכו של `a` שווה ל-45 (וערכיהם של `b` ושל `c` לא ישתנו).

שימו ♥: כיוון שמשתנה יכול להכיל ערך אחד בלבד בכל רגע, אז בעת ביצוע המשפט `y = expression;`, ערכו הקודם של `y` הולך לאיבוד, כלומר ערכו של הביטוי `expression` "דורך" על מה שהיה בתוך `y` לפני ביצוע המשפט. אבל כאמור משפט ההשמה אינו משפיע על אף משתנה חוץ מ-`y`.

שימו ♥: בביטוי חשבוני מתקיים סדר הקדימויות המוכר של פעולות החשבון. כלומר, לסוגריים עדיפות גבוהה ביותר, עדיפות נמוכה יותר לכפל ולחילוק, ועדיפות נמוכה ביותר לחיבור ולחיסור.

שאלה 3.10

נניח שהקלט בעת ביצוע התוכנית `Rectangle` הוא: 10 7.

א. מה יהיו ערכי המשתנים `length` ו-`width` לאחר ביצוע משפטי הקלט?

ב. מה יהיו ערכי כל המשתנים לאחר ביצוע משפט ההשמה השני?

ג. תארו את הדו-שיח בין המחשב למשתמש במהלך הרצת התוכנית.

שאלה 3.11

כתבו משפטי השמה לביצוע הפעולות הבאות:

א. איפוס המשתנה `a` (איפוס משמעותו השמת הערך 0).

ב. השמת תוצאת החישוב 3*(729-511) במשתנה `a`.

ג. השמת כפליים מערכו של המשתנה `b` במשתנה `a`.

ד. השמת סכום ערכי המשתנים `x` ו-`y` במשתנה `w`.

שאלה 3.12

נתון קטע תוכנית ובו המשפטים הבאים:

```
Console.Write("Enter first number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
b = int.Parse(Console.ReadLine());
c = a + b;
Console.WriteLine(c);
```

תנו שתי דוגמאות קלט שונות שהפלט עבורן הוא 5.

שאלה 3.13

כתבו סדרה של ארבעה משפטי השמה המבצעים, לפי הסדר הבא:

1. השמת הערך 3 במשתנה `a`.
2. השמת תוצאת החישוב של הביטוי 3*9 במשתנה `b`.
3. השמת סכום ערכי `a` ו-`b` במשתנה `c`.
4. השמת מכפלת ערכי `a` ו-`c` במשתנה `d`.

מהם ערכי ארבעת המשתנים `a`, `b`, `c` ו-`d` בתום ביצוע סדרת המשפטים?

שאלה 3.14

נתונים משפטי התוכנית הבאים:

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
Console.Write("Enter third number: ");
num3 = int.Parse(Console.ReadLine());
```

```
diff1 = num1 * num2 - num3;
diff1 = num2 * num3 - num1;
Console.WriteLine(diff1);
Console.WriteLine(diff2);
```

נניח שהקלט במהלך ההרצה הוא: 3 2 3.

א. מה יהיו ערכי המשתנים num1, num2 ו-num3 לאחר ביצוע משפטי הקלט?

ב. מה יהיו ערכי המשתנים diff1 ו-diff2 לאחר ביצוע משפטי ההשמה?

ג. תארו את הדו-שיח בין המחשב למשתמש במהלך משפטי התוכנית.

שאלה 3.15

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא מספר חיובי שלם, המציין אורך צלע של קובייה, והפלט שלו הוא נפח הקובייה ושטח הפנים שלה.

הדרכה: אם צלע הקובייה היא a, הנפח יחושב כ-a^3 ושטח הפנים יחושב כ-$6a^2$.

שימו ♥ : גם פעולת קלט כוללת למעשה השמה. למשל במשפט:

```
length = int.Parse(Console.ReadLine());
```

מתבצעת פעולת השמה של המספר הנקלט לתוך המשתנה length.

כזכור, כאשר אנו מצהירים על משתנה, מוקצה לו מקום בזיכרון, אך ערכו אינו ידוע. במקרים מסוימים נרצה לתת למשתנה ערך מיד בעת ההצהרה עליו. זאת כדאי לעשות כאשר ידוע לנו כבר בשלב ההצהרה מה צריך להיות ערכו ההתחלתי של משתנה, ואין הוא תלוי בהוראה שצריכה להתבצע מאוחר יותר, כמו הוראת קלט. מתן ערך התחלתי למשתנה נקרא אתחול.

אתחול של משתנה משמעותו מתן ערך התחלתי למשתנה.

הסיבה שלא כדאי לנו לדחות אתחול של משתנה, וכדאי לאתחל משתנה מיד כאשר ידוע לנו הערך המתאים לאתחול, היא שאם לא נעשה זאת, אנו עלולים לשכוח לעשות זאת מאוחר יותר. ערכו של המשתנה יישאר לא ידוע, וכאשר ננסה להשתמש בערכו – למשל להדפיסו או לשלבו בביטוי חשבוני, אין לדעת מה יהיה ערכו.

שפת C# מאפשרת לנו לשלב הצהרה והשמה באופן הזה:

```
int x = 3;
```

תוצאת ביצוע ההוראה היא הקצאת מקום בזיכרון עבור המשתנה x והשמת הערך 3 בתוכו. ההוראה הזאת שקולה לרצף ההוראות

```
int x;
x = 3;
```

גם הוראה כזאת ניתן לכתוב בשפת C#:

```
int x = num1 * num2;
```

וזאת בתנאי ש-num1 ו-num2 הם משתנים מטיפוס שלם, שהוצהרו קודם לכן וערכם ידוע.

אם כך, בעת ההצהרה יכול להופיע ביטוי גם בצד ימין של ההשמה, בתנאי שכל הערכים הכלולים בו כבר ידועים.

בהמשך הפרק נשתמש לעתים במונחים ערך התחלתי ומצב התחלתי:

הערך ההתחלתי של משתנה ביחס לתוכנית מסוימת או לקטע תוכנית מסוים, הוא הערך שיש במשתנה מיד לפני ביצוע ההוראה הראשונה בתוכנית או בקטע התוכנית.

מצב התחלתי של תוכנית או של קטע תוכנית כולל את ערכם ההתחלתי של כל המשתנים ביחס לאותה תוכנית או לקטע התוכנית.

למשל, אם השורה הראשונה בתוכנית היא `int x = 3;`, אז ערכו ההתחלתי של x ביחס לתוכנית הוא כמובן 3. אם השורה הראשונה היא `int x;`, אז ערכו ההתחלתי של x אינו ידוע. כאשר אנו בוחנים את ערכו של משתנה ביחס לקטע תוכנית, עלינו לבדוק מהו הערך האחרון שהושם לתוכו, **לפני** קטע התוכנית. זה יהיה ערכו ההתחלתי של המשתנה ביחס לקטע התוכנית. אם לא התבצעה שום השמה למשתנה לפני קטע התוכנית, הרי ערכו ההתחלתי ביחס לקטע התוכנית אינו ידוע.

3.3 טבלת מעקב

בסעיף הקודם ראינו הוראות השמה ראשונות פשוטות ובחנו לראשונה מהלך של השמת ערכים במשתנים בעת התקדמות הביצוע של אלגוריתם מהוראה להוראה.

בסעיף זה נראה הוראות השמה מורכבות יותר, ונציג דרך למעקב שיטתי אחר מהלך ביצוע של אלגוריתם.

בעיה 4

מטרת הבעיה ופתרונה: הצגת הוראת השמה אשר בה הערך החדש המושם במשתנה תלוי בערך הנוכחי של המשתנה, והצגת מעקב אחר מהלך ביצוע באמצעות טבלת מעקב.

נתבונן בשתי סדרות המספרים הבאות: 12 4 2 ו-30 10 5.

בשתי הסדרות האיבר הראשון (משמאל) הוא הקטן ביותר, האיבר השני גדול פי שניים מהאיבר הראשון, והאיבר השלישי גדול פי שלושה מהאיבר השני.

פתחו וישמו בשלבים אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי המציין איבר ראשון בסדרה (כדוגמת הסדרות המתוארות) והפלט שלו הוא האיבר השני והשלישי בסדרה בשורות נפרדות.

השתמשו באלגוריתם במשתנה אחד בלבד!

בדיקת דוגמאות

שאלה 3.16
מהו הפלט עבור הקלט 10, ומהו הפלט עבור הקלט 2?

חלוקה לתת-משימות

נחלק לתת-משימות באופן הבא:

1. קליטת מספר המייצג איבר ראשון בסדרה.
2. חישוב האיבר השני בסדרה והצגתו כפלט.
3. חישוב האיבר השלישי בסדרה והצגתו כפלט.

בחירת משתנים

כיוון שהוטלה המגבלה של שימוש במשתנה אחד, ברור שהמשתנה שנבחר ישמור בכל פעם אחד מאיברי הסדרה. נשתמש במשתנה מסוג שלם ונקרא לו **element**.

האלגוריתם

את התת-משימה הראשונה נבצע כמובן באמצעות קליטת האיבר הראשון בתוך element.

? כיצד נחשב באמצעות משתנה אחד בלבד את האיבר השני ואחר כך את האיבר השלישי בסדרה?

בביטוי שבהוראת השמה (כלומר, בצד ימין של ההוראה) אפשר לכלול גם את המשתנה אשר בו מושמת תוצאת החישוב של הביטוי (המשתנה המופיע בצד שמאל של ההשמה). במקרה כזה ערכו של המשתנה **לפני** ביצוע ההשמה משמש בחישוב ערכו החדש. לדוגמה:

*חשב את ערכו של הביטוי 2*num והשם את התוצאה ב-num*

משמעות הוראה זו היא שערכו של num לאחר ביצוע ההוראה יהיה פי 2 מערכו לפני ביצוע ההוראה. אם ערכו של num לפני ביצוע ההוראה הוא 5 אז ערכו לאחר הביצוע יהיה 10.

נשתמש בהוראת השמה כזו, שבה מופיע num בשני צדי משפט ההשמה. נחשב את האיבר השני על ידי הכפלה ב-2 של הערך השמור ב-element, נשים את התוצאה חזרה ב-element ונציג כפלט את ערכו. אחר כך נחשב את האיבר השלישי בסדרה באמצעות הכפלה ב-3 של הערך השמור ב-element, נשים את התוצאה חזרה ב-element ולבסוף נציג שוב את ערכו.

הנה האלגוריתם לפתרון הבעיה:

1. *קלוט מספר ב-element* // קליטת איבר ראשון בסדרה
2. *חשב: 2*element והשם את התוצאה ב-element* // חישוב האיבר השני בסדרה
3. *הצג כפלט את ערכו של element*
4. *חשב: 3*element והשם את התוצאה ב-element* // חישוב האיבר השלישי בסדרה
5. *הצג כפלט את ערכו של element*

יישום האלגוריתם

הנה התוכנית ליישום האלגוריתם שלעיל, ובה יש הערות כדי להבהיר את מטרתו של כל משפט. את מספרי השורות הוספנו לשם נוחות, ונשתמש בהם עוד מעט. הם אינם חלק מהוראות התוכנית!

```
/*
התוכנית נותנת כפלט את איבריה של סדרה בת שלושה איברים
*/
using System;
public class Sequence
{
    public static void Main ()
    {
        int element;
     1. Console.Write("Enter first element: ");
     2. element = int.Parse(Console.ReadLine());
     3. element = 2 * element;          //    חישוב האיבר השני
     4. Console.WriteLine("The second element is: {0}", element);
     5. element = 3 * element;          //    חישוב האיבר השלישי
     6. Console.WriteLine("The third element is: {0}", element);
```

```
    } // Main
}// class Sequence
```

מעקב

נעקוב אחר מהלך ביצוע התוכנית עבור דוגמת הקלט 5 :

אחרי ביצוע משפט הקלט יהיה ערכו של `element` שווה ל-5.

אחרי ביצוע משפט ההשמה בשורה 3 יהיה ערכו של `element` שווה ל-10.

אחרי ביצוע משפט ההשמה בשורה 5 יהיה ערכו של `element` שווה ל-30.

הדו-שיח בין המחשב למשתמש יהיה :

מחשב :	`Enter first element`
משתמש :	`5`
מחשב :	`The second element is: 10`
מחשב :	`The third element is:  30`

סוף פתרון בעיה 4

בפתרון בעיה 4 ראינו נקודה חשובה לגבי הוראות השמה :

משתנה יכול להופיע משני צדי הוראת השמה, כלומר, גם בתפקיד המשתנה שבו מתבצעת ההשמה, וגם כמשתנה שערכו משמש בביטוי שבצד ימין של ההשמה. בהוראה כזו ערכו החדש של המשתנה תלוי בערכו לפני ביצוע ההוראה.

למשל, ההוראה `counter = counter + 1;` תוסיף 1 לערכו של המשתנה `counter`. אם למשל לפני פעולת ההשמה היה ערכו של `counter` 5, אז אחרי ביצועה יהיה ערכו 6.

שאלה 3.17

הניחו שערכי המשתנים `a` ו-`b` לפני **כל אחד** ממשפטי ההשמה הבאים הם 3 ו-5, בהתאמה. מהו ערכו של `a` לאחר ביצוע כל משפט?

א. `a = 1;`

ב. `a = a + 1;`

ג. `a = 2 * a + 3;`

ד. `a = 2 * a + (a – 3);`

ה. `a = b;`

ו. `a = a * b;`

ז. `a = a + a * b;`

שימו ♥ : ערכו של `b` לא משתנה בעקבות אף אחת מההוראות האלו!

שאלה 3.18

כתבו משפטי השמה לביצוע ההוראות הבאות. את תוצאת החישוב יש לשמור במשתנה a :

א. הכפלת ערכו של המשתנה `a` ב-2.

ב. החסרת ערך המשתנה `b` מן המשתנה `a`.

ג. הכפלת ערכו של המשתנה `a` בסכום ערכי המשתנים `b` ו- `c`.

שאלה 3.19

נתון קטע התוכנית הבא:

```
Console.Write("Enter number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
b = int.Parse(Console.ReadLine());
a = a + b;
Console.WriteLine(a);
a = a - b;
Console.WriteLine(a);
```

פלט התוכנית הוא שני מספרים. הביאו שלוש דוגמאות קלט שונות, אשר עבור כל אחת מהן יהיה ההפרש בין שני מספרי הפלט שווה ל-9.

שאלה 3.20

האם המשפט `int x = 3 * x;` הוא משפט חוקי? נמקו את תשובתכם.

במהלך ביצוע התוכנית Sequence השתנה שוב ושוב ערכו של element. אמנם הצלחנו לבצע מעקב אחר ערכי המשתנה וגם מעקב אחר הודעות הפלט, אבל שיטת המעקב שהשתמשנו בה עלולה להיות מסורבלת עבור תוכניות ארוכות יותר ומורכבות יותר.

נוכל לעקוב אחר התוכנית באופן שיטתי באמצעות **טבלת מעקב**. בטבלת מעקב נעקוב אחר השינויים בערכי המשתנים ואחר הפלט הקורים במהלך ביצוע משפטי התוכנית.

נדגים את השימוש בטבלת מעקב אחר מהלך ביצוע התוכנית Sequence עבור הקלט 3:

מספר שורה	המשפט לביצוע	element	פלט
1	Console.Write("Enter first element: ");	?	Enter first element
2	element = int.Parse(Console.ReadLine());	3	
3	element = 2 * element;	6	
4	Console.WriteLine("The second element is: {0}", element);	6	The second element is: 6
5	element = 3 * element;	18	
6	Console.WriteLine("The third element is: {0}", element);	18	The third element is: 18

טבלת מעקב משמשת למעקב אחרי ביצוע של תוכנית שלמה או של קטע תוכנית, או של אלגוריתם. בטבלה יש שורה עבור כל הוראה לביצוע, עמודה עבור כל משתנה ועמודה עבור הפלט.

מבנה אפשרי לטבלת מעקב:

- עמודות הטבלה:
  - העמודה השמאלית ביותר – מספרי השורות של הוראות התוכנית (לפי מספריהן בתוכנית).
  - העמודה השנייה משמאל – "המשפט לביצוע", כלומר, הוראות התוכנית עצמן.
  - העמודות שבמרכז הטבלה (מהשלישית משמאל ועד השנייה מימין) – עמודה עבור כל משתנה של התוכנית.
  - העמודה הימנית ביותר – "פלט".

♦ שורות הטבלה:

- בעמודת "המשפט לביצוע" יהיו הוראות התוכנית זו אחר זו. נכתוב בטבלה רק הוראות שמשפיעות על ערכם של משתנים. למשל לא נכלול הצהרה על משתנה אם אינה כוללת אתחול.
- בעמודות המשתנים: אם בעקבות ההוראה המתאימה בשורה קיבל המשתנה ערך חדש, נכתוב את ערכו החדש, אחרת נכתוב את ערכו הנוכחי. משתנה שערכו אינו ידוע יסומן בסימן '?'.
- בעמודת הפלט: אם ההוראה המתאימה בשורה היא הוראת פלט, כגון `Console.WriteLine` או `Console.Write`, נכתוב את הפלט המתאים, אחרת המשבצת המתאימה בטבלה תישאר ריקה.

ניתן להגמיש מעט את מבנה הטבלה. למשל ניתן לאחד את שתי העמודות השמאליות ולכתוב את מספר השורה יחד עם המשפט שנמצא בשורה זו. אבל חשוב שתהיה בטבלה שורה לכל הוראה של התוכנית, וחשוב שתהיה עמודה לכל משתנה ועמודה עבור הפלט.

שאלה 3.21

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `Sequence` עבור הקלט 5, וטבלת מעקב עבור הקלט 1.

שאלה 3.22

נתון קטע התוכנית הבא:

```
1. c = 0;
2. a = (a + 5) * a;
3. b = b + 2 * a;
4. Console.WriteLine(a);
5. Console.WriteLine(b);
6. Console.WriteLine(c);
```

הניחו שהערכים ההתחלתיים של a, b ו- c (כלומר ערכיהם לפני תחילת ביצוע קטע התוכנית) הם 1, 2 ו-3 בהתאמה.

מלאו את טבלת המעקב עבור קטע התוכנית הנתון:

מספר השורה	המשפט לביצוע	a	b	c	פלט
		1	2	3	
1					
2					
3					
4					
5					
6					

שאלה 3.23

בנו טבלת מעקב אחר מהלך הביצוע של קטע התוכנית הבא עבור הקלט 3 2:

```
1. Console.Write("Enter first number: ");
2. num1 = int.Parse(Console.ReadLine());
3. Console.Write("Enter second number: ");
4. num2 = int.Parse(Console.ReadLine());
5. sum = num1 + num2;
```

```
sum = sum + sum;
sum = sum + sum;
Console.WriteLine(sum);
```

מהי מטרת קטע התוכנית? (כלומר, מה הוא מבצע עבור שני מספרים שלמים כלשהם?)

שאלה 3.24

כתבו קטע תוכנית ובו משפטי השמה אשר מכפילים את ערכו של המשתנה a ב-4, ולחיסור פעמיים של ערך המשתנה c מערכו של המשתנה b. בכל אחד מן הביטויים של משפטי ההשמה, השתמשו אך ורק בפעולת חיבור אחת או בפעולת חיסור אחת. בסוף קטע התוכנית יוצגו ערכי שלושת המשתנים.

כעת, בחרו ערכים התחלתיים כלשהם למשתנים a, b ו-c, ובנו טבלת מעקב אחר מהלך הביצוע של קטע התוכנית שכתבתם עבור ערכים אלה.

3.4 החלפה בין ערכי משתנים

בסעיף זה נראה כיצד לפתור בעיה אלגוריתמית בסיסית, ובפתרונה נוכל להיעזר בעתיד, בתוך אלגוריתמים אחרים. כלומר פתרונה יהווה תבנית שבה נוכל להשתמש שוב ושוב בהקשרים שונים.

בעיה 5

מטרת הבעיה ופתרונה: חידוד השימוש במשתנים, והצגת שימוש במשתנה עזר.

> במשתנים a ו-b יש ערכים התחלתיים כלשהם. כתבו אלגוריתם ובו הוראות השמה, המבצע החלפה של ערכי המשתנים. כלומר, לאחר ביצוע האלגוריתם יהיה ערכו של a שווה לערכו ההתחלתי של b וערכו של b יהיה שווה לערכו ההתחלתי של a.

הנה הצעה לפתרון:

1. *השם ב-a את ערכו של b*
2. *השם ב-b את ערכו של a*

ואחרי יישום כקטע תוכנית מחשב:

```
1. a = b;
2. b = a;
```

האם קטע זה משיג את המטרה?

נעקוב אחר מהלך ביצוע קטע התוכנית כאשר ערכו ההתחלתי של a הוא 5 וערכו ההתחלתי של b הוא 7:

מספר השורה	המשפט לביצוע	a	b
		5	7
1	a = b;	7	7
2	b = a;	7	7

לא השגנו את המטרה!

למעשה, "איבדנו" את ערכו של a בעקבות ביצוע שורה מספר 1.

? מה עלינו לעשות כדי למנוע את איבוד ערכו ההתחלתי של המשתנה a?

נגדיר משתנה נוסף, `temp` (מלשון temporary, שפירושו זמני, כלומר משתנה זמני), אשר ישמש לשמירת ערכו ההתחלתי של a במהלך ביצוע ההחלפה.

נשמור תחילה את ערכו ההתחלתי של a במשתנה הזמני `temp`. אחר כך נשים את ערכו של b ב-a, ולבסוף נשים ב-b את הערך השמור ב-`temp` (הלוא הוא ערכו ההתחלתי של a).

האלגוריתם לפתרון הבעיה יהיה:

1. *השם ב-temp את ערכו של a*
2. *השם ב-a את ערכו של b*
3. *השם ב-b את ערכו של temp*

ואחרי יישומו, נקבל את קטע התוכנית הבא:

```
temp = a;
a = b;
b = temp;
```

לשם המחשה נוכל לדמיין מצב שבו שמנו את הסוכר בכלי של המלח ואת המלח בכלי של הסוכר, על מנת להחליף ביניהם נהיה חייבים להשתמש בכלי עזר!

סוף פתרון בעיה 5

בבעיה זו למדנו שכדי להחליף ערכים של שני משתנים יש להשתמש במשתנה נוסף, שיעזור בביצוע ההחלפה, משתנה כזה נקרא משתנה עזר:

משתנה עזר הוא משתנה שנועד לסייע בביצוע משימה כלשהי.

שאלה 3.25

א. בנו שתי טבלאות מעקב אחר מהלכי ביצוע שני הפתרונות שהוצעו לבעיה 5, עבור הערכים ההתחלתיים 1 ו-2 במשתנים a ו-b: טבלה אחת עבור הפתרון השגוי של הבעיה וטבלה אחת עבור הפתרון הנכון של הבעיה.

ב. הביאו שתי דוגמאות קלט שונות אשר עבור כל אחת מהן יהיה פלט הפתרון השגוי 5 5.

ג. האם תיתכן דוגמת קלט שעבורה יהיה פלט הפתרון השגוי 6 5? נמקו.

שאלה 3.26

נתון קטע התוכנית הבא:

```
Console.Write("Enter number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
b = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
c = int.Parse(Console.ReadLine());
temp = a;
a = b;
b = c;
c = temp;
Console.WriteLine(a);
Console.WriteLine(b);
Console.WriteLine(c);
```

א. בנו טבלת מעקב אחר מהלך הביצוע של קטע התוכנית עבור הקלט 3 2 1.

ב. הביאו דוגמת קלט שהפלט המתקבל עבורה הוא 3 2 1.

ג. מהי מטרת קטע התוכנית?

ד. האם ניתן להשיג את מטרת קטע התוכנית ללא משפטי השמה כלל? אם כן, כיצד?

בעיה 5 עסקה בהחלפה של ערכי משתנים. להעמקה בתבנית ***החלפת ערכים בין שני משתנים*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 3.27

מטרת קטע התוכנית הבא היא הצגת נתוני הקלט בסדר **הפוך** לסדר קליטתם:

```
Console.Write("Enter number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
b = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
c = int.Parse(Console.ReadLine());
a = c;
c = a;
Console.Write(a);
Console.Write(b);
Console.Write(c);
```

א. הביאו שתי דוגמאות קלט שונות שעבור כל אחת מהן יוצג הפלט הדרוש.

ב. הביאו דוגמת קלט שעבורה **לא** יוצג הפלט הדרוש.

ג. תקנו את הקטע בלי לשנות את משפטי הקלט והפלט, כלומר, השלימו רק את השורות החסרות בקטע התוכנית שלהלן:

```
Console.Write("Enter number: ");
a = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
b = int.Parse(Console.ReadLine());
Console.Write("Enter number: ");
c = int.Parse(Console.ReadLine());

     השלימו כאן

Console.WriteLine(a);
Console.WriteLine(b);
Console.WriteLine(c);
```

שאלה 3.27 עסקה בהיפוך סדרת איברים. להעמקה בתבנית ***היפוך סדר האיברים בסדרה*** פנו לסעיף התבניות המופיע בסוף הפרק.

3.5 טיפוסים

בפתרון הבעיות שהוצגו עד עתה השתמשנו במספרים שלמים. יש בעיות שכדי לפתור אותן יש להשתמש במספרים שאינם בהכרח שלמים. בסעיף זה נראה דוגמאות לעיבוד מספרים ממשיים (באנגלית real), שהם ערכים מטיפוס ממשי. בסעיף 3.1 נתנו כבר הגדרה למונח טיפוס. עתה נרחיב אותה באופן הבא:

טיפוס של ערך (data type) מגדיר קבוצת ערכים ואת הפעולות שניתן לבצע על הערכים האלה.

כלומר יחד עם הערכים השייכים לטיפוס מסוים יש לציין גם את הפעולות המותרות עליהם.

בסעיף זה נראה עיבודים עם שני טיפוסי ערכים: טיפוס שלם, המוכר לנו כבר, וטיפוס ממשי.

ערכים מ**טיפוס שלם** הם המספרים השלמים, למשל: 3, 0, 700, 511-.
ערכים מ**טיפוס ממשי** הם מספרים ממשיים למשל: 5.0, 17.2, 73.1.

ערך מטיפוס ממשי כולל תמיד חלק שלם ושבר, למשל, במספר 3.5 החלק השלם הוא 3 והשבר הוא 0.5. במספר 5.0 החלק השלם הוא 5 והשבר הוא 0.

ניתן לבצע את הפעולות החשבוניות המוכרות לנו (חיבור, חיסור, כפל וחילוק) הן על ערכים מטיפוס שלם והן על ערכים מטיפוס ממשי. השימוש בפעולות אלו על ערכים מספריים יוצר ביטוי חשבוני. הזכרנו בסעיף 3.2 שביטוי חשבוני מורכב מפעולות חשבון בין ערכים מספריים מפורשים או בין משתנים השומרים ערכים מספריים. ביטוי חשבוני אף הוא מטיפוס מסוים, שלם או ממשי:

טיפוס של ביטוי חשבוני נקבע על פי טיפוס הערכים המפורשים, על פי המשתנים שבו, ועל פי הפעולות שבו. אם נכלל בביטוי לפחות ערך אחד או משתנה מטיפוס ממשי, או שנכללת בו פעולה אשר תוצאתה עשויה להיות מספר **לא** שלם, אז הביטוי הוא מטיפוס ממשי. רק אם **כל** המרכיבים של הביטוי הם מטיפוס שלם אז הביטוי הוא מטיפוס שלם, למשל: 6+7, 3*5, או num2-num1, כאשר num1 ו-num2 הוגדרו כמשתנים שלמים.

שימו ♥: קבוצת הערכים מטיפוס שלם היא בעצם תת-קבוצה של קבוצת הערכים מטיפוס ממשי. בהמשך הסעיף נדגים ונסביר את החשיבות שבהגדרת קבוצת הערכים השלמים כטיפוס נפרד.

בעיה 6

מטרת הבעיה ופתרונה: הצגת שימוש במשתנים משני הטיפוסים: שלם וממשי. היכרות עם תבנית חישוב ממוצע.

פתחו וישמו בשלבים אלגוריתם שהקלט שלו הוא ארבעה מספרים שלמים, והפלט שלו הוא ממוצע המספרים.

בדיקת דוגמאות

נוודא כי הבעיה מובנת לנו, בכך שנבחן את הפלט עבור שתי דוגמאות קלט שונות:

- עבור הקלט 40 30 20 10 הפלט הוא 25.
- עבור הקלט 14 13 12 11 הפלט הוא 12.5.

חלוקה לתת-משימות

נבצע את החלוקה הבאה לתת-משימות:

1. קליטת ארבעה מספרים שלמים.

2. חישוב סכום ארבעת המספרים.
3. חלוקת הסכום ב-4.
4. הצגה של תוצאת החילוק כפלט.

בחירת משתנים

אילו טיפוסים מתאימים לבעיה זו?

נגדיר ארבעה משתנים שבהם ייקלטו נתוני הקלט: num1, num2, num3 ו-num4. כיוון שידוע שנתוני הקלט הם מספרים שלמים, משתנים אלה יהיו מטיפוס שלם. נגדיר משתנה נוסף sum, והוא ישמור את סכום ארבעת נתוני הקלט. גם משתנה זה יהיה מטיפוס שלם.

נותר לנו להגדיר את המשתנה אשר ישמור את ממוצע ארבעת המספרים. נקרא לו average. ראינו בדוגמת הקלט/פלט השנייה שבחנו שייתכן שמשתנה זה ישמור ערך אשר איננו מספר שלם. לכן נגדיר את average כמשתנה מטיפוס ממשי.

ובסך הכול נקבל:

num1, num2, num3, num4 – מטיפוס שלם, ישמרו את נתוני הקלט.
sum – מטיפוס שלם, ישמרו את סכום נתוני הקלט.
average – מטיפוס ממשי, ישמור את ממוצע נתוני הקלט.

האלגוריתם לפתרון הבעיה יהיה:

1. *קלוט ארבעה מספרים ב-*num1, num2, num3, num4
2. *חשב את סכום ארבעת המספרים והשם את התוצאה ב-*sum
3. *חשב את ממוצע המספרים, על ידי* sum/4*, והשם את התוצאה ב-*average
4. *הצג כפלט את ערכו של* average

יישום האלגוריתם

כדי להגדיר בשפת C# משתנה מטיפוס ממשי, יש להצהיר עליו כעל משתנה מטיפוס **double**, כך:

```
double average;
```

היישום של הוראה 3 משתמש במשפט השמה הכולל ערכים שלמים וממשיים. כיצד מתפרשת הוראה כזאת ב-C#?

במשתנה מטיפוס ממשי ניתן לבצע השמה של ביטוי מטיפוס ממשי או מטיפוס שלם. אם הביטוי הוא מטיפוס שלם ערכו מומר לממשי לפני ביצוע ההשמה.

למשל, נניח ש-x משתנה מטיפוס ממשי, ונתבונן במשפט ההשמה `x = 3 + 4;`. הביטוי בצד ימין של ההשמה הוא ביטוי חשבוני שמורכב רק מערכים שלמים (3 ו-4). לכן הביטוי הזה הוא מטיפוס שלם וערכו 7. לפני ביצוע ההשמה ערכו מומר לערך הממשי המקביל 7.0, ובעקבות ההשמה ערכו של x הוא 7.0.

את הוראה 3 היינו רוצים ליישם במשפט השמה כזה:

```
average = sum / 4;
```

מאחר שכל המרכיבים של הביטוי `sum/4` הם שלמים, הרי הטיפוס של הביטוי כולו הוא שלם (בפרק הבא נלמד מה משמעות פעולת החלוקה עבור ערכי הטיפוס השלם). אבל אנו מעוניינים לחשב את הביטוי כמספר ממשי ולבצע השמה של תוצאת הביטוי (הממשית) בתוך משתנה מטיפוס ממשי. למשל, אם ערכו של `sum` הוא 10, אנו מעוניינים לשים את הערך הממשי 2.5 במשתנה `average`.

נוכל להשיג זאת אם נבקש להתייחס באופן זמני אל אחד ממרכיבי הביטוי כאל ממשי. בכך נגרום לערך הביטוי כולו להיהפך לממשי. כלומר אנו מבקשים **להמיר** את ערך המשתנה `sum` לממשי, רק לצורך חישוב הביטוי. פעולת ההמרה (casting) ב-C# מתבצעת כך: כתיבת שם הטיפוס (במקרה זה **`double`**) בתוך סוגריים, משמאל למשתנה שרוצים להמיר (במקרה זה `sum`). נרחיב עוד על המרה בפרק הבא.

לכן הדרך הנכונה ליישם ב-C# את הוראה 3 היא במשפט ההשמה הבא:

```
average = (double) sum / 4;
```

הנה התוכנית השלמה:

```
/*
התוכנית מחשבת ממוצע של ארבעה ערכים
*/
using System;
public class FourNumbersAverage
{
    public static void Main ()
    {
        int num1, num2, num3, num4;       // ארבעת נתוני הקלט
        int sum;                          // סכום נתוני הקלט
        double average;                   // ממוצע נתוני הקלט
     1. Console.Write("Enter first number: ");
     2. num1 = int.Parse(Console.ReadLine());
     3. Console.Write("Enter second number: ");
     4. num2 = int.Parse(Console.ReadLine());
     5. Console.Write("Enter third number: ");
     6. num3 = int.Parse(Console.ReadLine());
     7. Console.Write("Enter fourth number: ");
     8. num4 = int.Parse(Console.ReadLine());
     9. sum = num1 + num2 + num3 + num4;
     10.average = (double) sum /4;
     11.Console.WriteLine("The average is: {0}", average);
    } // Main
} // class FourNumbersAverage
```

נבנה טבלת מעקב אחר ביצוע מהלך התוכנית עבור הקלט 1 2 3 5:

מספר שורה	המשפט לביצוע	num1	num2	num3	num4	sum	average	פלט
1	`Console.Write("...");`	?	?	?	?	?	?	Enter first number:
2	`num1 = int.Parse(...);`	1	?	?	?	?	?	
3	`Console.Write("...");`	1	?	?	?	?	?	Enter second number:
4	`num2 = int.Parse(...);`	1	2	?	?	?	?	
5	`Console.Write("...");`	1	2	?	?	?	?	Enter third number:
6	`num3 = int.Parse(...);`	1	2	3	?	?	?	
7	`Console.Write("...");`	1	2	3	?	?	?	Enter fourth

								number:
8	num4 = **int**.Parse(...);	1	2	3	5	?	?	
9	sum = num1 + ...	1	2	3	5	11		
10	average = ...	1	2	3	5	11	2.75	
11	Console.WriteLine("... ");	1	2	3	5	11	2.75	The average is 2.75

הפלט המתקבל בשורה האחרונה הוא: `The average is 2.75`

סוף פתרון בעיה 6

בפתרון שהוצג לבעיה 6 למדנו כמה עובדות חשובות על משתנים ועל טיפוסי משתנים:

> מרכיב חשוב בהגדרת הייעוד של משתנה הוא הגדרת סוג הערכים שיישמרו במשתנה. סוג ערכים זה נקבע באמצעות טיפוס המשתנה. חשוב להתאים את טיפוס המשתנה לתפקידו.
>
> משתנה שומר ערכים מטיפוס (סוג) אחד בלבד.
>
> יש להצהיר בנפרד על משתנים מטיפוסים שונים.
>
> בשפת C# מצהירים על משתנה מטיפוס שלם באמצעות המילה **int** ועל משתנה מטיפוס ממשי באמצעות המילה **double**.
>
> במשפט השמה ניתן לשים במשתנה מטיפוס שלם רק ערך מטיפוס שלם, ואילו במשתנה מטיפוס ממשי ניתן לשים גם ערך מטיפוס שלם וגם ערך מטיפוס ממשי.

ניתן לקלוט ערך בתוך משתנה ממשי בדומה לקליטת ערך בתוך משתנה שלם. במקום להשתמש בהוראה `int.Parse` נשתמש בהוראה `double.Parse`. למשל, כך:

```
double x;
Console.Write("Enter a real number: ");
x = double.Parse(Console.ReadLine());
```

שאלה 3.28

בנו טבלת מעקב אחר ביצוע התוכנית `FourNumbersAverage` עבור הקלט 5 6 8 1.

שאלה 3.29

פתחו בשלבים אלגוריתם שהקלט שלו הוא שני מספרים ממשיים והפלט שלו הוא שורה שמופיעות בה תוצאות החילוק ב-4 של כל אחד משני המספרים, ושורה שבה מופיע סכום תוצאות החילוק. ישמו את האלגוריתם בשפת C#.

למשל, עבור הקלט 2.84 1.6 הפלט הוא: 0.71 0.4

1.11

שימו ♥ לבחירת טיפוסי המשתנים!

שאלה 3.30

פתחו בשלבים אלגוריתם שהקלט שלו הוא מחיריהם של שלושה מוצרים בשקלים. הפלט שלו הוא מחיר כולל המתקבל מסכום שלושת המחירים בתוספת מס בשיעור 20%.

בעיה 6 עסקה בחישוב ממוצע. להעמקה בתבנית ***ממוצע של סדרת מספרים*** פנו לסעיף התבניות המופיע בסוף הפרק.

כדאי לדעת – סיבות לאבחנה בין שלם לממשי:

אמנם המספרים השלמים הם תת-קבוצה של המספרים הממשיים, אבל כאשר אנו יודעים כי משתנה מסוים עתיד להכיל רק ערכים שלמים, רצוי להגדירו מטיפוס שלם ולא מטיפוס ממשי. יש לכך שלוש סיבות:

- בכך שנצהיר על טיפוסו המדויק של משתנה נסייע בהבנת תפקידו ובכך נהפוך את התוכנית לבהירה ולקריאה יותר.
- ערכים מטיפוס שלם וערכים מטיפוס ממשי מיוצגים בזיכרון המחשב באופן שונה. משום כך, ביצוע פעולות חישוב על ערכים מטיפוס שלם הן פשוטות ומהירות יותר מביצוע אותן פעולות על ערכים מטיפוס ממשי.
- הגדרת הטיפוסים משמשת את המהדר (הקומפיילר) לבדיקת ההתאמה של משפטי השמה. המהדר בודק אם טיפוס הביטוי המחושב בצד ימין מתאים לטיפוס הביטוי שמבצעים בו את ההשמה. משום כך, הצהרה מדויקת על טיפוסו של משתנה יכולה לסייע לנו באיתור שגיאות.

3.6 קבועים

בסעיף זה נכיר הרגל תכנותי שמסייע ליצור תוכניות בהירות, קריאות ועמידות בפני שגיאות.

כיתה י"ב 3 מתכוננת לסיום לימודיה בבית הספר התיכון. בכיתה 36 תלמידים. הגזבר של הכיתה מקבל הצעות מחיר ממארגני אירועים עבור סעיפים שונים בתכנון אירועי חגיגות הסיום. למשל, עלות רכישת חולצות עם הדפס שנבחר על ידי הכיתה, עלות הדפסת ספר מחזור, עלות צריבת דיסק עם השיר שהקליטו לכבוד המסיבה ועוד ועוד... מארגני האירועים נותנים הצעת מחיר לתלמיד יחיד, וכדי לחשב את עלותה עבור הכיתה כולה יש להכפיל במספר התלמידים.

אחת התלמידות בכיתה כתבה לגזבר תוכנית שתסייע לו בחישוב עלות הסעיפים השונים. הגזבר יוכל לתת לתוכנית כקלט את ההצעות שקיבל עבור הסעיפים השונים, והתוכנית תיתן כפלט את העלות של כל אחד מהסעיפים עבור הכיתה כולה ואת העלות הכוללת של כל הסעיפים.

הנה שלוש ההוראות הראשונות בקטע התוכנית. הקטע מחשב את העלות עבור כל אחד מהסעיפים לכיתה כולה:

```
totalShirtPrice = shirtPrice * 36;
totalBookPrice = bookPrice * 36;
totalDiskPrice = diskPrice * 36;
```

מאחר שבבית ספר זה נוהגים התלמידים לחגוג את סיום לימודיהם באופן מוגזם משהו, יש בקטע התוכנית הזה עוד שורות רבות...

קטע התוכנית הזה מתאים כמובן רק עבור כיתה י"ב 3, משום שהוא מתייחס באופן ישיר למספר התלמידים בכיתה (36). אם תרצה גם כיתה י"ב 6, ובה 37 תלמידים, להשתמש בתוכנית, תצטרך התלמידה שכתבה את התוכנית לעבור עליה ולשנות בכל מקום את הערך 36 ל-37. אמנם סביר שמספר הסעיפים אינו באמת גדול מאוד, ולכן שינוי זה לא יגזול זמן רב, אך עדיין ייתכן כי תטעה ותשכח לשנות את אחד הערכים באחד המקומות. בכך התוכנית תהפוך לשגויה, ותיתן פלט שגוי.

בתוכנית גדולה ומורכבת (לדוגמה: מערכת לניווט לוויינים), הצורך לשנות ערך שמופיע באופן מפורש במקומות רבים בתוכנית, מאות פעמים או אפילו אלפים, הוא בעייתי מאוד, ויש להימנע ממנו במידת האפשר כדי להפוך את התוכנית לעמידה יותר.

הדרך להימנע מכך, היא לתת לערך זה שם, ובכל מקום בתוכנית להשתמש בשמו ולא בערכו. רק במקום אחד בתוכנית נְקַשֵּׁר בין השם לערך. אם יהיה צורך בשינוי הערך, השינוי יתבצע רק במקום אחד. בשפת C# נוכל לעשות זאת על ידי שימוש ב**קבוע**.

הצהרה על קבוע בשפת C# דומה להגדרת משתנה. גם עבור קבוע מוקצה מקום בזיכרון ובו נשמר ערכו. אלא שלא כמו עבור משתנה, ערכו של קבוע לא ניתן לשינוי במהלך התוכנית. למשל בתחילת התוכנית שמחשבת את עלות חגיגות הסיום נוכל לכתוב את המשפט הבא:

```
const int NUM_OF_STUDENTS = 36;       // קבוע – מספר התלמידים בכיתה
```

מעתה נקפיד להשתמש בקבועים בתוכניות שנכתוב, כפי שמודגם כבר בתוכנית הראשונה בפרק הבא, פרק 4.

קבוע (constant) הוא תא זיכרון, אשר ערכו ההתחלתי לא ניתן לשינוי לאחר שאותחל.

כדי להצהיר על קבוע נכתוב את המילה **const** בתחילת שורת ההצהרה, למשל:

```
const int X = 5;
```

נהוג לכתוב את שם הקבוע באותיות גדולות. אם שם הקבוע כולל יותר ממילה אחת, נהוג להפריד את המילים בקו תחתון.

סיכום

בפרק זה פיתחנו אלגוריתמים בסיסיים לביצוע במחשב, ויישמנו אותם בתוכנית מחשב בשפת C#.
הכרנו את אבני הבניין של אלגוריתמים למחשב: משתנים, אשר בהם נשמרים נתונים ותוצאות חישוב, הוראות קלט והוראות פלט, אשר מורות על קליטה של נתונים והצגה של נתונים כפלט, והוראות השמה, אשר מורות על ביצוע חישובים ועל שמירת תוצאותיהם במשתנים.

משתנה (variable) הוא תא זיכרון אשר במהלך ביצוע אלגוריתם ניתן לשמור בו ערך ולקרוא את הערך השמור בו. הערך השמור במשתנה נקרא **ערך המשתנה**. פנייה למשתנה מתבצעת באמצעות שם הניתן לו על ידי מְפַתֵּחַ האלגוריתם, שם זה הוא **שם המשתנה**.

במשתנה נשמרים ערכים מטיפוסים אשר מתאימים לתפקידו של המשתנה. הטיפוס של הערכים הנשמרים במשתנה הוא **טיפוס המשתנה**.

טיפוס של ערך (data type) מגדיר אוסף של ערכים אפשריים ואת הפעולות שניתן לבצע עליהם.
בפרק זה הכרנו ערכים מטיפוס שלם (כלומר, מספרים שלמים) וערכים מטיפוס ממשי (כלומר, מספרים ממשיים). כל טיפוס מיוצג בזיכרון המחשב בצורה שונה.

בחירת טיפוס של משתנה נעשית כחלק מהגדרת הייעוד של המשתנה. סוג הערכים נקבע על פי נתונים שייקלטו במשתנה או על פי ערכים (של פעולות חישוב) שיושמו במשתנה, למשל כאשר יש לשמור במשתנה תוצאת ממוצע של שני מספרים, יהיה מתאים להגדירו כמשתנה מטיפוס ממשי.

כדי לעדכן את ערכו של משתנה נשתמש ב**פעולת השמה**. בפעולה זו מחושב תחילה ערכו של הביטוי הנמצא בצד ימין של הוראת ההשמה. תוצאת החישוב נשמרת במשתנה הרשום בצד שמאל של הוראת ההשמה. הביטוי לחישוב יכול להיות ביטוי פשוט כגון ערך מפורש או שם של

משתנה, או יכול להיות ביטוי המורכב מפעולות חשבוניות שונות בין ערכים ובין משתנים. משפט השמה משפיע **רק** על ערכו של המשתנה שישמור את תוצאת החישוב, ולא על משתנים אחרים המעורבים בביטוי המחושב. המשתנה שישמור את תוצאת החישוב יכול בעצמו להופיע כחלק מהביטוי המחושב. במקרה זה ערכו החדש תלוי בערכו הישן.

מתן ערך התחלתי למשתנה נקרא **אתחול**.

קליטת נתונים נעשית על ידי **הוראת קלט**, המבצעת השמה של נתון שנקרא מהקלט בתוך משתנה.

הצגת נתונים נעשית באמצעות **הוראת פלט**, ובאמצעותה ניתן להציג הודעות, ערכי משתנים וערכי ביטויים כפלט.

ביטוי אשר מורכב מפעולות חשבון בין ערכים ובין משתנים מטיפוס שלם או ממשי נקרא **ביטוי חשבוני**. **טיפוס של ביטוי חשבוני** נקבע על פי טיפוסי הערכים והמשתנים שבו, ועל פי הפעולות שבו.

פתרון בעיה אלגוריתמית נעשה בשלבים:

1. בחינת **דוגמאות קלט** שונות והבנת הקשר בין הקלט לפלט.
2. חלוקה של משימות האלגוריתם ל**תת-משימות**.
3. בחירת **משתנים** – תפקיד, שם וטיפוס לכל משתנה.
4. כתיבת ה**אלגוריתם**.
5. כתיבת התוכנית **ליישום** האלגוריתם בשפת התכנות.

לאחר כתיבת התוכנית כדאי לבצע **מעקב** אחר מהלך ביצועה עבור דוגמאות קלט מגוונות, כדי להשתכנע בנכונותה.

מעקב מסודר אחר מהלך ביצוע של אלגוריתם או של תוכנית נעשה באמצעות **טבלת מעקב**. בטבלת מעקב מפורטים השינויים בערכי המשתנים, ומפורט הפלט בעקבות ביצוע כל אחת ואחת מהוראות האלגוריתם או התוכנית.

ערך התחלתי של משתנה ביחס לתוכנית או לקטע תוכנית הוא הערך השמור בו מיד לפני תחילת ביצוע אותה תוכנית או אותו קטע תוכנית. **מצב התחלתי** של תוכנית או של קטע תוכנית מתאר את ערכם ההתחלתי של כל המשתנים לפני תחילת הביצוע.

המצב ההתחלתי מתואר בשורה הראשונה של טבלת המעקב. עבור תוכנית שלמה הערכים ההתחלתיים של המשתנים שלא אותחלו עם הצהרתם אינם ידועים (נסמן בסימן שאלה (?)). עבור קטע תוכנית נקבל מראש את הערכים ההתחלתיים של כל המשתנים ונכתוב אותם בשורה הראשונה של טבלת המעקב.

בכל אלגוריתם או תוכנית כדאי לכלול **תיעוד** כדי להסבירם לקורא. יש לצרף לכותרת האלגוריתם או התוכנית הערה המתארת את המטרה, כלומר, את המשימה שהפתרון מיועד למלא. את שמות המשתנים נבחר על פי תפקידיהם, ונוסיף הערות המתארות את תפקידיהם. ההערות מיועדות לקורא בלבד.

במהלך הפרק הצגנו שאלות רבות ומגוונות, שאפשר לחלק לשני סוגים:

- שאלות **פיתוח** ויישום של אלגוריתם.
- שאלות **ניתוח** אלגוריתם או קטע תוכנית נתון.

שאלות הפיתוח והיישום דורשות פיתוח מלא בשלבים או פיתוח חלקי (כלומר עד שלב החלוקה לתת-משימות או עד שלב בחירת המשתנים או עד שלב כתיבת האלגוריתם, ללא יישום). שאלות הניתוח דורשות מעקב אחר מהלך ביצוע עבור קלט נתון, הבאת דוגמת קלט עבור פלט נתון, תיאור מטרת קטע תוכנית או משימות ניתוח אחרות. ההתנסות בשני הסוגים של השאלות מפתחת את היכולת להבין אלגוריתמים ולפתחם, ובעקבות כך מפתחת את היכולת לפתרון בעיות.

בפרק הבא נפתור בעיות מורכבות יותר מהבעיות שהוצגו בפרק זה ונרחיב בפירוט את השלבים השונים של תהליך פיתוח אלגוריתם ויישומו בתוכנית. בפרקים הבאים אחר כך יוצגו בעיות מורכבות יותר ויותר. כדי להתמודד עם בעיות מורכבות חשובה לא רק היכולת לכתוב תוכנית מחשב, אלא חשובות גם היכולת להתקדם בשלבים והיכולת לנתח ולהבין אלגוריתם נתון (או קטע תוכנית).

סיכום מרכיבי שפת C# שנלמדו בפרק 3

בחלק זה נפרט את כללי שפת C# שלמדנו בפרק 3. פרט להוראות הקלט, הכללים המוצגים כאן הם הכללים של שפת C# הסטנדרטית, ואינם הכללים של סביבת עבודה מסוימת כגון Visual C#. איננו יכולים להניח שכל המשתמשים בספר זה עובדים באותה הסביבה. לכן לאורך הספר כולו אנו מציגים את כללי שפת C# הסטנדרטית. עם זאת, מאחר שהוראות הקלט מהמקלדת בשפת C# הן מורכבות למדי, בחרנו לחרוג מכלל הסטנדרטיות בנקודה זו.

מבנה תוכנית בשפת C#

- תוכנית בשפת C# היא אוסף של **מחלקות** (מחלקה אחת או יותר). אחת המחלקות היא המחלקה הראשית. **המחלקה הראשית** מכילה את הפעולה הראשית (**Main**), שממנה ביצוע התוכנית מתחיל ובתחומה משפטי התוכנית נכתבים באופן הבא:

```
public class TheNameOfTheProgram
{
    public static void Main ()
    {
        גוף התוכנית
    }  // Main סוף
}  // סוף המחלקה הראשית
```

- כל הוראה בתוכנית מסתיימת בסימן ; (נקודה-פסיק).

הערות

בין המשפטים השונים של התוכנית מופיעות **הערות** (comments). ההערות מיועדות למתכנת ולמשתמש בתוכנית, אך לא למחשב. הן עוזרות בקריאת התוכנית ובהבנתה. הערה המתחילה בסימן // נמשכת עד סופה של השורה. הערה התחומה בין הסימנים */ ו-/* יכולה להתפרש על פני כמה שורות. למשל:

```
/*  תוכנית לחישוב ממוצע  */
// משתנה השומר את הממוצע
```

שמות

- המתכנת בוחר את **שמות** מרכיבי התוכנית שהגדיר (שמות המשתנים, שמות המחלקות ובפרט שם המחלקה הראשית, כלומר שם התוכנית). אנו נוהגים לכנות משתנים בשמות משמעותיים, ולעתים נצמיד שתי מילים או יותר כדי ליצור שם משמעותי וברור יותר.

- על פי ה**מוסכמות** המקובלות במתן שמות, לא נהוג לקצר שמות. למשל אם המשתנה מתעתד להכיל בתוכו ממוצע נעדיף לקרוא לו `average` ולא `avg`. כך גם בשמות של מחלקות.
- שם מחלקה יתחיל באות גדולה, שם משתנה יתחיל באות קטנה. שאר האותיות יהיו קטנות, אלא אם השם מורכב מכמה מילים שהוצמדו זו לזו. במקרה זה נשתמש באות גדולה בראש כל מילה פרט למילה הראשונה. למשל `numOfChildren`, `sum`, `ReadWrite` וכדומה. הקפדה על כללים אלה יוצרת תוכנית ברורה וקריאה יותר.
- קיימת ב-C# קבוצת שמות מיוחדת הנקראת **מילים שמורות** (reserved words). לשמות אלה יש משמעות מוגדרת ב-C#, ואסור להשתמש בהם לשמות אחרים. למשל אסור להשתמש במילה **`class`** כשם של משתנה כי היא מילה שמורה עבור מחלקה. בתוכניות המופיעות בספר זה, מילה שמורה כתובה באותיות מודגשות.
- ההצהרות, השמות ומרכיבי ההוראות בתוכנית חייבים להיות מופרדים בתו רווח אחד לפחות. למשל, **לא** נוכל לכתוב: **`publicclass`**`MyProgram`.

ערכים ונתונים בתוכנית C#

טיפוסים

- הערכים המופיעים בתוכנית וערכי המשתנים והקבועים מסווגים ל**טיפוסים** (types). הטיפוסים שהכרנו עד עכשיו הם שלם (**`int`**) וממשי (**`double`**).
- **ערך מטיפוס שלם** המופיע בתוכנית C# הוא מספר שלם כשלשמאלו יכול להופיע הסימן פלוס (+) או הסימן מינוס (-). אם המספר שלילי יש לכתוב את הסימן מינוס. כתיבת הסימן פלוס אינה הכרחית (זוהי ברירת המחדל, כלומר אם לא כתוב אף סימן, המספר מפורש כמספר חיובי). אלה לדוגמה ערכים חוקיים מטיפוס שלם בשפת C#: 156, 156+, 3-, 0.
- **ערך מטיפוס ממשי** המופיע בתוכנית C# מורכב מארבעה חלקים:
  1. סימן + או הסימן - (כתיבת הסימן + אינה הכרחית כאשר המספר חיובי).
  2. סדרה לא ריקה של ספרות המייצגת את החלק השלם של המספר.
  3. נקודה עשרונית.
  4. סדרה לא ריקה של ספרות המייצגת את השבר של המספר.

  אלה לדוגמה ערכים חוקיים מטיפוס ממשי בשפת C#: 1.53, 0.2-, 7.0, 5.3+. הערכים 5. או 3. אינם ערכים ממשיים חוקיים.
- מספרים שלמים יכולים להיות מיוצגים בשפת C# בערכים שטיפוסם שלם או בערכים שטיפוסם ממשי. המספר השלם שלוש למשל יכול להיות מיוצג בערך 3 שטיפוסו שלם, וגם בערך 3.0 שטיפוסו ממשי.

משתנים

- **הצהרה על משתנים** תתבצע באמצעות הצהרה על הטיפוס ועל שמו של המשתנה. המילה **`int`** משמשת להצהרה על משתנה מטיפוס שלם, והמילה **`double`** משמשת להצהרה על משתנה מטיפוס ממשי.

  לדוגמה:

```
int num;
```

- ניתן להצהיר על כמה משתנים באותה השורה אם תפקידם דומה, בהפרדה בפסיקים. לדוגמה:

```
double num1, num2;
```

קבועים

קבוע הוא תא זיכרון שערכו לא יכול להשתנות לאחר שנקבע. ההצהרה על קבוע נעשית בדומה להצהרת משתנה, אך מקדימה אותה המילה **const** למשל:

```
const int MY_CONSTANT_INTEGER = 3;
```

הוראות ביצוע של תוכנית בשפת C#

קלט

- ליישום **הוראת קלט של ערך שלם** נשתמש בפעולה `int.Parse(Console.ReadLine())`.
  ל**הוראת קלט של ערך ממשי** נשתמש בפעולה `double.Parse(Console.ReadLine())`.
  הוראת הקלט כוללת שם של משתנה שיישמר בו הערך הנקלט. למשל המבנה הכללי של הוראת קלט של ערך שלם הוא:

```
שם משתנה = int.Parse(Console.ReadLine())
```

- ביצוע פעולת קלט גורם לעצירת התוכנית עד לקליטת ערך מתאים. לאחר קליטתו הוא נשמר בתוך המשתנה.
- נזכור לכתוב לפני פעולת הקלט פעולת פלט המנחה את המשתמש לגבי הקלט שהתוכנית מצפה לקבל, למשל:

```
Console.Write("Enter a positive integer number: ");
x = int.Parse(Console.ReadLine());
```

פלט

- **הוראת פלט** מיושמת ב-C# באמצעות הפעולה `Console.Write` (שאינה גורמת למעבר לשורה הבאה בפלט) או הפעולה `Console.WriteLine` (שגורמת למעבר לשורה הבאה לאחר הצגת הפלט המבוקש).

  למשל:

```
Console.Write("a message");
Console.WriteLine(x);
```
וכאשר x משתנה

שם הפעולה המלא הוא `System.Console.WriteLine`. כדי שנוכל לכתוב בצורה המקוצרת נוסיף בראש התוכנית את ההוראה הבאה:

```
using System;
```

- ניתן לצרף פריטים נוספים להודעה שברצוננו להציג. בתוך ההודעה נסמן את המקום שאמורים להשתלב בו הפריטים הנוספים ולאחר ההודעה נצרף את רשימת הפריטים, למשל:

```
Console.Write("The sum of {0} and {1} is: {2}", num1, num2, sum);
```

השמה

- המבנה הכללי של **משפט השמה** ב-C# הוא:

```
משתנה = ביטוי;
```

- **הביטוי** המופיע מימין לסימן =, הוא ערך מפורש, משתנה או ביטוי מורכב. אם הביטוי הוא ביטוי חשבוני סדר הקדימויות של פעולות החשבון זהה לסדר הקדימויות המקובל במתמטיקה.
- במשתנה מטיפוס ממשי אפשר לשים ערך מטיפוס שלם או ממשי. במשתנה מטיפוס שלם אפשר לשים רק ערכים שלמים.

שאלות נוספות

שאלות נוספות לסעיף 3.1

1. כתבו תוכנית להדפסת האות L מכוכביות באופן הבא:

```
*
*
* * *
```

2. נתון קטע התוכנית הבא:

```
left = int.Parse(Console.ReadLine());
right = int.Parse(Console.ReadLine());
Console.WriteLine("{0} {1}", right, left);
```

נניח שנתוני הקלט שהוקלדו הם 10 8:

א. מה יהיו ערכי המשתנים לאחר ביצוע משפטי הקלט?

ב. מה יהיה הפלט?

3. נתון קטע התוכנית הבא:

```
num1 = int.Parse(Console.ReadLine());
num2 = int.Parse(Console.ReadLine());
num3 = int.Parse(Console.ReadLine());
Console.WriteLine("{0} {1}", num2, num3);
Console.WriteLine("{0} {1} {2}", num3, num1, num3);
```

תנו דוגמת קלט שהפלט עבורה הוא:

```
5  9
9  5  9
```

4. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא שלושה מספרים שלמים, והפלט שלו הוא שלוש שורות של מספרים: בשורה הראשונה יופיע נתון הקלט השלישי, בשורה השנייה יופיע נתון הקלט השלישי והשני, ובשורה השלישית יופיע נתון הקלט השלישי השני והראשון. ישמו את האלגוריתם בשפת C#.

שאלות נוספות לסעיף 3.2

1. כתבו משפטי השמה לביצוע הפעולות הבאות:

א. השמה במשתנה a של סכום ערכי המשתנים c ו-b.

ב. השמה במשתנה c של ההפרש בין פעמיים ערכו של המשתנה d ובין ערך המשתנה b.

ג. השמה במשתנה e של סכום ערכו של המשתנה a וחמש פעמים ערכו של המשתנה f.

2. מחיר כרטיס כניסה לבריכת השחייה העירונית הוא 20 ₪ למבוגר ו-12 ₪ לילד. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר מבוגרים ומספר ילדים, והפלט שלו הוא הסכום לגבייה עבור הכרטיסים. ישמו את האלגוריתם בשפת C#.

שאלות נוספות לסעיף 3.3

1. נתון משפט ההשמה הבא: `a = a + b + c + d;`
   א. כתבו במקום משפט השמה זה סדרה של משפטי השמה אשר הביטוי בצד ימין של כל אחד מהם כולל סימן חיבור אחד בלבד. בסיום ביצוע סדרת המשפטים ערכו של a יהיה זהה לערכו המתקבל לאחר ביצוע המשפט הנתון. בצעו את המשימה ללא הוספת משתנים.
   ב. בנו טבלת מעקב אחר מהלך הביצוע של סדרת משפטי ההשמה שכתבתם בסעיף א עבור הערכים ההתחלתיים 1, 5, 2 ו-4 במשתנים a, b, c ו-d בהתאמה.

2. כתבו משפטי השמה לביצוע הפעולות הבאות:
   א. הקטנת ערכו של a ב-5.
   ב. הגדלת ערכו של b פי 3.
   ג. הקטנת ערכו של המשתנה a בערכו של המשתנה b.

3. עבור כל זוג משפטי השמה נתון כתבו משפט השמה **אחד** שמשיג אותה מטרה:

א. `n = m + 5;` `n = n - 2;`	**ב.** `a = (a + 2)* 3;` `a = a * 4 - 9;`
ג. `v = v + w;` `v = v * 5;`	**ד.** `x = x - 2;` `x = x - 3;`

שאלות נוספות לסעיף 3.4

1. נתון קטע התוכנית הבא:

```
x = int.Parse(Console.ReadLine());
y = int.Parse(Console.ReadLine());
x = x + y;
y = x – y;
x = x – y;
Console.WriteLine("{0} {1}", x, y);
```

   א. מהו פלט קטע התוכנית עבור הקלט 5 13? היעזרו בטבלת מעקב למציאת הפלט.
   ב. נסחו את הבעיה האלגוריתמית שקטע תוכנית זה פותר.
   ג. כתבו קטע תוכנית אחר לפתרון הבעיה.

2. מטרת סדרת המשפטים הבאה היא כי אחרי ביצועה יהיו במשתנים e, d, c ו-b ערכי המשתנים d, c, b, ו-a בהתאמה.

```
b = a;
c = b;
d = c;
e = d;
```

   א. תנו דוגמה של ערכים התחלתיים עבור המשתנים אשר המטרה לא מושגת עבורם.
   ב. מה מתבצע בסדרת המשפטים הנתונה? האם היא גורמת ל"אובדן" ערכי משתנים?

ג. כתבו סדרת משפטים שעבורה תושג המטרה.

שאלות נוספות לסעיף 3.5

1. במשרדי הממשלה עבור כל טופס שפקיד ממלא הוא מקבל שכר של 6.3 ₪ .
פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הטפסים שעל הפקיד למלא, והפלט שלו הוא השכר שהפקיד יקבל. ישמו את האלגוריתם בשפת #C.
למשל עבור הקלט 55 הפלט הדרוש הוא 346.5.

2. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר חיובי שלם המציין אורך צלע של ריבוע, והפלט שלו הוא שטח העיגול החסום בריבוע. ישמו את האלגוריתם בשפת #C.

3. עקב איחור בעונת הגשמים החליטו יצרני המטריות על מבצע מכירות בהנחה. יש לפתח וליישם אלגוריתם אשר הקלט שלו הוא מחיר מטרייה, אחוז ההנחה למטרייה, ומספר מטריות מבוקש, והפלט שלו הוא הסכום הכולל לתשלום.

נבחר את המשתנה הבא מטיפוס שלם:

num – ישמור את מספר המטריות המבוקש

ואת המשתנים הבאים מטיפוס ממשי:

price – ישמור את המחיר של המטרייה
discount – ישמור את אחוז ההנחה
newPrice – ישמור את המחיר של מטריה אחת לאחר הנחה
total – ישמור את הסכום הכולל לתשלום

פותח אלגוריתם שהוראותיו מיושמות במשפטי התוכנית הבאים:

```
Console.Write("How many umbrellas?: ");
num = int.Parse(Console.ReadLine());
Console.Write("Enter the price of one umbrella: ");
price = double.Parse(Console.ReadLine());
Console.Write("Enter discount percentage per umbrella: ");
discount = double.Parse(Console.ReadLine());
newPrice = _______________ ;
total = _________________ ;
Console.WriteLine("Total sum is {0}", total);
```

השלימו את משפטי התוכנית.

שאלות מסכמות לפרק 3

1. ניתן להמיר ערך של טמפרטורה המיוצג במעלות פרנהייט (F) לייצוג במעלות צלזיוס (C) על ידי הנוסחה: C = 5/9 (F-32).
פתחו בשלבים אלגוריתם אשר הקלט שלו הוא טמפרטורה הנתונה במעלות פרנהייט, והפלט שלו הוא ערך הטמפרטורה במעלות צלזיוס. ישמו את האלגוריתם בשפת #C.

2. פתחו בשלבים (אין צורך ליישם בתוכנית) אלגוריתם אשר הקלט שלו הוא אורכי שני הניצבים והיתר במשולש ישר זווית, והפלט שלו הוא היקף המשולש ושטח המשולש.

3. פתחו בשלבים אלגוריתם שהקלט שלו הוא שלושה מספרים שלמים, והפלט שלו הוא כל הסידורים האפשריים של שלושת המספרים. הניחו כי שלושת המספרים שונים זה מזה. ישמו את האלגוריתם בשפת C#.

הדרכה: חלקו את הפלט לשלושה חלקים: הסידורים שמתחילים בנתון הקלט הראשון, הסידורים שמתחילים בנתון הקלט השני, הסידורים שמתחילים בנתון הקלט השלישי.

למשל, עבור הקלט: 30 8 1 הפלט המתאים הוא:

```
1  8  30
1  30  8
8  1  30
8  30  1
30  1  8
30  8  1
```

4. נתון קטע התוכנית הבא:

```
a = int.Parse(Console.ReadLine());
b = int.Parse(Console.ReadLine());
a = a + b;
Console.WriteLine(a);
a = a – 2 * b;
Console.WriteLine(a);
a = a + b;
Console.WriteLine(a);
```

א. מהו פלט קטע התוכנית עבור הקלט 2 3? היעזרו בטבלת מעקב כדי לענות על השאלה.

ב. תנו דוגמת קלט אשר הפלט עבורה הוא 2 1 3.

ג. נסחו במילים את היחס שמוגדר בקטע התוכנית בין הפלט לקלט.

5. הזזה מעגלית של סדרת ערכים משמעותה העברת הערך האחרון בסדרה לתחילתה. למשל לאחר ביצוע הזזה מעגלית על הסדרה 4 3 2 1 מתקבלת הסדרה: 3 2 1 4. הזזה מעגלית היא תבנית שיכולה לשמש בפתרון בעיות אלגוריתמיות שונות.

נתון קטע התוכנית הבא שהקלט שלו הוא שלושה מספרים ומטרתו היא לתת כפלט את תוצאת ההזזה המעגלית על סדרת נתוני הקלט:

```
Console.Write("Enter first element: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter second element: ");
y = int.Parse(Console.ReadLine());
Console.Write("Enter third element: ");
z = int.Parse(Console.ReadLine());
x = y;
y = z;
z = x;
Console.WriteLine("{0} {1} {2}", x, y, z);
```

קטע התוכנית שגוי.

א. הסבירו מדוע הקטע שגוי.

ב. תקנו את התוכנית על ידי שינוי החלק של משפטי ההשמה (מבלי לשנות את משפט הפלט).
ג. תקנו את קטע התוכנית על ידי ביטול משפטי ההשמה ועל ידי שינוי משפט הפלט.

שאלה 5 עסקה בהזזה מעגלית של סדרת איברים. העמקה בתבנית ***הזזה מעגלית בסדרה*** נמצאת בסעיף הבא.

תבניות – פרק 3

פירוט מלא של התבניות ושל שאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט .

החלפת ערכים בין שני משתנים

שם התבנית: **החלפת ערכים בין שני משתנים**
נקודת מוצא: שני ערכים במשתנים element1 ו-element2
מטרה: החלפת הערכים ההתחלתיים בין שני המשתנים
אלגוריתם:
1. *השם ב-*temp *את ערכו של* element1
2. *השם ב-*element1 *את ערכו של* element2
3. *השם ב-*element2 *את ערכו של* temp

היפוך סדר האיברים בסדרה

שם התבנית: **היפוך סדר האיברים בסדרה**
נקודת מוצא: שני ערכים במשתנים element1 ו-element2
מטרה: היפוך הערכים בין שני המשתנים
אלגוריתם:
החלף את ערכי element1 ו-element2

ממוצע של סדרת מספרים

שם התבנית: **ממוצע של סדרת מספרים**
נקודת מוצא: שני מספרים ב-num1 ו-num2
מטרה: חישוב הממוצע של שני המספרים
אלגוריתם:
*השם ב-*sum *את ערכו של הביטוי החשבוני* num1 + num2
*השם ב-*average *את ערכו של הביטוי החשבוני* sum / 2

הזזה מעגלית בסידרה

שם התבנית: **הזזה מעגלית שמאלה בסדרה**
נקודת מוצא: שלושה ערכים במשתנים element1, element2 ו-element3
מטרה: הזזה מעגלית שמאלה של שלושת המשתנים
אלגוריתם:
החלף את ערכי המשתנים element1 ו-element2
החלף את ערכי המשתנים element2 ו-element3

שם התבנית: **הזזה מעגלית ימינה בסדרה**
נקודת מוצא: שלושה ערכים במשתנים element1, element2 ו-element3
מטרה: הזזה מעגלית ימינה של שלושת המשתנים
אלגוריתם:
החלף את ערכי המשתנים element2 ו-element3
החלף את ערכי המשתנים element1 ו-element2

פרק 4 – הרחבה בפיתוח אלגוריתמים

בפרק הקודם הצגנו את התהליך של פיתוח ויישום אלגוריתם בשלבים. בפרק זה נרחיב בכמה מהשלבים: בניתוח הבעיה, בפירוק המשימה לתת-משימות, ובבדיקת הפתרון עבור דוגמאות קלט שונות. הבעיות שיוצגו בפרק זה יהיו מורכבות יותר מהבעיות שהוצגו בפרק הקודם, וכך, ככל שנתקדם בפרקי הלימוד, הבעיות יהיו מורכבות יותר ויותר, וחשיבות הפתרון בשלבים תהיה משמעותית יותר ויותר.

באמצעות הבעיות שיוצגו בפרק זה ופתרונן, נכיר גם פעולות נוספות על ערכים מטיפוס שלם, טיפוס חדש שערכיו הם תווים, ואת המחלקה האחראית לפעולות מתמטיות בשפת C#.

4.1 מבט נוסף אל התהליך של פיתוח אלגוריתם ויישומו

כזכור, תהליך של פיתוח אלגוריתם ויישומו בתוכנית מחשב כולל את השלבים הבאים:

1. ניתוח ראשוני של הבעיה בעזרת דוגמאות
2. פירוק הבעיה לתת-משימות
3. בחירת משתנים, הגדרת תפקידיהם וטיפוסי הערכים שיישמרו בהם
4. כתיבת האלגוריתם
5. יישום האלגוריתם בתוכנית מחשב
6. בדיקת נכונות עבור דוגמאות קלט מגוונות בטבלת מעקב
7. כתיבת התוכנית המלאה, ובדיקתה בהרצה על דוגמאות קלט נוספות

בעת פיתוח האלגוריתם אנו נשפר ונשנה אותו. לעתים בעת העבודה על שלב, מתברר כי יש לתקן שלב קודם (לדוגמה, בעת כתיבת האלגוריתם, מסתבר כי יש להוסיף משתנה). במקרה כזה נחזור אחורה לשלב הקודם ונתקן לפני שנמשיך לשלב הבא. כמו כן, בבדיקת התוכנית עלולות להתגלות שגיאות, ותיקונן עשוי להביא לבחירת משתנים נוספים ולשינוי הוראות האלגוריתם. שיפור של אלגוריתם ושל תוכנית או תיקונים תוך חזרה משלב מתקדם לשלב קודם הוא תהליך טבעי ומקובל.

שימו ♥ לשני השלבים האחרונים:
בשלב 6 נבדקת נכונות התוכנית באמצעות מעקב "ידני" אחר מהלך הביצוע עבור מספר מצומצם של דוגמאות קלט; זאת על מנת לאמת ולקבל ביטחון ראשוני שהתוכנית אכן משיגה את מטרתה המיועדת.

בשלב 7 מורצת התוכנית המלאה במחשב. ההרצה מאפשרת בדיקה מלאה יותר של התוכנית, כיוון שניתן להריץ את התוכנית במהירות עבור דוגמאות קלט רבות ומגוונות.

בעיה 1

מטרת הבעיה הבאה ופתרונה: הדגמה מפורטת של פיתוח ויישום בשלבים של אלגוריתם.

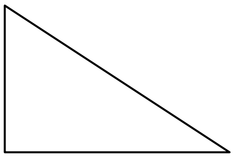
> טלי ואודי מעוניינים לבנות גלגלת שתחבר בין החלונות שלהם. הגלגלת מורכבת מחבל כפול באורך של המרחק שבין החלונות. על טלי למדוד את המרחק שבין החלונות כדי לדעת מה אורך החבל שיזדקקו לו. טלי בדקה ומצאה שכדי להגיע מהחלון שלה לחלון של אודי עליה לצעוד ישר מספר צעדים, לפנות ימינה ולצעוד מספר צעדים נוספים. טלי גם בדקה ומצאה שאורך כל צעד שלה הוא 42 ס"מ. (0.42 מ').

פתחו אלגוריתם שהקלט שלו הוא מספר הצעדים שעל טלי לצעוד ישר, ומספר הצעדים שעל טלי לצעוד לאחר שפנתה ימינה, והפלט שלו הוא אורך החבל המבוקש. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

ניתוח הבעיה בעזרת דוגמאות

נבדוק עבור מספר דוגמאות מה הפלט הרצוי עבור הקלט:

אם טלי צועדת 30 צעדים ישר, ואחר כך 21 צעדים ימינה, ניתן לתאר זאת באמצעות משולש ישר זווית, שניצב אחד שלו הוא 30 צעדים, והשני 21 צעדים. החלון של טלי הוא הקדקוד התחתון הימני של המשולש, והחלון של אודי הוא הקדקוד העליון של המשולש.



המרחק המבוקש הוא בדיוק אורך היתר במשולש זה. לכן, נשתמש במשפט פיתגורס, האומר כי אורך יתר במשולש ישר זווית הוא שורש סכום ריבועי שני הניצבים, כלומר: $c = \sqrt{a^2 + b^2}$

לכן, אם טלי צועדת 30 צעדים ישר, ואחר כך 21 צעדים ימינה, ניתן לחשב את אורך החבל המבוקש באופן הבא: $\sqrt{(30 \cdot 0.42)^2 + (21 \cdot 0.42)^2} = 15.38$.

לבסוף נכפיל את התוצאה ב-2 (כיוון שהחבל הינו כפול) והערך המתקבל הוא הפלט הנדרש.

ומה אם הנתון השני בקלט הוא 0? כלומר, טלי אינה צריכה לצעוד כלל ימינה, משום שהחלון של אודי נמצא בדיוק מול החלון שלה? במקרה זה, ברור כי המרחק המבוקש ניתן לחישוב ישירות לפי מספר הצעדים הנתון בקלט, אבל מסתבר שהנוסחה הכללית שמצאנו תתאים גם למקרה הפרטי הזה:

$$2 \cdot \sqrt{(30 \cdot 0.42)^2 + (0 \cdot 0.42)^2} = 2 \cdot \sqrt{(30 \cdot 0.42)^2} = 2 \cdot 30 \cdot 0.42$$

תהליך בדיקת הפלט עבור דוגמאות שונות עוזר לנו להבין את מהות הבעיה, להבין מה נדרש מאתנו, לחשוב על התהליכים הדרושים לפתרון הבעיה, ובכך מכוון אותנו לקראת השלבים הבאים של תכנון ושל כתיבת האלגוריתם עצמו.

פירוק הבעיה לתת-משימות

נפרק את הבעיה לשלוש תת-משימות:

1. קליטת שני מספרים חיוביים שלמים
2. חישוב אורך החבל הנדרש
3. הצגה של אורך החבל כפלט

התת-משימה הראשונה והשלישית הן פשוטות. התת-משימה השנייה היא התת-משימה העיקרית והמורכבת יותר. ניתן לפרק גם אותה לתת-משימות ובכך לפרט יותר את הרעיון לפתרון הבעיה:

2.1. הכפלת המספר הראשון ב-0.42, והעלאת התוצאה בריבוע
2.2. הכפלת המספר השני ב-0.42, והעלאת התוצאה בריבוע
2.3. חיבור שני הערכים שהתקבלו והוצאת שורש מהסכום
2.4. הכפלת הערך שהתקבל ב-2

כל הפעולות המפורטות כאן הן פעולות חישוב פשוטות לביצוע, שאינן דורשות ניתוח נוסף.

כעת, לאחר פירוק הבעיה לתת-משימות, ברור לנו הרעיון לפתרון הבעיה.

בחירת משתנים

שלושת המשתנים הראשונים שנבחר דרושים עבור פעולות הקלט והפלט:

stepForward – שלם, ישמור את מספר הצעדים קדימה

stepRight – שלם, ישמור את מספר הצעדים ימינה

ropeLength – ממשי, ישמור את אורך החבל הנדרש הסופי

נשתמש בעוד שני משתנים לשמירת ערכי הביניים של החישובים השונים:

side1 – ממשי, ישמור את ריבוע האורך של ניצב אחד במשולש

side2 – ממשי, ישמור את ריבוע האורך של ניצב שני במשולש

האלגוריתם

לפי החלוקה לתת-משימות ותוך שימוש במשתנים שבחרנו, נקבל את האלגוריתם הבא:

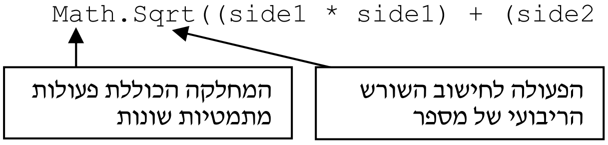
1. *קלוט מספר שלם חיובי ב-*stepForward
2. *קלוט מספר שלם חיובי ב-*stepRight
3. *חשב את מכפלת* stepForward *ב-0.42 והשם ב-*side1
4. *חשב את מכפלת* stepRight *ב-0.42 והשם ב-*side2
5. *העלה את* side1 *בריבוע*
6. *העלה את* side2 *בריבוע*
7. *חשב את סכום הערכים שהתקבלו*
8. *חשב את השורש הריבועי של הערך שהתקבל*
9. *הכפל את הערך שהתקבל ב-2 והשם ב-*ropeLength
10. *הצג כפלט את ערכו של* ropeLength

יישום האלגוריתם

הוראה 8 באלגוריתם כוללת חישוב שורש ריבועי. כיצד מבצעים זאת בשפת C#?

את הוראות 5 עד 8 ניתן לבטא בביטוי:

```
Math.Sqrt((side1 * side1) + (side2 * side2));
```



הפעולה `Sqrt` מקבלת בתוך הסוגריים מספר מטיפוס ממשי, ומחזירה מספר ממשי השווה לשורש הריבועי של המספר שקיבלה.

פעולה זו שייכת למחלקה `Math`. מחלקה זו נמצאת במרחב השמות System ולכן ההכרזה על השימוש ב-System מאפשר להשתמש בה ישירות.

את הערך 0.42 נגדיר כקבוע, כדי ליצור תוכנית קריאה ועמידה בפני שינויים.

הנה היישום של הוראות האלגוריתם:

```
Console.Write("Enter number of steps forward: ");
stepForward = int.Parse(Console.ReadLine());
Console.Write("Enter number of steps to the right: ");
stepRight = int.Parse(Console.ReadLine());
side1 = stepForward * STEP_SIZE;
side2 = stepRight * STEP_SIZE;
ropeLength = 2 * Math.Sqrt((side1 * side1) + (side2 * side2));
```

```
8. Console.WriteLine("The length of the rope is: {0}", ropeLength);
```

מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 21 30:

שורה לביצוע	stepForward	stepRight	side1	side2	ropeLength	**פלט**
1	?	?	?	?	?	Enter number of steps forward:
2	30	?	?	?	?	
3	30	?	?	?	?	Enter number of steps to the right:
4	30	21	?	?	?	
5	30	21	12.6	?	?	
6	30	21	12.6	8.82	?	
7	30	21	12.6	8.82	30.76	
8	30	21	12.6	8.82	30.76	The length of the rope is: 30.76

על פי המעקב הזה התוכנית מבצעת את מטרתה.

התוכנית המלאה

```
/* קלט: מספר הצעדים בין הבתים של טלי ושל אודי
פלט: אורך החבל הנדרש */
using System;
public class Rope
{
    public static void Main()
    {
        // הגדרת משתנים
        int stepForward, stepRight;  // המספרים הניתנים כקלט
        double side1, side2;         // ערכי הניצבים של המשולש
        double ropeLength;           // אורך החבל
        const double STEP_SIZE = 0.42;     // קבוע - גודל צעד
        // קלט
        Console.Write("Enter number of steps forward: ");
        stepForward = int.Parse(Console.ReadLine());
        Console.Write("Enter number of steps to the right: ");
        stepRight = int.Parse(Console.ReadLine());
        // חישוב אורך החבל
        side1 = stepForward * STEP_SIZE;
        side2 = stepRight * STEP_SIZE;
        ropeLength = 2 * Math.Sqrt((side1 * side1) +(side2 * side2));
        // פלט
        Console.WriteLine("The rope length is: {0}", ropeLength);
    } // Main
} // class Rope
```

סוף פתרון בעיה 1

פתרון הבעיות הבאות בפרק, בעיות מורכבות יותר מבעיה 1, יסתמך על התהליך שהצגנו. מעתה ואילך נניח שכאשר נדרשים פיתוח אלגוריתם ויישומו בשפת תכנות, נדרש פיתוח בשלבים, ולכן לא נציין זאת במפורש.

שאלה 4.1

דן, בן וחן קיבלו כל אחד דמי חנוכה מהוריהם. כל אחד קיבל סכום הגדול מ-20 ₪. שלושת החברים החליטו לאחד את כל סכום הכסף לקופה אחת ולקנות יחדיו כדורסל שמחירו 50 ₪. ביתרת הכסף יקנו מסטיקים שעלותם 1 ₪ כל אחד. פתחו אלגוריתם המקבל כקלט את דמי החנוכה שקיבל כל אחד מהחברים, ומציג כפלט את כמות המסטיקים שאפשר לקנות. ישמו את האלגוריתם בתוכנית בשפת C#.

המחלקה המתמטית

בפתרון בעיה 1 חישבנו שורש ריבועי על ידי שימוש בפעולה `Math.Sqrt`.

זהו שמה המלא של הפעולה `Sqrt` השייכת למחלקה המתמטית `Math`. כאמור, לפעולה אנו מעבירים ערך (פרמטר) שהוא מטיפוס ממשי, והיא מחזירה את השורש הריבועי של המספר שניתן לה. טיפוס הערך המוחזר הוא ממשי.

המחלקה המתמטית `Math` מכילה עוד פעולות מתמטיות רבות אשר נתונות לשימושנו, וכמוה יש מחלקות נוספות שנוכל להשתמש בהן לפעולות עזר מסוגים שונים.

פעולות שימושיות מהמחלקה המתמטית Math

הפעולה	תיאור הפעולה	טיפוסי פרמטרים	טיפוס ערך מוחזר	דוגמה – הפעולה	דוגמה – הערך המוחזר
Abs(num)	ערך מוחלט	שלם	שלם	Math.Abs(63)	63
		ממשי	ממשי	Math.Abs(-12.7)	12.7
Sqrt(num)	שורש ריבועי	ממשי	ממשי	Math.Sqrt(6.25)	2.5
Pow(num1,num2)	חזקה $num1^{num2}$	ממשי,ממשי	ממשי	Math.Pow(3,2)	9.0
Min(num1,num2)	הקטן מבין השניים	שלם, שלם	שלם	Math.Min(3,8)	3
		ממשי,ממשי	ממשי	Math.Min(8.0,8.8)	8.0
Max(num1,num2)	הגדול מבין השניים	שלם, שלם	שלם	Math.Max(3,8)	8
		ממשי,ממשי	ממשי	Math.Max(8.0,8.8)	8.8
Round(num)	עיגול מספר ממשי	ממשי	שלם	Math.Round(7.9)	8

בנוסף לפעולות המתוארות בטבלה זו שייכות למחלקה `Math` פעולות רבות נוספות. אתם מוזמנים לחפש ברשת, או בדפי ההדרכה של מיקרוסופט (MSDN), ולעיין בממשק של המחלקה המתמטית בשפת C# (`Math`) ולמצוא פעולות נוספות השייכות למחלקה זו.

עד כה הכרנו כמה מחלקות שימושיות: המחלקה `Console` האחראית לפעולות הקלט והפלט; המחלקה `Math` האחראית לפעולות מתמטיות. בהמשך לימודיכם תפגשו עוד מחלקות רבות המשמשות לצרכים שונים.

שאלה 4.2

פתחו אלגוריתם אשר הקלט שלו הוא גובהי שני תלמידים, נתונים במספרים ממשיים, והפלט שלו הוא הערך המוחלט של הפרשי הגבהים שלהם. ישמו את האלגוריתם בתוכנית בשפת C#. שימו לב לבחירת דוגמאות קלט מגוונות.

שאלה 4.3

שנו את התוכנית שכתבתם כפתרון לשאלה 4.2 כך שפלט התוכנית יהיה גובה התלמיד הנמוך מבין השניים.

שאלה 4.4

פתחו אלגוריתם אשר הקלט שלו הוא אורך ורוחב צלעות של מלבן (מספרים שלמים) והפלט שלו הוא שטח המלבן ואורך אלכסון המלבן.

4.2 פעולות חלוקה בשלמים

בפרק 3 אמרנו כי הגדרת טיפוס כוללת גם את פירוט הפעולות הניתנות לביצוע על ערכי הטיפוס. בפרק 3 הצגנו את הטיפוסים שלם וממשי, ורשימת הפעולות שהצגנו עבור כל אחד מהם כללה את הפעולות החשבוניות המוכרות. בסעיף זה נכיר פעולות חשבוניות המוגדרות רק עבור ערכים מטיפוס שלם.

בעיה 2

מטרת הבעיה הבאה ופתרונה: שימוש בפעולות לחישוב מנה ושארית בחלוקת מספרים שלמים.

ליונתן אוסף מטופח של גולות. הוא שומר את כל האוסף בקופסה, בקבוצות של 20, כלומר, כל קבוצה של 20 גולות ארוזה בשקית נפרדת. הגולות הנותרות מפוזרות בתחתית הקופסה. פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הגולות שיש ליונתן, והפלט הוא מספר השקיות שיש בקופסה ומספר הגולות המפוזרות בתחתיתה. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

למשל, עבור הקלט 135, הפלט הדרוש הוא: 15 6 כלומר, 6 שקיות ו-15 גולות פזורות.
(משום ש-135=15+20*6).

ניתוח הבעיה בעזרת דוגמאות

נבדוק את הפלט הרצוי עבור כמה דוגמאות קלט:

קלט	פלט
13	0 שקיות ו-13 פזורות
60	3 שקיות ו-0 פזורות
64	3 שקיות ו-4 פזורות
82	השלימו: _____ שקיות ו-_____ פזורות

על פי הדוגמאות אנו רואים כי מספר השקיות הוא מספר הפעמים שנכנס המספר 20 במספר הגולות הכולל, שנקלט מהקלט. הגולות הפזורות הן השארית, אלה שנותרו אחרי שאספנו קבוצות של 20 גולות ככל שיכולנו.

כלומר עלינו לבצע כאן שתי פעולות, כל אחת מהן מתבצעת על מספרים שלמים ומחזירה מספר שלם. הפעולה הראשונה מקבלת מספר x (במקרה שלנו, המספר הכולל של הגולות) ומספר y (במקרה שלנו, מספר הגולות בכל שקית, כלומר 20) ומחזירה את מספר הפעמים שהמספר y נכנס במספר x. הפעולה השנייה משלימה אותה, במובן מסוים: היא מקבלת x ו-y (כמו קודם) ומחזירה את מה שנותר אחרי שמחסירים מ-x כפולות שלמות של y ככל שניתן.

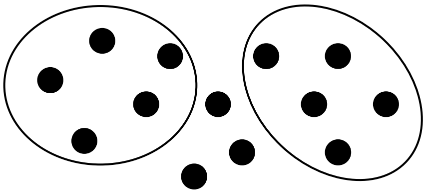
אלו הן פעולות המוגדרות על מספרים שלמים, כלומר על ערכים מטיפוס שלם: הפעולה הראשונה מחשבת את ה**מנה** של x חֶלְקֵי y. הפעולה השנייה היא פעולת ה**שארית (modulus)**.

בעברית מנה היא תוצאה של פעולת חלוקה. קיים קשר בין הפעולות האלו לפעולת החלוקה הרגילה. כאמור, אם נבצע חלוקה רגילה של מספר הגולות x במספר 20, לא נקבל בהכרח מספר שלם. למשל, אם יש 105 גולות, 105/20=5.25. אבל אם ננסה למלא מתוך 105 גולות כמה שיותר שקיות של 20 גולות, נצליח למלא בדיוק 5 שקיות. זהו בדיוק החלק השלם של תוצאת החלוקה 105/20.

שתי פעולות אלו נקראות פעולות חלוקה בשלמים:

פעולות החלוקה בשלמים מוגדרות רק על מספרים שלמים, וגם תוצאת הפעלתן היא מספר שלם: **מנת החלוקה** של מספר שלם x במספר שלם y שווה לחלק השלם של x/y, ובמילים אחרות, שווה למספר הפעמים שהערך y נכנס בערך x. **שארית החלוקה** (modulus) של מספר שלם x במספר שלם y מבטאת את השארית הנותרת לאחר חלוקה בשלמים של x ב-y, ובמילים אחרות, מבטאת את מה שנותר אחרי שמפחיתים מ-x כפולות שלמות של y ככל שניתן. השוני בין פעולות חלוקה בשלמים ובין פעולת החלוקה במספרים ממשיים מדגים את האבחנה בין הטיפוסים שלם לממשי.

למשל מנת החלוקה של 13 ב-5 שווה ל-2, משום ש-5 נכנס ב-13 פעמיים (ואכן 13/5=2.6 והחלק השלם הוא 2). לעומת זאת, שארית החלוקה של 13 ב-5 שווה ל-3, משום שאחרי שנפחית מ-13 שתי כפולות של 5 (כלומר 10), נקבל 3. נדגים זאת בעזרת האיור הבא, המציג חלוקת 13 נקודות לקבוצות של 5:



בחלוקה של 13 לקבוצות של 5 אנו אכן מקבלים כי מנת החלוקה היא 2, כיוון שהתקבלו שתי קבוצות שלמות בגודל 5 כל אחת. שארית החלוקה היא 3, כיוון שנשארו שלוש נקודות שלא שייכות לאף קבוצה.

את בעיית החלוקה הזו ניתן לראות באופן נוסף: אם נרצה לחלק 13 סוכריות באופן שווה ל-5 ילדים, כמה סוכריות יקבל כל ילד? כמה יישארו לנו? והתשובה תהיה זהה – כל ילד יקבל 2 סוכריות, ולנו יישארו 3.

פירוק הבעיה לתת-משימות

נפרק את הבעיה לארבע תת-משימות:

1. קליטת מספר הגולות
2. חישוב מספר השקיות
3. חישוב מספר הגולות הפזורות
4. הצגה של מספר השקיות ושל הגולות הפזורות כפלט.

הערכים בעיבוד הנתונים הם מספרים שלמים. אין לנו כאן עניין במספרים ממשיים, ולכן הפעולה שנרצה להשתמש בה לביצוע התת-משימה השנייה והשלישית איננה פעולת חלוקה רגילה, כיוון שפעולה כזאת תיתן תוצאה מטיפוס ממשי. למשל תוצאה של 64/20 תיתן 3.2, אך מספר זה אינו נותן לנו את מספר השקיות המלאות ואת מספר הגולות הפזורות. הפעולות המתאימות הן פעולות החלוקה בשלמים שהוצגו לעיל.

את מספר השקיות המלאות נחשב כמנה של חלוקת מספר הגולות ב-20; את מספר הגולות הפזורות נחשב כשארית של אותה חלוקה. שתי פעולות אלה הן שתי פנים של תבנית שימושית: חלוקת כמות פריטים לקבוצות בגודל נתון.

בחירת משתנים

amount – שלם, ישמור את המספר הניתן כקלט, מספר הגולות.
bags – שלם, ישמור את מספר השקיות המכילות 20 גולות כל אחת.
remainder – שלם, ישמור את מספר הגולות שנשארו פזורות.

האלגוריתם

1. *קלוט את מספר הגולות ב-*amount
2. *חשב את מספר השקיות המלאות בחישוב מנת החלוקה של* amount *ב-20 והשם את התוצאה ב-*bags
3. *חשב את מספר הגולות שנותרו פזורות בחישוב שארית החלוקה של* amount *ב-20 והשם את התוצאה ב-*remainder
4. *הצג כפלט את הערך* bags *ואת הערך* remainder

יישום האלגוריתם

> בשפת C#, הפעולה לחישוב שארית החלוקה של הערך a בערך b נכתבת בצורה `a % b`, כאשר גם a וגם b הם מטיפוס שלם.

למשל: 7%3=1, 3%5=3, ו-12%4=0.

> בשפת C#, הפעולה לחישוב מנת החלוקה של הערך a בערך b נכתבת בצורה `a / b`, בתנאי שגם a וגם b הם מטיפוס שלם.

למשל: 7/3=2, 13/5=2 ו-12/4=3.

שימו לב שאותו סימן משמש ב-C# הן לפעולת חלוקה במספרים ממשיים והן לפעולת חישוב של מנת החלוקה בשלמים. אם כך, איך נבדיל בין פעולת חישוב של מנת החלוקה בשלמים לבין פעולת חלוקה בממשיים? איך נדע אם הביטוי 13/5 ערכו שווה ל-2 או ל-2.6?

סוג החלוקה נקבע על פי חוק פשוט:

> חלוקת שלם בשלם תתפרש **תמיד** כפעולת חלוקה בשלמים.
> כל אחת מהחלוקות הבאות מתפרשת כפעולת חלוקה של מספרים ממשיים:
> שלם/ממשי
> ממשי/שלם
> ממשי/ממשי
> למשל 13/5=2, 13.0/5=2.6 ו-13/5.0=2.6.

ניישם את האלגוריתם בשימוש בפעולות החלוקה בשלמים בשפה, ובהגדרת קבוע (PER_BAG) עבור מספר הגולות בשקית:

```
1.  Console.Write("Enter amount of marbles: ");
2.  amount = int.Parse(Console.ReadLine());
3.  bags = amount / PER_BAG;
4.  remainder = amount % PER_BAG;
5.  Console.WriteLine("There are {0} bags, {1} are left over", bags,
    remainder);
```

מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 92:

שורה לביצוע	number	bags	remainder	**פלט**
1. Console.Write(...)	?	?	?	Enter amount of marbles:
2. amount = **int**.Parse...	92	?	?	
3. bags = amount / PER_BAG	92	4	?	
4. remainder = amount % PER_BAG	92	4	12	
5. Console.WriteLine(...)	92	4	12	There are 4 bags, 12 are left over

התוכנית המלאה

```
/*
התוכנית קולטת את מספר הגולות שיש ליונתן
ונותנת כפלט את מספר השקיות המלאות, ואת מספר הגולות שנותרו פזורות
*/
using System;
public class MarblesInBox
{
    public static void Main()
    {
        // הגדרת משתנים
        int amount;                  // כמות הגולות
        int bags;                    // מספר שקיות
        int remainder;               // מספר גולות פזורות
```

```
        const int PER_BAG = 20;      // מספר הגולות בשקית אחת
        // קלט
        Console.Write("Enter amount of marbles: ");
        amount = int.Parse(Console.ReadLine());
        // חישוב מספר השקיות
        bags = amount / PER_BAG;
        // חישוב מספר הגולות הפזורות
        remainder = amount % PER_BAG;
        // פלט
        Console.WriteLine("There are {0} bags, {1} are left over",
                                                     bags, remainder);
    } // Main
} // class MarblesInBox
```

סוף פתרון בעיה 2

שאלה 4.5

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `MarblesInBox` עבור הקלט 123.

שאלה 4.6

יונתן בעל הגולות החליט לשדרג את אופן האחסון של האוסף ושינה את פקודות התוכנית שבפתרון בעיה 2 כך שגם מספר הגולות בשקית יהיה חלק מהקלט, ולא קבוע:

```
Console.Write("Enter amount of marbles: ");
amount = int.Parse(Console.ReadLine());
Console.Write("Enter capacity": );
capacity = int.Parse(Console.ReadLine());
bags = amount / capacity;
remainder = amount % capacity;
Console.WriteLine("There are {0} bags, {1} are left over",
                                                     bags, remainder);
```

מה יהיה פלט הפתרון המשודרג לבעיית אחסון הגולות של יונתן עבור הקלט 30 125? מה יהיה הפלט עבור הקלט 50 200?

שאלה 4.7

השלימו את תוצאות הפעולות הבאות של חלוקה בשלמים:

א. 7/3 = __________

ב. 7%3 = __________

ג. 20/4 = __________

ד. 20%4 = __________

ה. 3/7 = __________

ו. 3%7 = __________

שאלה 4.8

נתון קטע התוכנית הבא, שהקלט שלו הוא שני מספרים שלמים, והמשתנים בו הם מטיפוס שלם:

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
a = num1 / num2;
```

```
b = num1 % num2;
Console.WriteLine("a: {0} b: {1}", a, b);
```

א. רשמו את פלט קטע התוכנית עבור כל אחד מהקלטים הבאים (משמאל לימין):

1. 5 2 ________________
2. 5 4 ________________
3. 5 5 ________________

ב. תנו דוגמה לקלט עבורו הפלט יהיה `a: 2 b: 2` ________________

להעמקה בתבנית ***חלוקת כמות פריטים לקבוצות בגודל נתון*** פנו לסעיף התבניות המופיע בסוף הפרק.

עוד על פעולת השארית

? בשאלה 4.7, בסעיף ד, התקבל הערך 0. מה משמעות הביטוי 20%4=0? באופן כללי, מתי פעולת שארית נותנת תוצאה השווה ל-0?

התוצאה 0 מצביעה על כך שהמספר הראשון (במקרה זה 20) מתחלק במספר השני (במקרה זה 4) **ללא שארית**. כל ומר, המספר 4 הוא אחד המחלקים של 20.

אם כך כיצד נבדוק אם מספר נתון הוא זוגי? מספר זוגי הוא מספר המתחלק ב-2 ללא שארית. לכן, נוכל לחשב את שארית החלוקה של המספר הנתון ב-2. אם המספר אכן זוגי השארית שנקבל תהיה שווה ל-0.

שימו ♥: שארית החלוקה של ערך x בערך y **אינה** שווה לחלק הלא שלם המתקבל מפעולת החלוקה בממשיים של x ב-y. למשל, אם נחשב 13/5 בפעולת חלוקה ממשית נקבל 2.6. לעומת זאת, שארית החלוקה 13%5 היא 3, ולא 6.

? אם נחלק מספר ב-2 ונבדוק את השארית. אילו מספרים יכולים להתקבל כתוצאה?

נבדוק את המקרים הבאים:

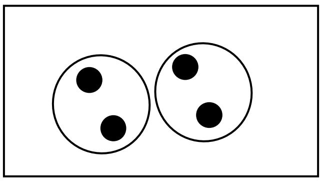
4 % 2 = 0

5 % 2 = 1

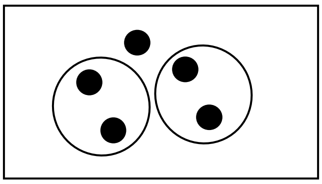
6 % 2 = 0

7 % 2 = 1

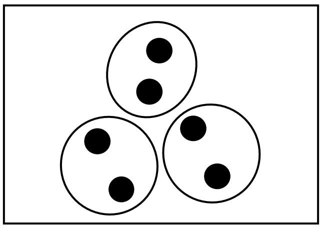
המספרים האפשריים כתוצאה הם אך ורק 0 ו-1. לא ייתכן כי נקבל תוצאה הגדולה מ-1. נראה זאת שוב בעזרת איור: כאשר אנו מחלקים מספר כלשהו של נקודות לקבוצות של שתי נקודות כל אחת, ובודקים כמה נקודות לא נכללות באף קבוצה, ייתכן כי לא נשארת אף נקודה ללא קבוצה, למשל, עבור 4 % 2:



ייתכן כי נשארת נקודה בודדת ללא קבוצה, למשל, עבור 2 % 5:



לא ייתכן שיישארו שלוש נקודות מחוץ לקבוצות, משום שאז ניתן לקחת שתיים מהן וליצור קבוצה חדשה: אם נגדיל את מספר הנקודות ב-1 ונחשב את 2 % 6, נראה כי נוצרה קבוצה חדשה מהנקודה הבודדת שנותרה קודם ומהנקודה החדשה:



אם כך, השארית של חלוקת מספר כלשהו ב-2 לא יכולה להיות גדולה מ-1.
מה השארית של חלוקת מספר כלשהו ב-3? באותו אופן בדיוק ניתן להראות כי השארית של חלוקת מספר כלשהו ב-3 לא יכולה להיות גדולה מ-2.
ובאופן כללי:

הערכים האפשריים לשארית של חלוקת מספר כלשהו ב-n הם בין 0 ל-(n-1).

שאלה 4.9

השלימו:

א. הערכים האפשריים לשארית החלוקה של מספר כלשהו ב-5 הם ____________________ .

ב. הערכים האפשריים לשארית החלוקה של מספר כלשהו ב-6 הם ____________________ .

שאלה 4.10

נתון קטע התוכנית הבא אשר הקלט שלו הוא פרק זמן הנתון בשעות, השמור במשתנה `hours`, והפלט שלו הוא מספר היממות השלמות והשעות הנותרות בפרק הזמן הנתון.
השלימו את קטע התוכנית.

```
// קלט
hours = int.Parse(Console.ReadLine());
days = ________________;
hoursLeft = ______________;
// פלט
Console.WriteLine("There are {0} days and {1} hours", days,
                                                        hoursLeft);
```

שאלה 4.11

פתחו אלגוריתם שהקלט שלו הוא פרק זמן הנתון בדקות, והפלט שלו הוא מרכיבי השעות והדקות בפרק זמן זה. ישמו אותו בשפת התכנות C#.

למשל, עבור הקלט 90 הפלט יהיה: 1 שעות, 30 דקות.
עבור הקלט 30 הפלט יהיה: 0 שעות, 30 דקות.
עבור הקלט 300 הפלט יהיה: 5 שעות, 0 דקות.

שאלה 4.12

טייס חלל יוצא ביום ראשון לטיסה ממושכת בחלל. עבור כל יום טיסה, יש לצייד את הטייס ב-3.8 ליטרים של מים.

פתחו אלגוריתם שהקלט שלו הוא מספר ימי הטיסה בחלל, והפלט שלו הוא מספר השבתות שייעדר הטייס מביתו, וכמות המים שיש לציידו לקראת הטיסה. ישמו את האלגוריתם בשפת התכנות C#.

שאלה 4.13

פקיד חרוץ ממלא טופס במשך 10 דקות. עבור כל טופס שהפקיד ממלא הוא מקבל שכר של 6.3 ₪. הפקיד עובד ללא הפוגה.

פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הטפסים שעל הפקיד למלא, והפלט שלו הוא משך העבודה, בשעות ובדקות, והשכר שהפקיד יקבל. ישמו את האלגוריתם בשפת C#.

למשל עבור הקלט 55 יהיה הפלט: משך העבודה הוא: 9 שעות ו-10 דקות והשכר הוא: 346.5.

שאלה 4.14

בפס ייצור במפעל לייצור משאיות מרכיבים גלגלים במשאיות.

פתחו בשלבים אלגוריתם אשר הקלט שלו הוא מספר הגלגלים הכולל בפס הייצור ומספר גלגלים למשאית, והפלט שלו הוא מספר המשאיות שיורכבו להן גלגלים.

למשל עבור הקלט 6 1000, שפירושו 1000 גלגלים בסך הכול בפס הייצור, ו-6 גלגלים למשאית, יהיה הפלט 166.

המרת ערך שלם לממשי

ראובן החליט, בנדיבות לבו, כי את דמי החנוכה שהוא מקבל מסבתו, הוא יחלק באופן שווה בין החברים שיגיעו למסיבת החנוכה שלו.

למשל, אם יקבל ראובן מסבתו 25 ₪, ולמסיבה יגיעו 10 חברים, יקבל כל אחד מהחברים 2.5 ₪.

ראובן כתב את ההוראות הבאות כדי לחשב את הפלט הנדרש:

```
   int money, friends;
   double gift;
1. Console.Write("Enter sum of money: ");
2. money = int.Parse(Console.ReadLine());
3. Console.Write("Enter number of friends: ");
4. friends = int.Parse(Console.ReadLine());
5. gift = money / friends;
6. Console.WriteLine("Every friend will get: {0} shekels", gift);
```

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט 25 10:

שורה לביצוע	money	friends	gift	פלט
1. Console.Write(...)	?	?	?	Enter sum of money:
2. money = **int**.Parse(..)…	25	?	?	
3. Console.Write(...)	25	?	?	Enter number of friends:
4. money = **int**.Parse(..)…	25	10	?	
5. gift = money / friends	25	10	2	
6. Console.WriteLine(...)	25	10	2	Every friend will get: 2 shekels

מדוע הפלט שגוי? ניזכר בהגדרת פעולת החילוק (/) בשפת C#: כאשר מחלקים שלם בשלם, פעולת החילוק מתפרשת **תמיד** כחלוקה בשלמים.

נתבונן בשורה השלישית: בביטוי `money / friends` מתבצעת חלוקה של שני ערכים מטיפוס שלם. אמנם תוצאת החלוקה מושמת במשתנה מטיפוס ממשי, אך דבר זה נעשה רק לאחר ביצוע פעולת החלוקה, וזו הייתה פעולת חלוקה בשלמים ולא פעולת חלוקה בממשיים, כפי שנדרש. עבור הקלט 10 25, מחושבת מנת החלוקה של 25 ב-10, והתוצאה המתקבלת היא שלמה (2). תוצאה זו מוצבת במשתנה `gift` המכיל כעת את הערך הממשי 2.0 (הצורה הממשית של המספר 2). אם כך, התשובה שקיבלנו היא שגויה, מכיוון שמחֲלֻקַּת 25 ב-10 ציפינו במקרה זה לקבל 2.5 ולא 2.0.

כדי לפתור את הבעיה יש להודיע כי **אף על פי** שהמשתנה `money` הוא משתנה מטיפוס שלם השומר ערך שלם, יש להתייחס זמנית אל ערכו כאל ערך ממשי. בכך החלוקה הופכת להיות חלוקה של ערך ממשי בערך שלם – לכן היא חלוקה בממשיים, כפי שנדרש. דבר זה מתבצע בשפת C# באמצעות פעולת **המרה** (casting).

לפני הביטוי שאת ערכו אנו מעוניינים לפרש זמנית כממשי ולא כשלם (במקרה זה, הביטוי `money`), נוסיף את הוראת ההמרה שמשמעותה: המר, רק לצורך החישוב הנוכחי, את ערכו של ביטוי זה לערך ממשי. ההוראה מיושמת בכתיבת שם הטיפוס שנרצה להמיר אליו, עטוף בסוגריים, לפני הביטוי המומר:

```
gift = (double)money / friends;
```

שימו ♥: המשתנים המעורבים בביטוי המומר (במקרה זה, המשתנה `money`) **אינם** הופכים מרגע זה למשתנים מטיפוס ממשי. הם נשארים משתנים מטיפוס שלם בדיוק כפי שהיו עד כה. ההמרה גורמת לכך שרק **בעת חישוב הביטוי**, ערכם נראה כממשי ולא כשלם.

בהתאם לכך, קטע הקוד הנכון הוא:

```
int money, friends;
double gift;
Console.Write("Enter sum of money: ");
money = int.Parse(Console.ReadLine());
Console.Write("Enter number of friends: ");
friends = int.Parse(Console.ReadLine());
gift = (double)money / friends;
Console.WriteLine("Every friend will get: {0} shekels", gift);
```

שאלה 4.15

לפניכם סדרת הוראות. השלימו בטבלה את ערכי המשתנים לאחר כל הוראה.

	num	fnum
`int num, x = 22, y = 5;`		
`double fnum;`		
`num = x / y;`		
`fnum = x / y;`		
`fnum = (double)x / y;`		
`num = x % y;`		

שאלה 4.16

אהרון מעוניין לדעת את ממוצע ציוניו במקצועות היסטוריה, תנ"ך וספרות.

פתחו אלגוריתם המקבל כקלט את שלושת הציונים במקצועות אלו (מספרים שלמים בין 0 ל-100), ומציג כפלט את הממוצע של שלושת הציונים. ישמו את האלגוריתם בשפת התכנות C#.

שאלה 4.17

דינה מעוניינת לדעת את הציון הכולל שלה במדעי המחשב.
ידוע כי שתי היחידות הראשונות מהוות 33% מהציון, היחידה השלישית מהווה 17% מהציון, ושתי היחידות האחרונות מהוות 50% מהציון הכללי.
פתחו אלגוריתם המקבל כקלט את שלושת ציוניה של דינה במדעי המחשב (בשתי היחידות הראשונות, ביחידה השלישית ובשתי היחידות האחרונות) כמספרים שלמים בין 0 ל-100, ומציג כפלט את ציונה הכולל. ישמו את האלגוריתם בשפת התכנות C#.

פירוק מספר דו-ספרתי לספרותיו

בסעיף זה נכיר שתי תבניות שימושיות. התבנית של פירוק מספר לספרותיו תודגם בבעיה 3. תבנית שמשלימה אותה במובן מסוים היא בניית מספר מִסְּפָרוֹת, ואליה נתייחס בשאלה 4.19.

בעיה 3

מטרת הבעיה הבאה ופתרונה: פיתוח ויישום בשלבים של אלגוריתם ושימוש בפעולות חלוקה בשלמים לצורך פירוק מספר דו-ספרתי לספרותיו.

> פתחו אלגוריתם שהקלט שלו הוא מספר דו-ספרתי, חיובי או שלילי, והפלט שלו הוא סכום ספרות המספר. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

ניתוח הבעיה בעזרת דוגמאות

נבדוק עבור מספר דוגמאות מה הפלט הרצוי עבור הקלט:

קלט	פלט
71	8
-53	8
49	13

גם במקרה זה, תהליך בדיקת הפלט עבור דוגמאות שונות מסייע לנו בהבנת משמעות הבעיה ובזיהוי התהליכים הדרושים לפתרונה. למשל עבור הקלט 71 והקלט 53 קיבלנו את אותו הפלט. כעת אנו רואים כי ייתכנו קלטים שונים אשר מתאים להם אותו פלט. לקלט -53 ולקלט 53 מתאים אותו פלט, וכך אנו רואים כי אין חשיבות לסימנו של המספר שבקלט. אלו הן אבחנות חשובות לצורך כתיבת אלגוריתם נכון לפתרון הבעיה.

פירוק הבעיה לתת-משימות

את הבעיה הזו נפרק לשלוש תת-משימות:

1. קליטת מספר דו-ספרתי חיובי או שלילי
2. פירוק המספר לספרותיו וסיכום הספרות
3. הצגה של סכום הספרות כפלט

ניתן לפרק גם את התת-משימה השנייה לתת-משימות ובכך לפרט יותר את הרעיון לפתרון הבעיה:

2.1. פירוק המספר לספרותיו
2.2. סיכום הספרות

כיצד נפרק את המספר לספרותיו?

ידוע לנו כי המספר הוא דו-ספרתי, ולכן פירוקו לספרותיו פירושו חישוב ספרת העשרות וספרת האחדות. נפרק אם כן את תת-משימה 2.1, פירוק המספר לספרותיו, כך:

2.1.1. חישוב ספרת העשרות של המספר

2.1.2. חישוב ספרת האחדות של המספר

? כיצד נבצע פעולות אלו?

נוכל להיעזר בפעולות חלוקה בשלמים, של המספר הדו-ספרתי הנתון במספר 10:

שארית החלוקה של מספר שלם **כלשהו** ב-10 נותנת תמיד את **ספרת האחדות** של המספר. מנת החלוקה של מספר שלם **דו-ספרתי** חיובי ב-10 נותנת תמיד את **ספרת העשרות** של המספר.

למשל 24%10=4 ו-24/10=2.

שימו ♥: חישוב מנת החלוקה של מספר דו-ספרתי ב-10 נותנת את ספרת העשרות של המספר, אם הוא חיובי, אך לא אם הוא שלילי! למשל -20/10=-2, ולא 2!

לכן יש לטפל באופן שונה במספר שלילי.

בדוגמאות ראינו כי הפלט עבור מספר חיובי ועבור המספר הנגדי שלו צריך להיות זהה. לכן, כל שעלינו לעשות עבור מספר שלילי, הוא "להיפטר" מהסימן -, כלומר להפוך את המספר לחיובי, עוד לפני פירוקו לספרותיו.

? כיצד נהפוך מספר שלילי לחיובי?

נוכל לעשות זאת על ידי חישוב ערכו המוחלט של המספר לפני פירוקו לספרותיו.

אם כן, בסך הכול, תת-משימה 2.1, פירוק המספר לספרותיו תורכב משלוש תת-משימות:

2.1.1. חישוב ערכו המוחלט של המספר

2.1.2. חישוב ספרת העשרות של המספר

2.1.3. חישוב ספרת האחדות של המספר

כעת, לאחר פירוק הבעיה לתת-משימות ברור לנו הרעיון לפתרון הבעיה וקל לכתוב את האלגוריתם עצמו.

בחירת משתנים

שני המשתנים הראשונים שנבחר דרושים לנו עבור פעולות הקלט והפלט:

num – שלם, ישמור את המספר הניתן כקלט

sum – שלם, ישמור את סכום ספרות המספר

המשתנים לשמירת החישובים השונים:

absNum – שלם, ישמור את ערכו המוחלט של המספר

tens – שלם, ישמור את ספרת העשרות של המספר

units – שלם, ישמור את ספרת האחדות של המספר

האלגוריתם

האלגוריתם, לפי החלוקה לתת-משימות ובשימוש במשתנים שבחרנו, יהיה:

1. *קלוט מספר שלם דו-ספרתי ב-*num
2. *חשב את ערכו המוחלט של המספר והשם ב-*absNum
3. *חשב את מנת החלוקה של* absNum *ב-10 והשם ב-*tens
4. *חשב את שארית החלוקה של* absNum *ב-10 והשם ב-*units
5. *חשב את הסכום של* tens *ושל* units *והשם ב-*sum
6. *הצג כפלט את ערכו של* sum

יישום האלגוריתם

כזכור, לצורך חישוב ערכו המוחלט של מספר, נוכל להשתמש בפעולה לחישוב הערך המוחלט ששייכת למחלקה המתמטית, ומתוארת בטבלת הפעולות. המשפט המתאים ליישום הוראה 2 באלגוריתם הוא:

```
absNum = Math.Abs(num);
```

התוכנית המלאה

```
/*
קלט: מספר דו-ספרתי שלם חיובי או שלילי
פלט: סכום ספרות המספר
*/
using System;
public class DigitSum
{
    public static void Main()
    {
        // הגדרת משתנים
        int num;   // המספר הניתן כקלט
        int absNum;      // ערכו המוחלט של המספר
        int tens;        // ספרת העשרות
        int units;       // ספרת האחדות
        int sum;         // סכום הספרות
        // קלט
      1. Console.Write("Enter a two digit number: ");
      2. num = int.Parse(Console.ReadLine());
        // קבלת ערכו המוחלט של המספר
      3. absNum = Math.Abs(num);
        // פירוק המספר לספרותיו
      4. tens = absNum / 10;
      5. units = absNum % 10;
        // סכום ספרות המספר
      6. sum = tens + units;
        // פלט
      7. Console.WriteLine("The sum of the digits is: {0}", sum);
    }  // Main
}  // class DigitSum
```

מעקב

נעקוב אחר מהלך ביצוע משפטי התוכנית עבור הקלט ‎-17:

שורה לביצוע	num	absNum	tens	units	sum	פלט
1	?	?	?	?	?	Enter a two digit number:
2	-17	?	?	?	?	
3	-17	17	?	?	?	
4	-17	17	1	?	?	
5	-17	17	1	7	?	
6	-17	17	1	7	8	
7	-17	17	1	7	8	The sum of the digits is: 8

על פי המעקב הזה התוכנית ביצעה את מטרתה.

אך יש עוד לבדוק את התוכנית עבור **דוגמאות קלט מגוונות** (כלומר, בעלות מאפיינים שונים אשר מבטאים את מגוון הקלטים האפשריים) כדי להשתכנע שאכן היא מבצעת את מטרתה עבור כל קלט אפשרי.

בטבלה זו בדקנו דוגמה לקלט שלילי. כדאי לבדוק אם התוכנית מבצעת את מטרתה גם עבור קלט חיובי.

שאלה 4.18

בנו טבלת מעקב אחר ביצוע משפטי התוכנית `DigitSum` עבור הקלט 53.

סוף פתרון בעיה 3

להעמקה בתבנית ***פירוק מספר חיובי לספרותיו*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 4.19

שנו את התוכנית `DigitSum` (התוכנית מפתרון בעיה 3) כך שהקלט יהיה המספר שהתקבל והפלט יהיה המספר בסדר ספרות הפוך. למשל עבור הקלט 43 הפלט הנדרש הוא 34.

הדרכה: בידיכם שני משתנים המבטאים את ספרת האחדות (`units`) ואת ספרת העשרות (`tens`) של המספר. חשבו על ביטוי חשבוני אשר משתמש בשני ערכים אלה ונותן את המספר הנדרש.

להעמקה בתבנית ***בניית מספר*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 4.20

שנו את התוכנית `DigitSum` (התוכנית מפתרון בעיה 3) כך שהפלט יהיה **המרחק** בין ספרות המספר. למשל עבור הקלט 41, הפלט הנדרש הוא 3.

הדרכה: היעזרו בפעולה לחישוב ערך מוחלט.

4.3 הטיפוס התווי

עד עתה עסקנו רק במספרים מטיפוס שלם או מטיפוס ממשי. בפרק זה נלמד כי עיבודים ניתן לבצע לא רק על מספרים, אלא גם על תווים. תו הוא כל סימן אשר ניתן להציג על מסך המחשב.

כגון: אותיות ('a', 'b', 'A', ...), סימנים מתמטיים ('<', '+', '*', '/', '%', ...), ספרות ('0', '1', '2', ...), סימני תחביר ('!', ',', ':', ...), רווח (' '), וכן הלאה.

לצורך הטיפול הפנימי של המחשב בתווים מותאם לכל תו מספר שלם, על פי קוד סטנדרטי, השומר על העקרון: לתווים עוקבים מותאם ערך מספרי עוקב. למשל אם לתו 'b' מותאם מספר כלשהו, לתו 'c' מותאם המספר העוקב לו. למעשה, כל התווים מסודרים בתת-קבוצות המכילות תווים עוקבים: האותיות האנגליות הגדולות (A..Z) מסודרות בתת קבוצה אחת והאותיות הקטנות (a..z) מסודרות בתת קבוצה אחרת, אותיות עבריות בתת-קבוצה נפרדת, וכך גם הספרות.

בהמשך לימודינו נלמד כיצד ניתן לעבד מילים ואף משפטים שלמים, אך בסעיף זה נתמקד כאמור בתווים בודדים.

ערך מטיפוס תווי מצוין בשפת C# בין גרשיים בודדים. למשל 'Z'.

בעיה 4

מטרת הבעיה הבאה ופתרונה: הצגת הטיפוס התווי

פתחו אלגוריתם אשר הקלט שלו הוא אות אנגלית גדולה והפלט שלו הוא האות האנגלית הקטנה המתאימה. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

ניתוח הבעיה בעזרת דוגמאות

הדוגמאות הבאות מבהירות את הנדרש בבעיה 4:

קלט	פלט
'A'	'a'
'D'	'd'
'V'	'v'

פירוק הבעיה לתת-משימות

1. קליטת אות אנגלית גדולה
2. חישוב האות האנגלית הקטנה המתאימה
3. הצגת האות האנגלית הקטנה המתאימה כפלט

? כיצד נחשב את האות האנגלית הקטנה המתאימה?

אנו יודעים כי לכל תו מתאים מספר, ולתווים עוקבים מתאימים מספרים עוקבים.
הנה דוגמה אפשרית למספור כזה:

'A' – 65 'B' – 66 'C' – 67 'D' – 68 ...	'a' – 97 'b' – 98 'c' – 99 'd' – 100 ...

מכיוון שמספור האותיות הגדולות מתחילות ממספר מסוים, וממשיך משם בסדר עולה רציף, וכך גם לגבי האותיות הקטנות, הרי **ההפרש** בין המספר המייצג אות גדולה למספר המייצג אות קטנה

המתאימה לה הוא קבוע. כלומר, אם למשל, ההפרש בין 'a' ל-'A' הוא 32 (כפי שמתקיים בדוגמה שלעיל כי 32=97-65), זהו גם ההפרש בין 'b' ל-'B' (32=98-66), בין 'c' ל-'C', וכן הלאה.

אם כך, כל שיש לעשות הוא להוסיף למספר המייצג את האות האנגלית הגדולה שקלטנו את ההפרש הקבוע שבין אות קטנה והאות הגדולה המתאימה לה. את ההפרש הקבוע הזה נוכל לחשב עבור זוג אותיות כלשהו למשל עבור האותיות 'a' ו-'A'.

הנה הפירוק של תת-משימה 2:

2.1. חישוב ההפרש הקבוע בין אות קטנה לאות הגדולה המתאימה לה, על ידי הפחתת המספר המייצג את 'A' מהמספר המייצג את 'a'.
2.2. הוספת ההפרש שחושב למספר המייצג את האות הגדולה שנקלטה

בחירת משתנים

נשתמש במשתנים הבאים:

capitalLetter – מטיפוס תווי, ישמור את התו הניתן כקלט, כלומר את האות הגדולה

smallLetter – מטיפוס תווי, ישמור את תו הפלט, כלומר את האות הקטנה המתאימה

האלגוריתם

הנה האלגוריתם המתאים לפירוק לתת-משימות ולמשתנים שנבחרו:

1. *קלוט אות גדולה ב-*capitalLetter
2. *חשב את ההפרש בין המספר המייצג את* 'a' *למספר המייצג את* 'A' *הוסף את ההפרש שחושב ל-*capitalLetter *והשם את התוצאה ב-*smallLetter
3. *הצג כפלט את ערך* smallLetter

יישום האלגוריתם

האלגוריתם משתמש במשתנים מטיפוס תווי, ולכן צריך כמובן להצהיר עליהם.

משתנה מטיפוס תווי מוצהר באמצעות המילה **char** (קיצור של המילה character, ומשמעותה תו באנגלית).

למשל:

```
char capitalLetter;
```

יישום הוראה 1 באלגוריתם צריך להיות משפט קלט, הקולט תו. פעולת הקלט של ערכים מטיפוס תווים היא הפעולה `char.Parse`, והשימוש בה דומה לשימוש בפעולות קלט אחרות שראינו. לכן, היישום של הוראה 1 יהיה:

```
Console.Write("Enter a capital letter: ");
capitalLetter = char.Parse(Console.ReadLine());
```

כיצד נבצע את החישובים הנדרשים בהוראה 2?

בשפת C# תו מזוהה עם הערך המספרי המתאים לו. לכן ניתן לשלב תווים בביטויים חשבוניים, והפעולות שכלולות בביטוי יפעלו על הערכים המספריים המתאימים לתווים. לכן, הביטוי 'a'-'A' נותן לנו את ההפרש שאנו זקוקים לו. אם כך, הביטוי הבא מבטא את החישובים בהוראה 2:

```
capitalLetter +  'a' - 'A';
```

אבל מאחר שבשפת C# לכל תו יש ערך מספרי, הרי כאשר אנו מבצעים פעולות חשבוניות על ערכים מטיפוס תווי, התוצאה היא מטיפוס שלם. ליישום הוראה 2, נרצה להשים את ערך הביטוי בתוך משתנה מטיפוס תווי, `smallLetter`, ולכן עלינו להשתמש בהמרה (casting), הפעם כדי להמיר את ערך הביטוי מטיפוס שלם לטיפוס תווי.

נשתמש בפעולת ההמרה באותו האופן שהשתמשנו בה בסעיף הקודם. כאן נמיר ערך ביטוי מטיפוס שלם להיות ערך מטיפוס תווי. שימו לב שההמרה צריכה לחול על כל הביטוי, ולכן נעטוף אותו בסוגריים.
אם כך, היישום ב-C# של הוראה 2 הוא:

```
smallLetter = (char)(capitalLetter +  'a' - 'A');
```

שימו ♥: הוראת ההמרה לטיפוס תווי מבוטאת על ידי זוג הסוגריים השמאלי. זוג הסוגריים הימני מגדיר את התחום שמופעלת עליו ההמרה.

שימו ♥: בפתרון זה אנו מתייחסים הן למשתנים מטיפוס תווי (`smallLetter`, `capitalLetter`) והן לערכים תוויים, כלומר ערכים קבועים מטיפוס תווי ('a', 'A'). הערכים התוויים מוקפים בגרשיים. ההתייחסות למשתנים היא דרך שמותיהם, בדיוק כפי שנעשה עבור משתנים מטיפוס מספרי.

פלט של תווים נעשה ב-C# בעזרת הוראות הפלט המוכרות לנו.

נייֵשם את האלגוריתם תוך שימוש בהוראות קלט ופלט עבור תווים:

```
1. Console.Write("Enter capital letter: ");
2. capitalLetter = char.Parse(Console.ReadLine());
3. smallLetter = (char)(capitalLLetter +  'a' - 'A');
4. Console.WriteLine("The corresponding small letter is: {0}",
                                                    smallLetter);
```

שימו ♥: הוראות אלו אינן מסתמכות על ידיעת ערך ההפרש עצמו, או על ידיעה מפורשת של הערך המספרי המותאם לכל תו. בכך אנו מסייעים לתוכנית להיות עמידה יותר: גם אם גרסאות מאוחרות יותר של C# ישתמשו בקידוד אחר לתווים, הרי כל עוד יישמר העיקרון שתווים עוקבים מתאימים למספרים עוקבים, התוכנית תישאר נכונה.

התוכנית המלאה

```
/* התוכנית קולטת אות אנגלית גדולה
ומציגה כפלט את האות האנגלית הקטנה המתאימה לה */
using System;
public class CapitalLetterToSmall
{
    public static void Main()
    {
        // הגדרת משתנים
        char capitalLetter;    //  תו הקלט – אות אנגלית גדולה
        char smallLetter;      //  תו הפלט – אות אנגלית קטנה
        // קלט
        Console.Write("Enter a capital letter: ");
        capitalLetter = char.Parse(Console.ReadLine());
        // חישוב האות הקטנה
        smallLetter = (char)(capitalLetter + 'a' - 'A');
        // פלט
```

```
        Console.WriteLine("The corresponding small letter is: {0}",
        smallLetter);
    } // Main
} // class CapitalLetterToSmall
```

סוף פתרון בעיה 4

שאלה 4.21

שנו את התוכנית CapitalLetterToSmall כך שתקבל כקלט אות אנגלית **קטנה**, ותציג כפלט את האות האנגלית ה**גדולה** המתאימה לה.

רמז: שימו לב לסדר פעולות חשבון! בדקו עצמכם בזהירות!

שאלה 4.22

פתחו אלגוריתם הקולט תו ומציג כפלט את התו הבא אחריו בסדר הקידוד המספרי. ישמו את האלגוריתם כתוכנית בשפת C#.

שימו ♥: מה התו אחרי התו 'a'? מה התו אחרי התו 'S'? מה התו אחרי התו 'z'? בחנו דוגמאות קלט מגוונות ככל שניתן.

המרה מתו המייצג ספרה לערך מספרי מתאים

כפי שנאמר בתחילת הפרק, ערכי הטיפוס התווי כוללים כל סימן אשר ניתן להציג על מסך המחשב. בין סימנים אלה נכללות גם הספרות. למשל ערך מטיפוס תו יכול להיות '4' או '7'. לעתים, בהינתן תו המייצג ספרה, נרצה לדעת את ערכה המספרי של הספרה (כלומר להתאים לתו '4' את הערך השלם 4).

הנה דוגמה אפשרית למספור הפנימי עבור התווים המייצגים ספרות:

'0' – 48
'1' – 49
'2' – 50
'3' – 51
...

התווים המייצגים ספרות ('0', '1', '2' וכו') הם תווים עוקבים, ולכן גם המספור שלהם עוקב.

? כיצד ניתן לנצל את העובדה שמספור התווים המייצגים ספרות הוא עוקב כדי לחשב, עבור תו המייצג ספרה, את ערכה המספרי של הספרה?

נשים לב כי התו '1' מרוחק תו אחד מהתו '0'. בדומה, התו '2' מרוחק שני תווים מהתו '0', התו '3' מרוחק שלושה תווים מהתו '0', וכך הלאה. לכן נוכל להשתמש בפעולת חיסור כדי לחשב את הערך הדרוש. למשל אם נפחית מהמספר המתאים לתו '3' את המספר המתאים לתו '0', נקבל בדיוק 3 ('3'-'0' = 51-48 = 3).

גם הפעם אין אנו משתמשים באופן ישיר בערכי הקידוד עצמם, אלא רק מסתמכים על כך שהקידוד נותן ערכים עוקבים לספרות עוקבות.

לכן, אם במשתנה charDigit (מטיפוס תווי) שמור תו המייצג ספרה, אז לאחר החישוב שלהלן יכיל המשתנה digit, מטיפוס שלם, את ערכה המספרי של הספרה המיוצגת על ידי התו שב-charDigit.

```
digit = charDigit – '0';
```

שאלה 4.23

פתחו אלגוריתם המקבל כקלט תו המייצג ספרה.

(יש עשרה תווים אפשריים לקלט זה: '0', '1', '2' ... '9'). פלט האלגוריתם יהיה מורכב משלושה מספרים: הספרה המיוצגת על ידי תו הקלט (שנקרא לה לשם קיצור ספרת הקלט), ספרת הקלט לאחר הכפלתה ב-10, וספרת הקלט לאחר הכפלתה ב-15. ישמו את האלגוריתם כתוכנית בשפת C#.

הדרכה: חשבו מהו המספר המתקבל אחרי הכפלת ספרת הקלט ב-10.

שאלה 4.24

פתחו אלגוריתם המקבל כקלט שני תווים, כל אחד מהם מייצג ספרה. פלט האלגוריתם יהיה תוצאת החיבור בין שני המספרים המתאימים. למשל, אם הקלט הוא '5' '9' הפלט יהיה 14.

ישמו את האלגוריתם כתוכנית בשפת C#.

שימו ♥: כדי לקלוט מספר תווים מבלי להקיש Enter לאחר הקלדת כל תו, נוכל להשתמש בהוראת הקלט Read באופן הבא:

```
ch1 = (char)Console.Read();
ch2 = (char)Console.Read();
```

בשפת C# קיימות מחלקות מוכנות המיועדות לשימוש המתכנת. על אחת מהן למדנו בתחילת פרק זה, המחלקה Math ובה פעולות מתמטיות. בסעיף הבא נלמד על מחלקה נוספת בשם Random שיכולה לשמש אותנו ליצירת מספרים אקראיים.

4.4 בחירה אקראית

הגרלת מספרים אקראיים היא פעולה שימושית מאוד במדעי המחשב. למשל, בתחום בדיקות התוכנה, השימוש במספרים אקראיים מאפשר בדיקת תוכנה על אוסף קלטים גדול ואקראי שנוצר אוטומטית. גם בתחום ההצפנה מספרים אקראיים הם שימושיים מאוד. בסעיף זה נכיר מחלקה המוגדרת בשפת C# ומאפשרת עבודה נוחה עם מספרים אקראיים.

כיצד מגרילים מספר שלם?

כדי ליישם הוראת הגרלה בשפת C# נזדקק ל**מחלקת מספרים אקראיים:** `Random`.

בעזרת עצם מהמחלקה `Random` ניתן לבצע הגרלות של ערכים בכל תחום שנבחר. כדי שנוכל להשתמש בעצם מהמחלקה עלינו להצהיר עליו וליצור אותו באופן הבא:

```
Random rnd = new Random();
```

הוראה זו מצהירה על עצם מהמחלקה `Random`, בשם `rnd`, יוצרת אותו ומקצה לו מקום בזיכרון. כעת, כשיש בידינו עצם מהמחלקה `Random` אנו יכולים להגריל בעזרתו מספר שלם על ידי הפעלת הפעולה `Next` של העצם. השימוש בפעולה זו דומה לשימוש בפעולות של המחלקה `Math` שנלמדו בפרק זה. פעולה זו מקבלת בסוגריים פרמטר יחיד, שהוא ערך שלם המתאר את טווח המספרים להגרלה. אם נכתוב `Next(n)` יוגרל מספר שלם בתחום שבין 0 ל-(n-1). מאחר שזו פעולה של העצם, יש להשתמש בסימון הנקודה כדי להפעילה. למשל בעקבות ביצוע ההוראות:

```
int num;
num = rnd.Next(6);
```

יושם במשתנה `num` מספר אקראי כלשהו בין 0 ל-5.

לפעולה זו גרסה נוספת, המקבלת שני פרמטרים, ערכים שלמים, המתארים את גבולות ההגרלה: הפרמטר הראשון מבטא את הגבול התחתון של תחום ההגרלה והפרמטר השני מבטא את הגבול העליון, כך: ערך הביטוי `rnd.Next(n,m)` הוא מספר שלם אקראי בתחום שבין n ל- `(m-1)`.

שימו ♥: כאשר משתמשים בפעולות של המחלקה `Math` מפעילים אותן ישירות מהמחלקה `Math` (למשל: `Math.Sqrt(x)`), ואילו כאשר משתמשים בפעולה `Next` של `Random`, צריך ליצור תחילה עצם מסוג `Random` ועליו להפעיל את הפעולה `Next`. הבדל זה נובע מכך שהפעולות במחלקה `Math` מוגדרות כפעולות סטטיות השייכות למחלקה ולא לעצם. פרק 11 מרחיב בנושא זה.

בעיה 5

מטרת הבעיה הבאה ופתרונה: הצגת מחלקת מספרים אקראיים ואופן השימוש בה.

> פתחו אלגוריתם אשר ידמה הטלה של 2 קוביות, כלומר יגריל שני ערכים בתחום 1-6 ויציג אותם כפלט. ישמו את האלגוריתם בשפת התכנות C#.

בבעיה זו אנו מתבקשים לדמות הטלת שתי קוביות, כלומר להגריל שני ערכים בתחום 1-6.

פירוק הבעיה לתת-משימות

נפרק את הבעיה באופן הבא:

1. הגרלת 2 מספרים בין 1 ל-6.
2. הצגה של המספרים שהוגרלו כפלט.

בחירת משתנים

die1 – מספר שלם שערכו בין 1-6 ובו המספר הראשון שיוגרל.

die2 – מספר שלם שערכו בין 1-6 ובו המספר השני שיוגרל.

האלגוריתם

1. *הגרל מספר בין 1 ל-6 והשם ב-***die1**
2. *הגרל מספר בין 1 ל-6 והשם ב-***die2**
3. *הצג כפלט: "ערך שתי הקוביות הוא "* **die1, die2**

יישום האלגוריתם

אם נעביר לפעולה `Next` את הערך 6, נקבל מספר אקראי בין 0 ל-5. אם נעביר לפעולה `Next` את הערך 7, נקבל מספר אקראי בין 0 ל-6. אם כך:

? כיצד נדמה הטלת קובייה? כלומר כיצד נגריל מספר בין 1 ל-6?

נעביר לפעולה `Next` את הערך 6, על מנת לקבל מספר אקראי בין 0 ל-5. כדי "להמיר" את הערך שהתקבל לערך אפשרי עבור הטלת קובייה, נוסיף לו את הערך 1. כך יהיה בידינו מספר אקראי בין 1 ל-6. אפשרות נוספת היא להעביר לפעולה `Next` את הערכים: 1 ו-7 כך שיוגרל מספר בין 1 ל-6.

בתוכנית זו עלינו לבצע שתי הגרלות, ואכן לאחר שיצרנו עצם מהמחלקה `Random`, אנו יכולים להשתמש בו שוב ושוב להגרלות נוספות (אפילו מתוך תחומים שונים).

המחלקה Random שייכת למרחב השמות System לכן ההכרזה על השימוש ב-System מאפשרת לתוכנית להשתמש ישירות במחלקה Random.

התוכנית המלאה

```
/*
התוכנית מדמה שתי הטלות קובייה ומדפיסה את ערכן
*/
using System;
public class TwoDice
{
    public static void Main ()
    {
            // הגדרת משתנים
            int die1;          // תוצאת הטלת קובייה אחת
            int die2;          // תוצאת הטלת קובייה שנייה
            Random rnd = new Random(); // Random יצירת עצם מסוג
            die1 = rnd.Next(1,7);   // הטלת קובייה ראשונה
            die2 = rnd.Next(1,7);   // הטלת קובייה שנייה
            // הדפסות
            Console.WriteLine("The first one {0}: ", die1);
            Console.WriteLine("The second one {0}: ", die2);
        } // Main
} // class TwoDice
```

סוף פתרון בעיה 5

נסכם את המושגים החדשים שהוצגו בפתרון בעיה 5:

בשפת C# מוגדרת **מחלקה למספרים אקראיים** הנקראת Random. באמצעות עצם מהמחלקה ניתן להגריל מספרים אקראיים בתחום מבוקש.

לפני ביצוע הגרלות יש להצהיר על עצם מהמחלקה Random וליצור אותו, למשל כך:

```
Random rnd = new Random();
```

הפעולה Next של עצם מהמחלקה Random מקבלת כפרמטר מספר שלם חיובי n ומחזירה ערך אקראי שלם בתחום 0 עד n-1, או מקבלת כפרמטרים שני מספרים שלמים חיוביים n,m ומחזירה איבר אקראי בתחום n עד m-1. מאחר שזו פעולה של עצם, יש להפעילה באמצעות סימון הנקודה, למשל כך:

```
rnd.Next(101)
```

```
rnd.Next(1,7)
```

באמצעות עצם מהמחלקה Random, ניתן לבצע הגרלות חוזרות ונשנות, גם מתחומים שונים.

שאלה 4.25

בהינתן העצם rnd מהמחלקה Random,

א. מהו טווח הערכים האפשריים למשתנה השלם num בעקבות ביצוע ההוראה הבאה?

```
num = rnd.Next(6) + 1;
```

ב. מהו טווח הערכים האפשריים שיוצגו על המסך בעקבות ביצוע ההוראה הבאה?

```
Console.WriteLine(rnd.Next(10));
```

ג. מהו טווח הערכים האפשריים שיוצגו על המסך בעקבות ביצוע ההוראה הבאה?

```
Console.WriteLine(rnd.Next(11,111));
```

שאלה 4.26

בהינתן העצם `rndNum` מהמחלקה `Random`,

מהו טווח הערכים האפשריים עבור כל אחד מהביטויים הבאים?

א. `rndNum.Next(100)` ________________

ב. `rndNum.Next(1,100)` ________________

ג. `rndNum.Next(101)` ________________

ד. `rndNum.Next(101,201)` ________________

שאלה 4.27

בכל אחד מהסעיפים הבאים תארו ביטוי, המתייחס לעצם `rndNum` מהמחלקה `Random`, כך שערך הביטוי הוא בתחום המבוקש.

א. מספר אקראי שלם בין 0 ל-9 ________________

ב. מספר אקראי שלם בין 10 ל-100 ________________

ג. מספר אקראי שלם בין 100 ל-500 ________________

ד. מספר אקראי שלם **וזוגי** בין 0 ל-100 ________________

ה. מספר אקראי שלם **ותלת-ספרתי** ________________

שאלה 4.28

א. נסו לחשוב על נוסחה כללית כיצד נגריל מספר בטווח הערכים בין שני ערכים שלמים כלשהם x ו-y, כאשר ידוע ש-x הוא הערך הקטן מבין השניים.

ב. כתבו קטע תוכנית בשפת C# שיקלוט שני מספרים שלמים x ו-y (כאשר x<y), ויגריל מספר בתחום שבין x ל-y. השתמשו בנוסחה שכתבתם בסעיף א.

סיכום

פיתוח ויישום אלגוריתמים

הדגמנו בפרק זה את שלבי התהליך של פיתוח ויישום אלגוריתם, באמצעות בעיות ברמות שונות של קושי ושל מורכבות. למרות הבדלי הרמות, הקפדנו על שלבי התהליך בפתרון כל הבעיות: פיתוח ראשוני של הבעיה תוך בחינת הפלט עבור דוגמאות קלט מגוונות, ניסוח רעיון לפתרון על ידי פירוק לתת-משימות, בחירת משתנים, כתיבת אלגוריתם, יישום האלגוריתם במשפטי תוכנית, בדיקת מהלך ביצוע התוכנית באמצעות טבלת מעקב וכתיבת התוכנית המלאה והרצתה.

גם בשאר פרקי הספר, בעת פתרון בעיות אלגוריתמיות, נשתדל להקפיד על פיתוח אלגוריתם בשלבים ועל יישומו על פי שלבים אלה.

ייתכן כי בפתרון בעיות בהמשך, נאחד לפעמים חלק מן השלבים. נעשה זאת רק לאחר רכישת מיומנות מספקת בפתרון מפורט בשלבים. בכל מקרה נקפיד תמיד על ביצוע שלבי הבדיקה – הן הבדיקה ה"ידנית" והן הבדיקה באמצעות ההרצה.

פעולות חלוקה בשלמים

על ערכים מטיפוס שלם מוגדרות שתי פעולות: **מנת חלוקה ושארית חלוקה** (modulus):

מנת החלוקה של מספר שלם x במספר שלם y שווה לחלק השלם של x/y, ובמילים אחרות, למספר הפעמים שהערך y נכנס בערך x.

שארית החלוקה של מספר שלם x במספר שלם y מבטאת את השארית הנותרת לאחר חלוקה בשלמים של x ב-y, ובמילים אחרות, את מה שנותר אחרי שמפחיתים מ-x כפולות שלמות של y ככל שניתן.

תוצאת שארית החלוקה של מספר שלם כלשהו ב-n, יכולה להיות כל מספר שלם בתחום בין 0 ל-(n-1).

פעולות אלו מוגדרות עבור ערכים שלמים בלבד!

ניתן להיעזר בפעולות חלוקה בשלמים כדי לבצע **פירוק מספר דו-ספרתי לספרותיו**.

הטיפוס התווי

לכל תו **מותאם ערך מספרי** השמור לו.

לתווים עוקבים יש ערכים מספריים עוקבים.

סיכום מרכיבי שפת C# שנלמדו בפרק 4

פעולות חלוקה בשלמים

פעולת **מנת החלוקה** בשלמים מסומנת בשפת C# בסימן /.

כאשר הפעולה / מופעלת על שני ערכים שלמים היא מפורשת בשפת C# כפעולת חלוקה בשלמים.

בכל מקרה אחר (שלם וממשי, ממשי ושלם, ממשי וממשי) הפעולה / מפורשת בשפת C# כפעולת חלוקה על ערכים ממשיים.

פעולת **שארית החלוקה** בשלמים מסומנת בשפת C# בסימן %.

כדי לבצע פעולת חלוקה ממשית בין שני ערכים שלמים יש לבצע **המרה** (casting) של המחלק או של המחולק לערך ממשי.

המרה של ערך שלם x לערך ממשי נעשית בשפת C# כך: `(double) x`.

לפעולת ההמרה אין השפעה על טיפוס המשתנה המעורב בביטוי המומר. היא רק מנחה להתייחס אל ערכו כאילו היה מטיפוס ממשי, באופן זמני, רק לצורך החישוב הנוכחי.

הטיפוס התווי

ערך מטיפוס תווי מצוין ב-C# בין **גרשיים** בודדים, למשל כך: 'a'.

הצהרה על משתנה מטיפוס תווי נעשית באמצעות המילה `char`.

ניתן לשלב תווים בביטויים חשבוניים. בזמן חישוב הביטוי נלקחים הערכים המספריים המותאמים לכל תו ותו.

הטיפוס של ביטוי חשבוני הכולל תווים הוא מספרי. במידת הצורך, ניתן להמיר את ערכו של ביטוי כזה לערך מטיפוס תווי באמצעות פעולת ההמרה, למשל: `(char)('a' + 3)`.

בחירה אקראית

בשפת C# ניתן **להגריל מספר אקראי** באמצעות המחלקה `Random`.

כדי להשתמש במחלקה `Random` בתוך תוכנית C# צריך להכריז על שימוש במרחב השמות `System`:

```
using System;
```

לפני ביצוע הגרלות יש להצהיר על עצם מהמחלקה וליצור אותו. פעולת היצירה **new** מקצה לו מקום בזיכרון:

```
Random שם העצם = new Random();
```

הגרלת מספר אקראי שלם בתחום שבין 0 ל-n-1 מתבצעת באמצעות הפעלת הפעולה `Next(n)` של עצם מהמחלקה `Random`. הגרלת מספר שלם בתחום שבין n ל-m-1 מתבצעת על ידי הפעלת הפעולה `Next(n,m)` של עצם מהמחלקה `Random`. הפעולה מופעלת באמצעות סימון הנקודה ומחזירה ערך אקראי שלם בתחום המבוקש.

באמצעות עצם מהמחלקה `Random`, שהוקצה עבורו מקום בזיכרון, ניתן לבצע הגרלות חוזרות ונשנות, גם מתחומים שונים.

שאלות נוספות

שאלות נוספות לסעיף 4.1

1. פתחו אלגוריתם אשר הקלט שלו הוא מקדמים של משוואה ריבועית: a, b ו- c והפלט הוא שני הפתרונות האפשריים של המשוואה הריבועית. הניחו כי הקלט תקין ולמשוואה הריבועית אכן קיימים שני פתרונות. ישמו את האלגוריתם בשפת C#.
להזכירכם, הנה הנוסחה לחישוב פתרונותיה של משוואה ריבועית:

$$X_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

שאלות נוספות לסעיף 4.2

1. פתחו אלגוריתם אשר הקלט שלו הוא פרק זמן הנתון בימים והפלט שלו הוא מספר השבועות השלמים הכלולים בפרק הזמן הנתון. לדוגמה עבור הקלט 18 הפלט הוא 2. ישמו את האלגוריתם כתוכנית בשפת C#.

2. נהגי מוניות שירות יוצאים לדרכם רק כאשר כל המושבים במונית תפוסים.
פתחו אלגוריתם אשר הקלט שלו הוא מספר הנוסעים הממתינים למונית בתחנה ומספר המושבים במונית, והפלט שלו הוא מספר המוניות שניתן למלא במלואן ומספר הנוסעים שייוותרו בתחנה (הניחו כי לכל המוניות בתחנה יש מספר מושבים זהה).למשל:
עבור הקלט 25 7 יהיה הפלט: `ניתן למלא 3 מוניות, 4 נוסעים יוותרו בתחנה.`
עבור הקלט 35 7 יהיה הפלט: `ניתן למלא 5 מוניות, 0 נוסעים יוותרו בתחנה.`

3. נתון קטע התוכנית הבא : (num1 ו-num2 הם משתנים מטיפוס שלם)

```
a = num1 / 2;
b = num1 % 2;
c = num2 / 10;
d = num2 % 10;
e = num1 / num2;
d = num1 % num2;
```

א. ציינו ערך התחלתי של num1 שעבורו ערכיהם הסופיים של a ושל b יהיו 3 ו-0 בהתאמה.
ב. ציינו ערך התחלתי של num2 שעבורו ערכיהם הסופיים של c ושל d יהיו 6 ו-3 בהתאמה.
ג. תנו שתי דוגמאות לערכים התחלתיים של num1 ושל num2 שעבורם ערכיהם הסופיים של e ושל f יהיו 3 ו-2, בהתאמה.

שאלות מסכמות לפרק 4

1. במכולת של חנניה מוכרים מסטיקים בקבוצות על פי גודל האריזות הקיימות בחנות, והמסטיקים הנותרים נמכרים בודדים. שווי כל מסטיק הוא 0.2 ₪.
יש לפתח אלגוריתם שהקלט שלו הוא מספר המסטיקים הנמצא במכולת וגודל האריזות הקיימות בחנות (כל האריזות באותו הגודל). הפלט של האלגוריתם הוא השווי הכולל של המסטיקים בחבילות השלמות ושווי המסטיקים שנותרים לא ארוזים.
למשל, עבור הקלט 7 100 הפלט הדרוש הוא 0.4 19.6.
בחנו את הפלט עבור דוגמאות קלט מייצגות, והציגו את חלוקת המשימה המתוארת לתת-משימות.

2. חנניה מהמכולת מעוניין באלגוריתם אשר יעזור לו להחזיר עודף במטבעות בצורה היעילה ביותר, כלומר במספר המטבעות הקטן ביותר.
בהנחה כי חנניה יכול להחזיר עודף אך ורק במטבעות של 1 ₪, של 5 ₪ ושל 10 ₪, כתבו אלגוריתם המקבל כקלט את הסכום שחנניה צריך להחזיר כעודף (מספר שלם), ומציג כפלט :
א. את מספר המטבעות שיחזיר חנניה מכל סוג .
ב. את מספר המטבעות הכולל שיחזיר חנניה.
למשל עבור הקלט 18 יתקבל הפלט :
א. 1 ₪ : 3, 5 ₪ : 1, 10 ₪ : 1
ב. 5 מטבעות.
ישמו את האלגוריתם בשפת #C.

3. כתבו קטע תוכנית בשפת #C המגריל מספר תלת-ספרתי. התוכנית תדפיס את ספרות המספר כל אחת בשורה נפרדת. מאות עשרות ואחדות

תבניות – פרק 4

פירוט מלא של התבניות ושל שאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

חלוקת כמות פריטים לקבוצות בגודל נתון

שם התבנית: **מנת החלוקה לקבוצות של כמות פריטים**
נקודת מוצא: שני מספרים שלמים חיוביים; quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)
מטרה: מספר הקבוצות המלאות מחלוקה של quantity הפריטים לקבוצות בגודל num
אלגוריתם:
השם ב-groups את מנת החלוקה של quantity ב-num

שם התבנית: **שארית החלוקה לקבוצות של כמות פריטים**
נקודת מוצא: שני מספרים שלמים חיוביים; quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)
מטרה: מספר הפריטים העודף בחלוקה של quantity הפריטים לקבוצות בגודל num
אלגוריתם:
השם ב-remainder את שארית החלוקה של quantity ב-num

פירוק מספר חיובי לספרותיו

שם התבנית: **ספרת האחדות של מספר**
נקודת מוצא: מספר דו-ספרתי חיובי num
מטרה: חישוב ספרת האחדות של num
אלגוריתם:
השם ב-units את שארית החלוקה של num ב-10

שם התבנית: **ספרת העשרות של מספר**
נקודת מוצא: מספר דו-ספרתי חיובי num
מטרה: חישוב ספרת העשרות של num
אלגוריתם:
השם ב-tens את מנת החלוקה של num ב-10

בניית מספר

שם התבנית: **בניית מספר**
נקודת מוצא: שתי ספרות left ו-right
מטרה: בניית מספר דו-ספרתי מהספרות הנתונות
אלגוריתם:
*השם ב-*num *את הערך של הביטוי החשבוני* left * 10 + right

פרק 5 – ביצוע מותנה

בשני הפרקים הקודמים ראינו אלגוריתמים שבמהלך ביצועם מתבצעת כל אחת מהוראות האלגוריתם. בפרק זה נכיר אלגוריתמים אשר במהלך ביצועם לא מתבצעות תמיד כל הוראות האלגוריתם. אלגוריתמים אלה כוללים הוראות המורות על ביצוע קבוצת הוראות אחת או על קבוצת הוראות אחרת, בהתאם לקיומו או לאי-קיומו של תנאי. הוראות אלו נקראות **הוראות לביצוע-בתנאי**. קיומו של התנאי תלוי בקלט לאלגוריתם.

למשל כאשר נערכות בחירות בין שני מועמדים, משווים את מספרי הקולות לכל מועמד. המועמד שצבר יותר קולות הוא המנצח בבחירות. הקלט של אלגוריתם להכרזת המנצח יהיה מספר הקולות אשר צבר כל מועמד. אם יתקיים התנאי שהנתון הראשון בקלט גדול מן השני אז תתבצע באלגוריתם הוראה להכרזת המועמד הראשון כמנצח. אחרת תתבצע הוראה להכרזת המועמד השני כמנצח.

5.1 הוראה לביצוע-בתנאי

הוראה לביצוע-בתנאי במבנה אם... אחרת...

בעיה 1

מטרת הבעיה ופתרונה: הצגת אלגוריתם הכולל הוראה לביצוע-בתנאי.

> פָּלִינְדְּרוֹם (palindrome) הוא מילה, מספר או משפט שניתן לקרוא משני הכיוונים, משמאל לימין ומימין לשמאל, ולקבל אותה תוצאה. למשל השם ישי הוא שם פלינדרומי, וכן המילים זוז, שמש, הסוסה. המילה הפלינדרומית הארוכה ביותר בעברית שיש לה משמעות היא "ולכשתשכלו". המשפט: "ילד כותב בתוך דלי" גם הוא פלינדרומי. המספרים 17371 ו-4994 הם דוגמאות למספרים פלינדרומים.
> פתחו אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי תלת-ספרתי, והפלט שלו הוא הודעה אם המספר הנתון הוא פלינדרום. ישמו את האלגוריתם בתוכנית מחשב בשפת C#.

ניתוח הבעיה בעזרת דוגמאות

דוגמאות למספרים שלמים חיוביים תלת-ספרתיים פלינדרומים: 787 424 777

דוגמאות למספרים שלמים חיוביים תלת-ספרתיים שאינם פלינדרומים: 778 234 192

שאלה 5.1

הגדירו כלל פשוט המתאר מתי מספר הוא פלינדרום ומתי אינו פלינדרום.

בדוגמאות הפשוטות שבחנו התברר שמספר הוא פלינדרומי רק כאשר ספרת האחדות שווה לספרת המאות.

פירוק הבעיה לתת-משימות

על פי ניתוח הבעיה נפרק את המשימה העומדת לפנינו באופן הבא:

1. קליטת מספר שלם חיובי תלת-ספרתי.
2. מציאת ספרת האחדות.

3. מציאת ספרת המאות.
4. השוואת הספרות. אם הספרות שוות נציג הודעה שהמספר פלינדרומי, אחרת נציג הודעה שהמספר אינו פלינדרומי.

בחירת משתנים

num – שלם, ישמור את המספר התלת-ספרתי הנקלט.
units – שלם, ישמור את ספרת האחדות.
hundreds - שלם, ישמור את ספרת המאות.

האלגוריתם

את תת-משימה 2 ו-3 למדנו לבצע בפרק הקודם. על מנת למצוא את ספרת האחדות של מספר כלשהו, נחשב את תוצאת שארית החילוק של המספר ב-10. על מנת למצוא את ספרת המאות של מספר תלת-ספרתי, נחשב את תוצאת החילוק של המספר ב-100.

כיצד ננסח באלגוריתם את תת-משימה 4?
ננסח תנאי אשר על פי קיומו ניתן לקבוע אם המספר הוא פלינדרומי. התנאי יהיה: ספרת האחדות שווה לספרת המאות.
אם התנאי מתקיים יש להודיע: המספר פלינדרומי.
אם התנאי לא מתקיים יש להודיע: המספר אינו פלינדרומי.

הוראה זו מבטאת את הרעיון של בחירה באחת מבין שתי אפשרויות לביצוע על פי תנאי. נציג אותה במבנה הבא:

אם ספרת האחדות שווה לספרת המאות
הצג כפלט: המספר פלינדרומי
אחרת
הצג כפלט: המספר אינו פלינדרומי

הוראה במבנה זה נקראת **הוראה לביצוע-בתנאי**. הוראה לביצוע-בתנאי היא הוראת **בקרה**, משום שהיא משפיעה על מהלך הביצוע של האלגוריתם, כלומר קובעת אם יבוצעו הוראות אלו או אחרות.

יישום האלגוריתם

ההוראה לביצוע-בתנאי המופיעה באלגוריתם מיושמת בשפת C# בהוראת **if…else…**, ואופן כתיבתה דומה לצורת הכתיבה העברית *אם...אחרת...*.

אבל כיצד כותבים ב-C# תנאי כמו זה המופיע באלגוריתם: *ספרת האחדות שווה לספרת המאות*? פעולת ההשוואה נכתבת בעזרת סימן הפעולה == (שני סימני שוויון רצופים).

לכן, את תת-משימה 4 באלגוריתם ניתן ליישם בשפת C# כך:

```
if (units == hundreds)
{
    Console.WriteLine("The number {0} is a palindrome", num);
}
else
{
    Console.WriteLine("The number {0} is not a palindrome", num);
}
```

התוכנית המלאה

```
/*
הקלט: מספר שלם חיובי תלת-ספרתי
הפלט: הודעה אם המספר הוא פלינדרום
*/
using System;
public class Palindrome
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int num;         // מספר שלם חיובי תלת-ספרתי
        int units;       // ספרת האחדות
        int hundreds;    // ספרת המאות
        // קליטת המשתנים
1.      Console.Write("Enter a 3 digit number: ");
2.      num = int.Parse(Console.ReadLine());
        // פירוק ספרת האחדות וספרת המאות
3.      units = num % 10;
4.      hundreds = num / 100;
        //  ההוראה לביצוע-בתנאי
5.      if (units == hundreds)
        {
            //הצגת הודעה: המספר פלינדרומי
5.1.        Console.WriteLine("{0} is a palindrome", num);
        }
6.      else
        {
            //הצגת הודעה: המספר אינו פלינדרומי
6.1.        Console.WriteLine("{0} is not a palindrome", num);
        }
    } // Main
} // class Palindrome
```

המעקב

נבדוק את התוכנית Palindrome באמצעות מעקב אחר ביצועה עבור דוגמאות קלט שונות.

עבור הקלט 363, יתקיים התנאי *ספרת האחדות שווה לספרת המאות*, ותתקבל טבלת המעקב הבאה:

מספר השורה	המשפט לביצוע	num	units	hundreds	units == hundreds	פלט
1	`Console.Write("Enter a 3 digit number: ");`	?	?	?		Enter a 3 digit number:
2	`num = int.Parse(...);`	363	?	?		
3	`units = num % 10;`	363	3	?		
4	`hundreds = num / 100;`	363	3	3		
5	`if (units == hundreds)`	363	3	3	אמת	
5.1	`Console.WriteLine("{0} is a palindrome", num);`	363	3	3		363 is a palindrome

עבור הקלט 366, לא מתקיים התנאי, ומתקבלת טבלת המעקב הבאה:

מספר השורה	המשפט לביצוע	num	units	hundreds	units == hundreds	פלט
1	`Console.Write("Enter a 3 digit number: ");`	?	?	?		Enter a 3 digit number:
2	`num = int.Parse(...);`	366	?	?		
3	`units = num % 10;`	366	6	?		
4	`hundreds = num / 100;`	366	6	3		
5	`if (units == hundreds)`	366	6	3	שקר	
6.1	`Console.Write("{0} is not a palindrome",num);`	366	6	3		366 is not a palindrome

שימו ♥ להבדל בין עמודת "המשפט לביצוע" בשתי הטבלאות. בטבלה הראשונה מופיע המשפט:

```
Console.WriteLine("{0} is a palindrome", num);
```

זאת מכיוון שהתנאי מתקיים ולכן מתבצע תחום ה-**if** של משפט ה-**if**.

לעומת זאת, בטבלה השנייה מופיע המשפט:

```
Console.WriteLine("{0} is not a palindrome", num);
```

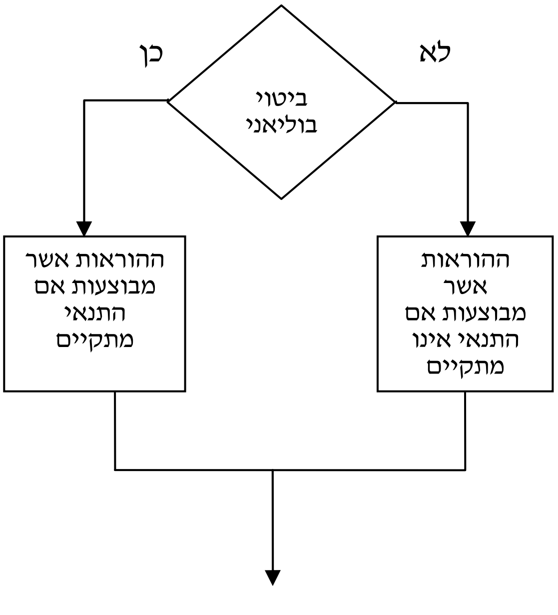
זאת מכיוון שהתנאי אינו מתקיים ולכן מתבצע תחום ה-**else** של המשפט.

סוף פתרון בעיה 1

הוראה לביצוע-בתנאי כוללת בתוכה כמובן תנאי, שמכוון את המשך הביצוע. באלגוריתם שנתנו לפתרון בעיה 1 התנאי ששילבנו בהוראה הוא *ספרת האחדות שווה לספרת המאות*.

התנאי העומד בבסיסה של הוראת ביצוע-בתנאי מיוצג ב**ביטוי בוליאני**. כמו ביטוי חשבוני, גם לביטוי בוליאני יש ערך. אלא שערך זה אינו ערך מספרי. ערכו של ביטוי בוליאני יכול להיות אחד משניים – אמת (**true**) או שקר (**false**). אם התנאי שמייצג הביטוי הבוליאני מתקיים אז ערכו של הביטוי הבוליאני הוא **true**. אם התנאי שמייצג הביטוי הבוליאני אינו מתקיים אז ערכו של הביטוי הבוליאני הוא **false**.

ניתן להמחיש את המשמעות של הוראה לביצוע-בתנאי באמצעות תרשים הזרימה הבא:



הידעת?
המילה "בוליאני" נגזרה משמו של המתמטיקאי האנגלי ג'ורג' בול (1864-1815). הוא היה בין שאר עיסוקיו לוגיקן – חוקר תורת ההיגיון. הוא היה המתמטיקאי הראשון שראה בלוגיקה ענף של המתמטיקה. הוא נחשב לאבי הלוגיקה הסימבולית (אלגברה של הלוגיקה) הקרויה היום אלגברה בוליאנית.

במהלך הספר לא נציג אלגוריתמים באמצעות תרשימים. הצגה זו ארוכה מאוד וכן שונה מאוד מן המראה של תוכנית מחשב. אם השימוש בתרשימים מסייע לכם, כדאי לכם להשתמש בהם מדי פעם במהלך פיתוח אלגוריתם.

לעתים ננסח את התנאי באופן מילולי ולעתים נעדיף לנסח אותו בעזרת סימנים. ניתן לבחור בכל צורת ניסוח, כל עוד התנאי המתקבל הוא ברור וחד-משמעי. למשל עבור המשימה: השוואת ערכי המשתנים a ו-b והשמת הערך הקטן מביניהם במשתנה min, ניתן לנסח את התנאי במילים או בעזרת סמלים:

אם ערכו של a *קטן מערכו של* b   או   *אם* $a < b$

נכיר ביטויים בוליאניים המציינים השוואה של ערכים, של משתנים ושל ביטויים חשבוניים. למשל: units = = hundreds , $a + b > 0$ ו-$a < b$.

לסיכום – נציג את אופן הכתיבה האלגוריתמית של הוראה לביצוע-בתנאי ואת יישומה בשפת C#:

הוראה לביצוע-בתנאי נכתבת בצורה זו:

אם <*ביטוי בוליאני*>

    <*סדרת הוראות 1*>

אחרת

    <*סדרת הוראות 2*>

ביטוי בוליאני מייצג תנאי, שערכו יכול להיות **true** (אמיתי) או **false** (שקרי).

ביצוע של הוראה לביצוע-בתנאי מתחיל תמיד בחישוב ערכו של הביטוי הבוליאני. אם ערכו אמיתי, תתבצע סדרת ההוראות הראשונה, אחרת תתבצע סדרת ההוראות השנייה.

בשפת C# הוראה לביצוע-בתנאי מיושמת במשפט `if`.

מבנה משפט `if` ב-C# הוא:

```
if (ביטוי בוליאני)
{
    ההוראות אשר יבוצעו אם התנאי מתקיים
}
else
{
    ההוראות אשר יבוצעו אם התנאי אינו מתקיים
}
```

החלק שנמצא בתוך זוג הסוגריים המסולסלים ({...}) הראשון נקרא **תחום ה-`if`** ובו נמצאות ההוראות אשר יבוצעו אם התנאי מתקיים. החלק שנמצא בתוך זוג הסוגריים המסולסלים השני נקרא **תחום ה-`else`** ובו נמצאות ההוראות אשר מבוצעות אם התנאי אינו מתקיים.

שימו ♥: במקרים שיש הוראה אחת לביצוע בתחום ה-**if**, אפשר להשמיט את הסוגריים המסולסלים של התחום. גם במקרים שיש הוראה אחת לביצוע בתחום ה-**else**, דין הסוגריים זהה וניתן להשמיטם.

הביטוי הבוליאני שהופיע במשפט ה-`if` שראינו בפתרון בעיה 1 השתמש בסימן ההשוואה `==`. בשפת C# קיימים סימני השוואה נוספים. בטבלה הבאה מובאים סימני ההשוואה בשפת C#.

סימן השוואה ב-C#	סימן ההשוואה המקובל במתמטיקה	משמעות סימן ההשוואה	דוגמה ב-C#	דוגמה במתמטיקה
==	=	שווה	`x == 5`	$x = 5$
!=	$\neq$	שונה	`x != y`	$x \neq y$
<	<	קטן	`x < 2`	$x < 2$
<=	$\leq$	קטן או שווה	`x <= 1`	$x \leq 2$
>	>	גדול	`y > 0`	$y > 0$
>=	$\geq$	גדול או שווה	`y >= 8`	$y \geq 8$

סימני ההשוואה בשפת C#

שימו ♥ : הסימנים המתמטיים $\leq$, $\geq$, = ו - $\neq$ כתובים ב-C# בצורה שונה במקצת. בשפת C# ההשוואה שבתוך התנאי מתבצעת על ידי סימן הפעולה `==` ולא על ידי הסימן `=`. הסימן `=` שמור ב-C# להשמה.

שאלה 5.2

נסחו הוראה לביצוע-בתנאי עבור כל אחת מן המשימות הבאות:

א. השוואת ערכי המשתנים `a` ו-`b` והצגת הודעה אם הערכים שווים או שונים.

ב. השוואת ערכי המשתנים `a` ו-`b` והפחתת ערכו של המשתנה הקטן מהמשתנה הגדול. אם שני המשתנים שווים יש להפחית את ערכו של b מערכו של a.

שאלה 5.3

כתבו כל אחד מן התנאים המילוליים הבאים כביטוי בוליאני המשתמש בסימני ההשוואה של C#:

תנאי	ביטוי בוליאני
ערך המשתנה `a` שווה ל-0.	
ערך המשתנה `a` שווה לערך המשתנה `b`.	
ערך המשתנה `a` שווה לכפליים ערכו של המשתנה `b`.	
ערך המשתנה `a` שונה מערך המשתנה b.	
סכום ערכי המשתנים `a` ו-`b` קטן או שווה ל-10.	

שאלה 5.4

נניח שערכי המשתנים `a` ו-`b` הם 1 ו-2 בהתאמה. ציינו עבור כל אחד מן הביטויים הבוליאניים הבאים אם ערכו **true** או **false**.

ביטוי בוליאני	ערך
`a == 1`	
`a == b`	
`3a == b + 1`	
`2a <= b`	
`2a != b`	

שאלה 5.5

במשתנים `girafaHeight` ו-`girafHeight` שמורים הגבהים של ג'ירף ושל ג'ירפה. כתבו משפט **if** מתאים לביצוע המשימות הבאות:

א. הצגת הודעה אם הג'ירפה גבוהה מ-1.70 מ', או לא.

ב. הצגת הודעה אם הג'ירף גבוה יותר או אינו גבוה יותר מהג'ירפה.

כפי שראינו בפתרון בעיה 1: בטבלת מעקב אחר מהלך ביצוע של תוכנית הכוללת משפט **if**, נוח להוסיף עמודה עבור הביטוי הבוליאני, ולציין בה את ערך הביטוי. בטבלה שבפתרון בעיה 1 כתבנו "אמת" או "שקר". מעתה נכתוב **true** או **false**.

שימו ♥: כאשר מופיע בתוכנית משפט **if** (הוראה לביצוע-בתנאי), יש לבדוק את מהלך הביצוע עבור דוגמאות קלט מגוונות. על הדוגמאות לכלול קלט שיביא לכך שערכו של הביטוי הבוליאני שבמשפט יהיה **true** וכן קלט שיביא לכך שערכו של הביטוי הבוליאני שבמשפט יהיה **false**. קלטים כאלו נקראים קלטים מייצגים.

לא מספיק לבדוק את מהלך הביצוע רק עבור אחד משני המקרים, כיוון שמקרה אחד אינו מעיד על האחר. עלינו לבדוק את שניהם, כפי שמדגימה השאלה הבאה.

שאלה 5.6

מטרתו של משפט ה-**if** הבא היא השמת הערך הגדול מבין המשתנים a ו-b במשתנה max. חישוב מקסימום היא תבנית שימושית מאוד, שמשמשת בפתרון בעיות רבות.

```
if (a > b)
{
        max = a;
}
else
{
        max = b;
}
```

בחרו שתי דוגמאות של קלטים מייצגים לבדיקת המשפט.

באחת יהיה ערכו ההתחלתי של a גדול מערכו ההתחלתי של b, ובשנייה יהיה ערכו ההתחלתי של a קטן מערכו ההתחלתי של b.

בדקו את מהלך ביצוע המשפט באמצעות טבלת מעקב עבור כל אחת מן הדוגמאות.

מה יהיה מהלך ביצוע המשפט עבור המקרה שבו הערכים ההתחלתיים של a ו-b שווים?

הוראה לביצוע-בתנאי במבנה *אם...*

לעתים ברצוננו לבצע חלק של אלגוריתם כאשר תנאי מסוים מתקיים, ואיננו רוצים לבצע מאומה כאשר התנאי אינו מתקיים. נראה זאת בפתרון הבעיה הבאה.

בעיה 2

מטרת הבעיה ופתרונה: הצגת הוראה לביצוע-בתנאי במבנה *אם...*(ללא החלק *אחרת...*).

> תלמידי כיתות י' בבית הספר טסים לירת. יש להזמין מספר מתאים של חלליות כך שלכל תלמיד יהיה מקום ישיבה ושמספר החלליות יהיה קטן ככל האפשר.

פתחו וישמו אלגוריתם להזמנת חלליות לירח כך שהקלט שלו הוא מספר התלמידים ומספר המושבים בחללית, והפלט שלו הוא מספר החלליות שיש להזמין.

ניתוח הבעיה בעזרת דוגמאות

ייתכן כי מספר התלמידים הוא כפולה של מספר המושבים בחללית. במקרה כזה בכל החלליות שיוזמנו יהיו כל המושבים תפוסים.

לעומת זאת, ייתכן כי מספר התלמידים אינו כפולה של מספר המושבים בחללית. במקרה כזה תוזמן גם חללית אחת אשר רק חלק מהמושבים בה יהיו תפוסים .

שאלה 5.7

בחרו שתי דוגמאות קלט מייצגות וציינו את הפלט עבור כל אחת מהן.

? כיצד נחשב את מספר החלליות הדרוש?

יש לחשב את המנה ואת השארית של חלוקת מספר התלמידים במספר המושבים בחללית. מנת החלוקה שווה למספר החלליות המלאות. שארית החלוקה תקבע אם יש צורך בחללית נוספת. אם שארית החלוקה היא 0 הרי מספר התלמידים הוא כפולה של מספר המושבים בחללית. אחרת יש להזמין חללית נוספת שתפוסתה תהיה חלקית ושווה לשארית.

פירוק הבעיה לתת-משימות

1. קליטת מספר התלמידים ומספר המושבים
2. חישוב מספר החלליות המלאות
3. חישוב מספר התלמידים שיישארו לאחר מילוי החלליות המלאות
4. אם צריך חללית נוספת, חישוב מספר החלליות הכולל
5. הצגת הפלט: מספר החלליות

בחירת משתנים

נבחר את המשתנים הבאים מטיפוס שלם:

studentsNum – ישמור את מספר התלמידים.

seatsPerShip – ישמור את מספר המושבים בחללית.

shipsNum – ישמור את מספר החלליות שיש להזמין.

leftoverNum – ישמור את מספר התלמידים אשר ייוותרו לאחר מילוי החלליות המלאות.

האלגוריתם

כיצד ננסח באלגוריתם את תת-משימה 4?

ניתן לתאר תת-משימה זו באופן הבא:

אם מספר התלמידים שנשארו גדול מ-0

הגדל ב-1 את shipsNum

האלגוריתם לפתרון הבעיה כולל הוראה לביצוע-בתנאי במבנה *אם...* (ללא החלק *אחרת...*):

המשמעות של **הוראה לביצוע-בתנאי במבנה** *אם...* היא שאם התנאי שבהוראה מתקיים יבוצעו ההוראות המתאימות, אחרת לא יבוצע דבר.

יישום האלגוריתם

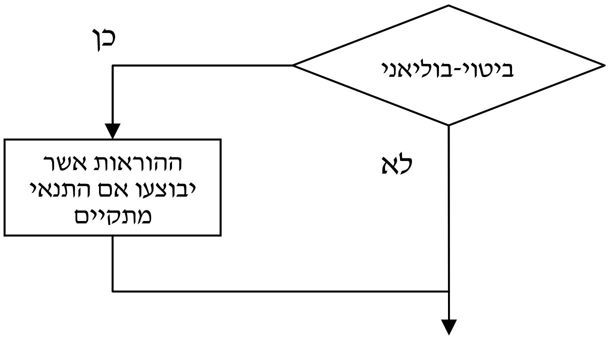
```
/*
קלט: מספר התלמידים והמושבים
פלט: מספר החלליות להזמנה
*/
using System;
public class ShipspaceOrder
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int studentsNum;    // מספר התלמידים
        int seatsPerShip;   // מספר המושבים בחללית
        int shipsNum;        // מספר החלליות שיש להזמין
        int leftoverNum;    // מספר התלמידים שיישארו
            // לאחר מילוי החלליות המלאות
        // קליטת המשתנים
1.      Console.Write("Enter number of students:");
2.      studentsNum = int.Parse(Console.ReadLine());
3.      Console.Write("Enter number of seats in a spaceship: ");
4.      seatsPerShip = int.Parse(Console.ReadLine());
              // חישוב מספר החלליות המלאות
5.      shipsNum = studentsNum / seatsPerShip;
        // חישוב מספר התלמידים שיוותרו
6.      leftoverNum = studentsNum % seatsPerShip;
        // ההוראה לביצוע-בתנאי
7.      if (leftoverNum > 0) // דרושה חללית נוספת שתפוסתה תהיה חלקית.
        {
7.1.        shipsNum = shipsNum + 1;
        }
                Console.WriteLine("The number of spaceships is {0}",
                                                              shipsNum);
    } // Main
} // class ShipspaceOrder
```

שאלה 5.8

עקבו אחר ביצוע התוכנית באמצעות שתי טבלאות מעקב. האחת, כאשר התנאי מתקיים, והשנייה, כאשר אין התנאי מתקיים. לדוגמה: עבור הקלט 120 תלמידים ו-40 מושבים בכל חללית התנאי לא יתקיים, ואילו עבור הקלט 120 תלמידים ו-50 מושבים בכל חללית התנאי יתקיים.

סוף פתרון בעיה 2

ניתן להמחיש הוראה לביצוע-בתנאי במבנה *אם*... באמצעות תרשים הזרימה הבא:



בשפת C# מבנה משפט ה-`if` ליישום הוראה לביצוע-בתנאי במבנה *אם*... הוא:

```
if (ביטוי בוליאני)
{
      ההוראות אשר יבוצעו אם התנאי מתקיים
}
```

שימו ♥: במקרים שיש הוראה אחת לביצוע, אפשר להשמיט את הסוגריים המסולסלים של תחום ה-**if**.

שימו ♥: הערה שמלווה את התנאי (דרושה חללית...) מבהירה ביטוי בוליאני שמשמעותו איננה ברורה מיד עם קריאתו. נקרא להערה זו **תיאור משמעות של קיום תנאי**.

תיאור משמעות של קיום תנאי הוא תיעוד המסביר את תפקידו של ביטוי בוליאני בהוראה לביצוע-בתנאי. תיאור משמעות קיום תנאי מסייע לנו בקריאת תוכנית ובהבנתה.

בתוכניות שנפתח נשתדל לצרף תיאורי משמעות של קיום או של אי-קיום תנאי במשפטי `if` אשר כדאי להבהירם.

שאלה 5.9

בנו טבלת מעקב אחר ביצוע התוכנית `SpaceshipOrder` לפתרון בעיה 2 עבור הקלט: 180 תלמידים ו-60 מושבים.

שאלה 5.10

ביטוי בוליאני עשוי לכלול השוואה בין ביטויים חשבוניים שאינם פשוטים. למשל, `(a / 2) == 0` או `(a + b) == (c + d)`.

כתבו משפט `if` אשר בו:

א. אם ערכו של `a` גדול מפעמיים ערכו של b, מוכפל ערכו של `c` ב-2.

ב. אם ערכו של `c` קטן מסכום ערכי `a` ו-b, מופחת מ-`c` ערכו של `a`.

ג. אם ערכו של `a` הוא כפולה של 10, מושם ב-`b` ערכו של `a`.

ד. אם מכפלת ערכי `a` ו-`b` גדולה מסכום ערכי `b` ו-`c`, סימנו של הערך הנתון ב-`c` מתהפך.

שאלה 5.11

המשתנה a מכיל מספר שלם חיובי קטן מ-100.

השלימו את תיאור המשמעות של קיום תנאי בכל אחד מן המשפטים הבאים, כלומר הסבירו את תפקידו של כל תנאי ומה הוא בודק:

א.

```
//___________________________
if ((a / 10) > 5)
    Console.WriteLine("Pass");
```

ב.

```
//___________________________
if (a == (a % 10))
    Console.WriteLine("A");
```

שאלה 5.12

נתון קטע התוכנית הבא, שהמשתנים בו הם מטיפוס שלם:

```
max = a;
if (b > a)
    max = b;
```

א. בנו טבלאות מעקב אחר מהלך ביצוע קטע התוכנית עבור הערכים ההתחלתיים 30 ו-30 ועבור הערכים ההתחלתיים 40 ו-70 במשתנים a ו-b בהתאמה.

ב. מהי מטרת קטע התוכנית?

ג. נתון קטע התוכנית הבא:

```
if (a >= b)
    max = a;
else
    max = b;
```

האם יש הבדל בין מטרת קטע התוכנית הנתון בסעיף זה ובין מטרת קטע התוכנית שהוצג בתחילת השאלה? הסבירו את תשובתכם.

להעמקה בתבנית ***מציאת מקסימום*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 5.13

פתחו אלגוריתם אשר הקלט שלו הוא שני ציונים של תלמיד, שערכם הוא מספר שלם בין 0 ל-100 והפלט שלו הוא מספר המציין כמה מן הציונים גבוהים מ-80.

א. מהם ערכי הפלט האפשריים?

ב. ישמו את האלגוריתם בשפת C#.

התניית ביצוע של שתי הוראות או יותר

עד כה ראינו הוראות פשוטות לביצוע-בתנאי: אם התנאי התקיים התבצעה הוראה יחידה. גם במקרה שהתנאי לא התקיים התבצעה הוראה יחידה. בבעיה הבאה נציג הוראה לביצוע-בתנאי שתחומיה השונים כוללים יותר מהוראה אחת, ומורים על ביצוע כמה תת-משימות.

בעיה 3

מטרת הבעיה ופתרונה: הצגת הוראה לביצוע-בתנאי שתחומיה כוללים מספר הוראות.

> פתחו אלגוריתם המתאר משחק שנקרא "שש אש". הקלט של האלגוריתם הוא מספר שלם x. אם x מתחלק ב-6 יזכה המשתתף ב- 6·x ₪, אך אם x אינו מתחלק ב-6, יפסיד המשתתף 10·x ₪. פלט האלגוריתם הוא הודעת סכום הזכייה או ההפסד מוקף בכוכביות. ישמו את האלגוריתם בשפת C#.

פירוק הבעיה לתת-משימות

1. קליטת המספר
2. חישוב סכום הזכייה או ההפסד
3. הצגת הסכום מוקף בכוכביות בצירוף הודעה מתאימה

בחירת משתנים

נבחר את המשתנים הבאים מטיפוס שלם:

num – לשמירת המספר הניתן כקלט

sum – לשמירת סכום הזכייה או ההפסד

האלגוריתם

1. *קלוט מספר ב-*num
2. *אם ערכו של* num *מתחלק ב-6 ללא שארית*
   2.1. *חשב את מכפלת* num *ב-6 והשם ב-*sum
   2.2. *הצג הודעה על זכייה בסכום* sum *המוקפת בכוכביות*
3. *אחרת*
   3.1. *חשב את מכפלת* num *ב-10 והשם ב-*sum
   3.2. *הצג הודעה על הפסד של הסכום* sum *המוקפת בכוכביות*

יישום האלגוריתם

```
/*
קלט: מספר שלם
פלט: הודעה על זכייה אם המספר מתחלק ב-6, או על הפסד אם המספר אינו
מתחלק ב-6. ההודעה תכלול את סכום הזכייה או ההפסד
*/
using System;
public class SheshEsh
{
    public static void Main ()
```

```
    {
        // הצהרה על משתנים בתוכנית
        int num;   //המספר שנקלט
        int sum;   //סכום הזכייה או ההפסד
        // קליטת המשתנים
1.      Console.Write("Enter a number: ");
2.      num = int.Parse(Console.ReadLine());
        // ההוראה לביצוע-בתנאי: חישוב סכום הזכייה או ההפסד
3.      if (num % 6 == 0)
        {   // הזכייה
3.1.        sum = num * 6;
3.2.        Console.WriteLine("**********************");
3.3.        Console.WriteLine("* You won {0} shekels *", sum);
3.4.        Console.WriteLine("**********************");
        }
4.      else
        {     // ההפסד
4.1.        sum = num * 10;
4.2.        Console.WriteLine("************************");
4.3.        Console.WriteLine("* You  lost {0} shekels *", sum);
4.4.        Console.WriteLine("************************");
        }
    }// Main
}// class SheshEsh
```

מעקב

נעקוב אחר מהלך ביצוע התוכנית SheshEsh עבור הקלט 12:

	המשפט לביצוע	num	sum	num%6==0	**פלט**
1	Console.Write("Enter a number: ");	?	?		Enter a number:
2	num = **int**.Parse(Console. ReadLine());	12	?		
3	**if** (num % 6 == 0)	12	?	**true**	
3.1	sum = num * 6;	12	72		
3.2	Console.WriteLine("**…")	12	72		*********************
3.3	Console.WriteLine("You Won {0} shekels", sum);	12	72		* You won 72 shekels *
3.4	Console.WriteLine("**…")	12	72		*********************

סוף פתרון בעיה 3

שאלה 5.14

בנו טבלת מעקב אחר ביצוע התוכנית SheshEsh מפתרון בעיה 3 עבור הקלט 10.

שאלה 5.15

במשתנים x ו-y שמורים שני ערכים מטיפוס שלם.

מטרת התוכנית היא להציג כפלט את המספר הגדול ראשון ואחריו את המספר הקטן.

השלימו את משפט ה-**if** הבא באמצעות ביטוי בוליאני מתאים.

```
if (_________________)
{
```

```
    temp = x;
    x = y;
    y = temp;
}
Console.WriteLine(x, y);
```

שאלה 5.16

פתחו אלגוריתם שהקלט שלו הוא שני מספרים חיוביים. המספר הראשון מציין את משקלו של החתול גארפילד בק״ג והמספר השני מציין את משקלו של הכלב סנופי. אם משקלו של גארפילד גדול ממשקלו של סנופי, האלגוריתם צריך לחשב כמה ק״ג שוקל גארפילד יותר מסנופי ולהציג הודעה מתאימה הכוללת את הערך שחושב. אחרת האלגוריתם צריך לחשב כמה ק״ג שוקל סנופי יותר מגארפילד ולהציג הודעה מתאימה הכוללת את הערך שחושב.

למשל פלט מתאים עבור הקלט 15 10 הוא:

```
Snoopy is heavier than Garfield, the difference is 5 kg
```

פלט מתאים עבור הקלט 13 17 הוא:

```
Garfield is heavier than Snoopy, the difference is 4 kg
```

ישמו את האלגוריתם בשפת C#.

שאלה 5.17

פתחו אלגוריתם שהקלט שלו הוא שני מספרים שלמים. אם המספר הראשון גדול מהשני, האלגוריתם מחשב את סכומם ומציג אותו כפלט. אחרת הוא מחשב את מכפלת המספרים ומציג אותה כפלט. ישמו את האלגוריתם בשפת C#.

ביטויים בוליאניים הכוללים תווים

הביטויים הבוליאניים שראינו עד כה כללו פעולות השוואה על מספרים. עם זאת המספרים השלמים או המספרים הממשיים אינם הטיפוסים היחידים שערכיהם ניתנים להשוואה. בגלל ההתאמה של תווים למספרים שלמים, גם ערכיו של הטיפוס התווי ניתנים להשוואה, כפי שמדגימה הבעיה הבאה.

בעיה 4

מטרת הבעיה ופתרונה: הצגת פעולות השוואה על טיפוס תווי.

בסדרות של סימנים, הכוללות מספר סופי של סימנים, לכל סימן יש סימן עוקב מלבד לאחרון. עבור סדרות כאלה נהוג להגדיר את הסימן הראשון כ״עוקב מעגלית״ לסימן האחרון.
פתחו וישמו אלגוריתם אשר הקלט שלו הוא אות מן האותיות הגדולות של הא״ב האנגלי (A..Z), והפלט שלו הוא האות העוקבת ״בצורה מעגלית״ והודעה מתאימה. עבור אות קלט השונה מהאות Z יהיה הפלט האות הבאה בא״ב והודעה המכריזה שזו האות העוקבת. עבור האות Z הפלט יהיה האות A והודעה המכריזה על חזרה להתחלה.

פירוק הבעיה לתת-משימות

1. קליטת אות
2. חישוב האות העוקבת "בצורה מעגלית"

3. הצגת האות העוקבת בצירוף הודעה מתאימה

בחירת משתנים

נבחר את המשתנים הבאים מטיפוס תווי:

letter – ישמור את האות הניתנת כקלט

nextLetter – ישמור את האות העוקבת "בצורה מעגלית" לאות הקלט

האלגוריתם

1. *קלוט אות ב-*letter
2. *אם ערכו של* letter *הוא* 'Z'
   - 2.1. *השם את האות* 'A' *ב-*nextLetter
   - 2.2. *הצג כפלט את ההודעה* "Back to start:" *ואת ערכו של* nextLetter
3. *אחרת*
   - 3.1. *חשב את האות העוקבת לאות הנתונה ב-*letter *והשם ב-*nextLetter
   - 3.2. *הצג כפלט את ההודעה* "The next letter is:" *ואת ערכו של* nextLetter

יישום האלגוריתם

כיצד נבצע את ההשוואה בשורה 2 של האלגוריתם? ניתן להשוות בין ערכים מטיפוס תווי, בדיוק כשם שניתן להשוות בין ערכים מטיפוס שלם או ממשי, בשימוש באופרטור ההשוואה הרגיל של השפה. לכן הביטוי הבוליאני המתאים הוא `letter == 'Z'`.

התוכנית המלאה

```
/*
קלט: אות אנגלית גדולה
פלט: הודעה הכוללת את האות העוקבת "בצורה-מעגלית" לאות הנתונה
*/
using System;
public class NextLetterInCircle
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        char letter; // אות הקלט
        char nextLetter; // האות העוקבת
        // קליטת המשתנים
1.      Console.Write("Enter a letter from the ABC: ");
2.      letter = char.Parse(Console.ReadLine());
        // ההוראה לביצוע-בתנאי: חישוב האות העוקבת
3.      if (letter ==  'Z')
        {   // המקרה המיוחד - האות האחרונה
3.1.        nextLetter = 'A';
3.2.        Console.WriteLine("Back to start: {0}", nextLetter);
        } // if
4.      else
        {      // האות איננה האחרונה
4.1.        nextLetter = (char) (letter + 1); //המרת טיפוס (casting)
4.2.        Console.WriteLine("The next letter is: {0}", nextLetter);
        } // else
```

```
    } // Main
} // class NextLetterInCircle
```

מעקב

נעקוב אחר מהלך ביצוע התוכנית `NextLetterInCircle` עבור הקלט 'C':

	המשפט לביצוע	letter	nextLetter	letter=='Z'	פלט
1	`Console.Write("Enter a letter … ");`	?	?		Enter a letter …
2	`letter = char.Parse( Console.ReadLine() );`	'C'	?		
3	`if (letter == 'Z')`	'C'	?	**false**	
4.1	`nextLetter = (char) (letter + 1);`	'C'	'D'		
4.2	`Console.WriteLine( "The next letter is: {0}", nextLetter);`	'C'	'D'		The next letter is: D

סוף פתרון בעיה 4

שאלה 5.18

בנו טבלת מעקב אחר ביצוע התוכנית `NextLetterInCircle` לפתרון בעיה 4 עבור הקלט 'Z'.

כזכור, קבוצת התווים של המחשב כוללת גם את הספרות '9',...,'1','0'. כלומר ניתן להתייחס אל ספרה כאל ערך מטיפוס תווי. ספרות עוקבות מסודרות כתווים עוקבים בקבוצת התווים. השאלה הבאה מתייחסת לספרות כאל תווים.

שאלה 5.19

נתון קטע התוכנית הבא:

```
char digit;
char x;
Console.Write("Enter a digit between 0 and 9: ");
digit = char.Parse(Console.ReadLine());
if (digit == '0')
    x = '9';
else
    x = (char)(digit - 1);
Console.WriteLine(x);
```

הקלט לקטע התוכנית הוא ספרה בין '0' ל-'9'.

א. מהו הפלט עבור הקלט '5', ומהו הפלט עבור הקלט '0'?

ב. מהו הקלט אשר הפלט עבורו יהיה 8?

ג. מהי מטרת קטע התוכנית?

מאחר שערכי הטיפוס התווי ניתנים להשוואה, ניתן להשוות ערכים מטיפוס תווי גם באמצעות הסימנים >, <, =>, =<, ולא רק באמצעות הסימנים == ו-=!.

למשל נניח שערכי המשתנים מטיפוס תווי `let1` ו-`let2` הם 'E' ו-'T' בהתאמה. אז ערכו של כל אחד מן הביטויים הבוליאניים הבאים הוא **true**: `'A' < 'B'`, `'7' > '3'`, `let1 >= 'E'` ו-`let2 <= let1`.

ערכיו של הטיפוס התווי ניתנים להשוואה. לכן ניתן להשתמש בכל **פעולות ההשוואה על תווים** בדומה לשימושן על ערכים מספריים.

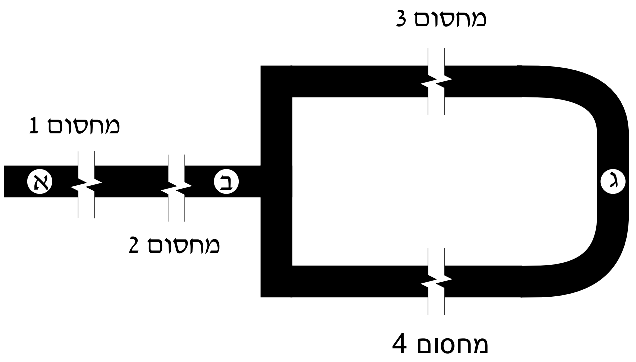
שאלה 5.20

פתחו אלגוריתם אשר הקלט שלו הוא שתי אותיות שונות מן הא״ב האנגלי, והוא מציג את אותיות הקלט פעמיים: בשורה אחת ב״סדר עולה״ ובשורה הבאה ב״סדר יורד״. ״סדר עולה״ פירושו: האות שמופיעה קודם בא״ב האנגלי תוצג משמאל, והאות האחרת תוצג מימינה. ״סדר יורד״ הוא סדר הפוך ל״סדר עולה״. ישמו את האלגוריתם בשפת C#.

להעמקה בתבנית ***סידור ערכים בסדרה*** פנו לסעיף התבניות המופיע בסוף הפרק.

5.2 תנאי מורכב

בסעיף זה נכיר תנאים מורכבים. תנאים מורכבים הם תנאים הבנויים מקישור של תנאים פשוטים יותר. נבחן את השרטוט הבא המתאר מערכת כבישים:



איור 5.1 – מערכת כבישים

במערכת הכבישים המתוארת באיור 5.1 ניתן להגיע מנקודה א לנקודה ג. כדי לעשות זאת יש להגיע מנקודה א לנקודה ב, ומנקודה ב לנקודה ג. על הכבישים נמצאים מחסומים. כדי להגיע מנקודה א לנקודה ב יש לעבור במחסום 1 ובמחסום 2. כלומר, רק אם אפשר לעבור בשני המחסומים, אפשר להגיע מנקודה א לנקודה ב. ניתן לתאר זאת כך:

אם מחסום 1 מורם וגם מחסום 2 מורם
ניתן להגיע מנקודה א לנקודה ב
אחרת
לא ניתן להגיע מנקודה א לנקודה ב

תנאי המעבר מנקודה א לנקודה ב מתואר ב**תנאי מורכב**, שהוא קישור התנאי *מחסום 1 מורם* אל התנאי *מחסום 2 מורם* באמצעות המילה *וגם*.

נתבונן כעת באפשרות להגיע מנקודה ב לנקודה ג. כדי לעשות זאת יש לעבור במחסום 3 או במחסום 4. כלומר, אם אפשר לעבור באחד המחסומים (או בשניהם) אפשר להגיע מנקודה ב לנקודה ג. ניתן לתאר זאת כך:

אם מחסום 3 מורם או מחסום 4 מורם
ניתן להגיע מנקודה ב לנקודה ג
אחרת
לא ניתן להגיע מנקודה ב לנקודה ג

תנאי המעבר מנקודה ב לנקודה ג מתואר על ידי תנאי מורכב, שהוא קישור התנאי *מחסום 3 מורם* אל התנאי *מחסום 4 מורם* באמצעות המילה *או*.

> *וגם* ו-*או* הם **קְשָׁרִים לוגיים**, המאפשרים ליצור מביטויים בוליאניים פשוטים ביטויים בוליאניים מורכבים.

הקֶשֶׁר *וגם*

ראשית נתמקד בקשר הראשון מבין השניים שהודגמו בניתוח של איור 5.1.

בעיה 5

מטרת הבעיה ופתרונה: הצגת תנאי מורכב הכולל את הקשר *וגם*.

> סדרת מספרים נקראת "סדרה עולה ממש" אם ערכו של כל איבר בסדרה קטן ממש מערכו של האיבר הבא אחריו. כלומר ערכו של האיבר הראשון בסדרה קטן ממש מערכו של האיבר השני, ערכו של האיבר השני קטן ממש מערכו של האיבר השלישי, וכן הלאה. למשל סדרת המספרים 10 7 2 1 היא סדרה עולה ממש, וסדרת המספרים 7 3 1 1 איננה סדרה עולה ממש.
> פתחו אלגוריתם אשר הקלט שלו הוא סדרה של שלושה מספרים שלמים, והפלט שלו הוא הודעה האם סדרת המספרים היא סדרה עולה ממש.
> אם הסדרה עולה ממש, יש לצרף להודעה את סדרת ההפרשים שבין איברי הסדרה המקורית. כלומר את ההפרש בין המספר השני לראשון, ואת ההפרש בין המספר השלישי לשני.
> ישמו את האלגוריתם בשפת התכנות #C.

פירוק הבעיה לתת-משימות

1. קליטה של שלושת איברי הסדרה.
2. בדיקה אם הסדרה עולה ממש והצגת הודעה מתאימה.

בחירת משתנים

נבחר שלושה משתנים מטיפוס שלם לשמירת שלושת איברי הסדרה:

num1, **num2**, **num3** – ישמרו את שלושת מספרי הסדרה הנתונה.

האלגוריתם

? בכתיבת האלגוריתם עלינו לנסח תנאי אשר יתקיים כאשר הסדרה הנתונה עולה ממש, ולא יתקיים כאשר הסדרה איננה עולה ממש. מה יהיה התנאי המתאים?

התנאי אשר יתקיים כשהסדרה עולה ממש יהיה:

המספר השלישי גדול מהמספר השני וגם המספר השני גדול מהמספר הראשון

ניתן לכתוב זאת גם כך:

num3>num2 *וגם* num2>num1

> **שימו ♥:** אופן הכתיבה הבא אינו חוקי: num3 > num2 > num1

תנאי זה הוא תנאי מורכב, הכולל קישור באמצעות המילה *וגם* בין שני התנאים.

נציג אלגוריתם לפתרון הבעיה, תוך שימוש בתנאי שניסחנו:

1. קלוט שלושה איברי סדרה ב-num1, ב-num2 וב-num3
2. אם num3>num2 וגם num2>num1
   2.1. הצג כפלט הודעה כי הסדרה עולה ממש
   2.2. הצג כפלט את ערך ההפרש num2-num1 ואת ערך ההפרש num3-num2
3. אחרת
   3.1. הצג כפלט הודעה כי הסדרה אינה עולה ממש

שימו ♥: באלגוריתם מופיעה הוראת פלט שכוללת ערכי הפרשים בין משתנים. לא בחרנו משתנים לשמירת ההפרשים, אלא כללנו בהוראת הפלט ביטויים המבטאים את ההפרשים.

יישום האלגוריתם

ב-#C, הקשר *וגם* נכתב באמצעות הסימן &&.

```
/*
קלט: 3 מספרים שלמים
פלט: הודעה אם סדרת המספרים היא סדרה עולה ממש
*/
using System;
public class CheckSequence
{
    public static void Main ()
    {
        // הצהרה על משתנים בתוכנית
        int num1, num2,num3;
        // קליטת המשתנים
1.      Console.Write("Enter first number: ");
2.      num1 = int.Parse(Console.ReadLine());
3.      Console.Write("Enter second number: ");
4.      num2 = int.Parse(Console.ReadLine());
5.      Console.Write("Enter third number: ");
6.      num3 = int.Parse(Console.ReadLine());
7.      if ((num3 > num2) && (num2 > num1))
        {
            // הסדרה עולה ממש
7.1.        Console.WriteLine("The sequence of numbers is " +
                                                 " strongly increasing");
7.2.        Console.WriteLine("The sequence of differences " +
                         "is {0} {1}", (num2 - num1), (num3 - num2));
        }
8.      else
8.1.        Console.WriteLine("The sequence of numbers is not " +
                                                 " strongly increasing");
    } // Main
} // class CheckSequence
```

מעקב

נעקוב אחר מהלך ביצוע התוכנית עבור הקלט 4 2 1:

מספר שורה	המשפט לביצוע	num1	num2	num3	Num3>num2	num2>num1	פלט
1	`Console.Write("Enter first number: ");`	?	?	?			Enter first number:
2	`num1 = int.Parse( Console.ReadLine());`	1	?	?			
3	`Console.Write("Enter second number: ");`	1	?	?			Enter second number:
4	`num2 = int.Parse( Console.ReadLine());`	1	2	?			
5	`Console.Write("Enter third number: ");`	1	2	?			Enter third number:
6	`num3 = int.Parse( Console.ReadLine());`	1	2	4			
7	`if ((num3 > num2) && (num2 > num1))`	1	2	4	**true**	**true**	
7.1	`Console.WriteLine( "The sequence of numbers is strongly increasing");`	1	2	4			The sequence of numbers is strongly increasing
7.2	`Console.WriteLine("The sequence of differences is ... ");`	1	2	4			The sequence of differences is 1 2

סוף פתרון בעיה 5

תנאי מורכב הוא תנאי המורכב מקישור בין תנאים פשוטים (ביטויים בולאניים פשוטים). עד כה ראינו דרך אחת ליצור תנאי מורכב על ידי קישור תנאים פשוטים: באמצעות הקשר ***וגם*** (and), אשר משמעותו היא שהתנאי המורכב מתקיים רק כאשר שני התנאים הפשוטים מתקיימים ביחד בו-זמנית.

הטבלה הבאה מתארת את ערכיו של ביטוי בוליאני המורכב מקשר ***וגם*** בין שני ביטויים בולאניים. הטבלה נקראת **טבלת אמת של קשר *וגם***:

ביטוי 2 \ ביטוי 1	false	true
false	**false**	**false**
true	**false**	**true**

כפי שניתן לראות בטבלה, ערכו של הביטוי הבוליאני המורכב הוא **true** (אמיתי, נכון) **רק** כאשר ערך שני הביטויים 1 ו-2, הוא **true**. בכל מקרה אחר, ערכו של הביטוי הבוליאני המורכב הוא **false** (שקרי, לא נכון).

הקשר **וגם** מיושם ב-#C בסימן הפעולה `&&`. יש להקיף בסוגריים את הביטויים הבוליאניים המקושרים ב-`&&`.

בטבלת מעקב אחר מהלך ביצוע תוכנית שבה משפט **if** הכולל ביטוי בוליאני מורכב, נקצה עמודה לכל אחד מן הביטויים הבוליאניים הפשוטים המרכיבים את הביטוי הבוליאני המורכב.

שאלה 5.21

בתוכנית `CheckSequence` לפתרון בעיה 5 מופיע הביטוי הבוליאני המורכב הבא:

```
((num3 > num2) && (num2 > num1))
```

השלימו את הטבלה הבאה:

num1	num2	num3	num3 > num2	num2 > num1	(num3 > num2)&& (num2 > num1)	**פלט**
1	2	2				
1	1	2				
1	1	1				
0	1	2				

שאלה 5.22

השלימו את התנאי המורכב המתאים בכל אחת מן ההוראות הבאות:

א. *אם האות הראשונה שווה לרביעית וגם* ____________________
המילה בת ארבע האותיות היא פלינדרום

ב. *אם שנת הלידה שלך גדולה מ-1985 וגם* ____________________
שנת הלידה שלך היא בין 1985 ל-1995

ג. *אם* let>='A' *וגם* ____________________
let *מייצג אות אנגלית גדולה*

שאלה 5.23

כתבו את התנאים הבאים המנוסחים במילים, כביטויים בוליאניים:

א. ערכו של המשתנה `x` גדול מ-0 וקטן מ-50.

ב. ערכו של המשתנה `let` אינו התו 'a' ואינו התו 'z'.

ג. הערך המוחלט של הפרשי ערכי המשתנים `x` ו-`y` גדול מערכו של `x` וקטן מערכו של `y`.

שאלה 5.24

ביטוי בוליאני מורכב יכול לכלול יותר משני ביטויים בוליאניים פשוטים.

במשתנים `temp1`, `temp2` ו-`temp3` ערכים כלשהם.

א. כתבו ביטוי בוליאני המבטא כי ערכי המשתנים `temp1`, `temp2` ו-`temp3` שונים זה מזה. האם נחוצים יותר משני ביטויים בוליאניים פשוטים לכתיבת הביטוי?

ב. כתבו ביטוי בוליאני המבטא כי ערכי המשתנים `temp1`, `temp2` ו-`temp3` שווים זה לזה. האם נחוצים יותר משני ביטויים בוליאניים פשוטים לכתיבת הביטוי?

שאלה 5.25

עבור כל אחד מן הביטויים הבוליאניים המורכבים הבאים, תנו דוגמה לערך של המשתנה `num` או `let` אשר עבורו יהיה ערכו של הביטוי הבוליאני **true**, ותנו דוגמה לערך אשר עבורו יהיה ערכו של הביטוי הבוליאני **false**.

ביטוי	true	false
`(num >= 0) && (num <= 5)`		
`(num > 0) && (num != 1)`		
`(num > 2) && ((num % 2) == 0)`		
`(let != 'z') && (let > 'x')`		

שאלה 5.26

פתחו אלגוריתם אשר הקלט שלו הוא שלושה תווים, והפלט שלו הוא התו העוקב לגדול מבין התווים, אם שלושת התווים הם תווים עוקבים ונתונים בסדר עולה.

למשל עבור הקלט B C D יהיה הפלט E, ועבור הקלט A C D יהיה הפלט ריק (כלומר לא יוצג דבר, כיוון שהתווים אינם תווים עוקבים).

ישמו את האלגוריתם בשפת התכנות C#.

כתבו את חישוב התו העוקב כביטוי במשפט הפלט.

להעמקה בתבנית ***ערכים עוקבים?*** פנו לסעיף התבניות המופיע בסוף הפרק.

הקֶשֶׁר *או*

בניתוח של איור 5.1 הזכרנו קשר נוסף – הקשר *או*. נדון כעת בשימוש בקשר זה בכתיבת אלגוריתמים וביישומם.

בעיה 6

מטרת הבעיה ופתרונה: הצגת תנאי מורכב הכולל את הקשר *או*.

פתחו אלגוריתם שהקלט שלו הוא שני מספרים שלמים חיוביים דו ספרתיים, והפלט שלו הוא הודעה אם שני המספרים מורכבים מאותן ספרות.

למשל, עבור הקלט: 19 91, הפלט יהיה הודעה שהמספרים מורכבים מאותן ספרות, ועבור הקלט 19 81, הפלט יהיה הודעה שהמספרים אינם מורכבים מאותן ספרות.

ישמו את האלגוריתם בשפת התכנות C#.

ניתוח הבעיה בעזרת דוגמאות

נבחן מספר דוגמאות קלט מגוונות אשר כוללות את המספר 91:

עבור כל אחד מן הקלטים: 91 91, 91 19, 19 91 תוצג כפלט הודעה שהמספרים מורכבים מאותן ספרות.

עבור כל קלט אחר הכולל את המספר 91, מלבד שלושת הקלטים האלה, תוצג כפלט הודעה שהמספרים אינם מורכבים מאותן ספרות.

מהתבוננות בשלושת הקלטים המתוארים ניתן לראות שתוצג כפלט הודעה שהמספרים מורכבים מאותן ספרות אם ורק אם המספר הנוסף ל-91 הוא 91 עצמו או שהוא 19, כלומר, המספר המתקבל מ-91 על ידי היפוך סדר הספרות.

פירוק הבעיה לתת-משימות

נוכל לנסח רעיון ראשוני לפתרון: ראשית נפרק את המספר הראשון לספרותיו ונבנה את המספר המתקבל ממנו לאחר היפוך בסדר הספרות. כעת ניתן לבדוק אם המספר השני שווה למספר הראשון או למספר ההפוך לראשון. אם המספר השני שווה לאחד משניהם אז תוצג כפלט הודעה שהמספרים מורכבים מאותן ספרות.

נפרק לתת-משימות על פי הרעיון שהצענו:

1. קליטת שני מספרים שלמים חיוביים דו-ספרתיים
2. פירוק המספר הראשון לספרותיו
3. הרכבת מספר חדש שמתקבל מהמספר הראשון לאחר היפוך סדר הספרות
4. השוואת המספר השני למספר הראשון ולמספר החדש
5. הצגת הודעת פלט מתאימה

בחירת משתנים

נבחר משתנים מטיפוס שלם על פי התת-משימות המתוארות:

num1 – לשמירת המספר הראשון
num2 – לשמירת המספר השני
tens – לשמירת ספרת העשרות של num1
units – לשמירת ספרת האחדות של num1
invNum1 – לשמירת המספר ההפוך בסדר ספרותיו ל-num1

האלגוריתם

? כיצד נבנה את המספר ההפוך למספר הראשון בסדר ספרותיו?

זוהי תבנית ***בניית מספר*** דו-ספרתי, המוכרת לנו מפרק 4: ספרת העשרות של המספר החדש היא ספרת האחדות של המספר הראשון. ספרת האחדות של המספר החדש היא ספרת העשרות של המספר הראשון. לכן יש להכפיל את ערכו של units ב-10 ולחבר למכפלה את ערכו של tens.

? בכתיבת האלגוריתם עלינו לנסח תנאי אשר יתקיים כאשר num2 יהיה שווה לאחד המספרים הנתונים ב-num1 וב-invNum1. מהו התנאי המתאים?

התנאי הוא: num2 == num1 ***או*** num2 == invNum1

תנאי זה הוא תנאי מורכב הכולל קישור בין שני תנאים באמצעות המילה ***או***.

? האם ייתכן שיתקיימו בו-זמנית שני התנאים הפשוטים הכלולים בתנאי המורכב שניסחנו?

כן, זה ייתכן. למשל אם הקלט הוא 11 11. במקרה כזה המספר השני שווה גם למספר הראשון, וגם למספר שמתקבל מהמספר הראשון בהיפוך סדר הספרות. גם במקרה כזה, אנחנו מעוניינים כמובן שהתנאי המורכב יתקיים. כלומר התנאי המורכב מתקיים כאשר num2 שווה לאחד המספרים הנתונים ב-num1 וב-invNum1, או לשניהם.

יישום האלגוריתם

ב-#C, הקשר **או** נכתב באמצעות הסימן || (שני קווים אנכיים רצופים).

```
/*
קלט: 2 מספרים חיוביים שלמים
 פלט: הודעה אם המספרים מורכבים מאותן ספרות
*/
using System;
public class DigitEquality {
      public static void Main()
      {
        // הצהרה על משתנים בתוכנית
        int num1;        // מספר ראשון
        int num2;        // מספר שני
        int tens;        // ספרת העשרות של המספר הראשון
        int units;       // ספרת האחדות של המספר הראשון
        int invNum1;     // המספר השני הפוך
        // קליטת המשתנים
1.      Console.Write("Enter first number: ");
2.      num1 = int.Parse(Console.ReadLine());
3.      Console.Write("Enter second number: ");
4.      num2 = int.Parse(Console.ReadLine());
       //חישוב ספרת האחדות והעשרות וחישוב המספר ההפוך
5.      tens = num1 / 10;
6.      units = num1 % 10;
7.      invNum1 = units * 10 + tens;
        //בדיקה אם המספר השני שווה למספר הראשון או למספר ההפוך לו
8.      if ((num2 == num1) || (num2 == invNum1))
8.1.        Console.WriteLine("The numbers have the same digits");
9.      else
9.1.        Console.WriteLine("The numbers don't have the" +
                                                 " same digits");
    }// Main
}// class DigitEquality
```

סוף פתרון בעיה 6

דרך נוספת ליצור תנאי מורכב היא לקשר תנאים פשוטים באמצעות הקשר **או** (or). משמעותו היא שהתנאי המורכב מתקיים כאשר לפחות אחד משני התנאים הפשוטים מתקיים.

הטבלה הבאה מתארת את ערכיו של ביטוי בוליאני המורכב מקשר **או** בין שני ביטויים בוליאניים. הטבלה נקראת **טבלת אמת של קשר או**:

ביטוי 1 / ביטוי 2	false	true
false	**false**	**true**
true	**true**	**true**

כפי שניתן לראות בטבלה, ערכו של הביטוי הבוליאני המורכב הוא **true** כאשר לפחות ערך אחד משני הביטויים 1 או 2, הוא **true**. רק אם ערכי שני הביטויים הם **false** אז ערכו של הביטוי הבוליאני המורכב הוא **false**.

הקשר *או* מיושם ב-C# בסימן הפעולה `||`. יש להקיף בסוגריים את הביטויים הבוליאניים המקושרים ב-`||`.

שאלה 5.27

בתוכנית `DigitEquality` לפתרון בעיה 6 מופיע הביטוי הבוליאני המורכב הבא:

```
(num1 == num2) || (num2 == invNum1)
```

ציינו עבור כל אחד מן הקלטים הבאים: מהו ערכו של הביטוי הפשוט השמאלי, מהו ערכו של הביטוי הפשוט הימני, מהו ערכו של הביטוי המורכב, ומהו הפלט במהלך ביצוע התוכנית.

קלט	num1 == num2	num2 == invNum1	(num1 == num2) || (invNum1 == num2)	פלט
25 56				
25 25				
25 52				
55 55				

שאלה 5.28

השלימו את התנאי המורכב המתאים בכל אחת מן ההוראות הבאות:

א. *אם האות הראשונה היא* A *או* ________________
לפחות אחת האותיות היא A *במילה בת שתי אותיות*

ב. *אם גילך נמוך מ-18 או* ________________
גילך אינו הגיל המקובל של חייל בשירות סדיר

ג. *אם* num > 10 *או* ________________
num *שומר ערך שהוא גדול מ-10 או קטן מ-5*

שאלה 5.29

כתבו את התנאים הבאים המנוסחים במילים כביטויים בוליאניים.

א. ערכו של המשתנה x חיובי או ערכו של המשתנה y הוא התו 'A'.

ב. ערכו של המשתנה x קטן מ-1 או גדול מ-7.

ג. ערכו של המשתנה x זוגי או מתחלק ב-3 ללא שארית.

להעמקה בתבנית ***זוגיות מספר*** פנו לסעיף התבניות המופיע בסוף הפרק.

להעמקה בתבנית ***מחלק של?*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 5.30

במשתנים side1, side2 ו-side3 שמורים אורכי שלוש צלעות של משולש.

א. כתבו ביטוי בוליאני המבטא שהמשולש שווה שוקיים (במשולש שווה שוקיים לפחות שתי צלעות שוות).

ב. כתבו ביטוי בוליאני המבטא שהמשולש ישר זווית (במשולש ישר זווית סכום ריבועי שני הניצבים שווה לריבוע היתר – משפט פיתגורס).

שאלה 5.31

עבור כל אחד מן הביטויים הבוליאניים המורכבים הבאים, תנו דוגמה לערך של המשתנה num אשר עבורו יהיה ערכו של הביטוי הבוליאני **true**, ותנו דוגמה לערך אשר עבורו יהיה ערכו של הביטוי הבוליאני **false**.

ביטוי	**ערכו של הביטוי false**	**ערכו של הביטוי true**
`num != 0`	num =	num =
`(num < 2)||(num > 2)`	num =	num =
`(num > 0 )||(num == -5)`	num =	num =
`((num%2) == 0)||(num < 0)`	num =	num =

שאלה 5.32

לפעמים ניתן לצמצם ביטוי בוליאני מורכב לביטוי פשוט. בהנחה ש-num מייצג מספר שלם צמצמו כל אחד מן הביטויים המורכבים לביטוי פשוט (כלומר ללא שימוש בקשרים):

א. `(num < -1) || (num > 1)` ________________________

ב. `(num > -1) && (num < 1)` ________________________

שאלה 5.33 (מבגרות 2003)

לפניכם קטע תוכנית:

```
a = int.Parse(Console.ReadLine());
b = int.Parse(Console.ReadLine());
if ( a < b || a < 100 )
    Console.WriteLine("The expression value: true");
else
    Console.WriteLine("The expression value: false");
```

בחרו במספר שייקלט ב-a ובמספר שייקלט ב-b, כך שיתקבל הפלט

```
The expression value: false
```

נמקו את בחירתכם.

שאלה 5.34

פתחו אלגוריתם אשר הקלט שלו הוא תו. האלגוריתם בודק אם התו הוא תו חוקי לניחוש בטופס ספורטוטו (כלומר 1, 2 או x). האלגוריתם מציג הודעה מתאימה כפלט. ישמו את האלגוריתם בשפת C#.

שאלה 5.35

בתחרות קליעת כדור לארגז, קולעים כדור לארגז שאורכו מטר אחד. תחילתו של הארגז היא במרחק 10 מטרים מן הקולע.

יש לפתח אלגוריתם אשר הקלט שלו הוא מרחק נפילת הכדור מהקולע והפלט שלו הוא הודעה אם הכדור נכנס לארגז. אם הכדור לא נכנס לארגז, יש לצרף לפלט גם את המרחק בין מקום נפילת הכדור ל**מרכז** הארגז.

למשל, עבור הקלט 10.3 הפלט הוא: נכנס.

ועבור הקלט 12 הפלט הוא: לא נכנס, החטאת את מרכז הארגז ב-1.5 מ'.

נתחו תחילה את הבעיה (בעזרת דוגמאות קלט מייצגות) ובחרו משתנים. לאחר מכן:

א. נסחו תנאי מורכב באמצעות קשר *וגם* שיתקיים כאשר הכדור נכנס לארגז.

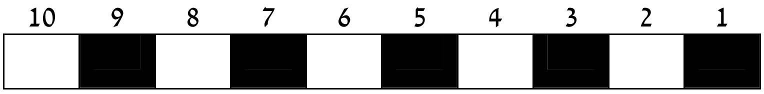
ב. נסחו תנאי מורכב באמצעות קשר *או* שיתקיים כאשר הכדור **לא** נכנס לארגז.

ג. נסחו את התנאי שבסעיף ב כתנאי לא מורכב.

ד. כתבו את האלגוריתם לפתרון הבעיה (בחרו את אחד מהתנאים שבסעיפים א-ג), וישמו את האלגוריתם בתוכנית בשפת C#.

שאלה 5.36

נתון הלוח הבא ובו עשר משבצות הממוספרות מ-1 עד 10:



במשתנה x שמור מספר שלם בין 1 ל-10 המבטא את מקום הכלי X על הלוח. במשתנה y שמור מספר בין 1 ל-10 המבטא את מקום הכלי Y על הלוח.

א. כתבו ביטוי בוליאני שערכו **true** אם הכלי X נמצא בחצי השמאלי של הלוח (כלומר על אחת מחמש המשבצות השמאליות).

ב. השלימו את תיאור המשמעות של קיום התנאי ואת הפלט המתאים במשפט ה-**if** הבא:

```
if (Math.Abs(x - y) == 1) // ______________________________
    Console.WriteLine("____________________________________");
else // ___________________________________________________
    Console.WriteLine("____________________________________");
```

ג. הביטוי הבוליאני הבא אמור לבטא מצבים שבהם הכלי X נמצא מימין לכלי Y על הלוח:

```
x != y
```

הביטוי שגוי. תקנו אותו.

ד. כתבו ביטוי בוליאני פשוט (לא מורכב) שערכו **true** אם הכלי Y נמצא על משבצת שחורה. שימו לב שבביטוי עליכם לבטא את המשותף לחמש המשבצות השחורות.

ה. השלימו את תיאורי המשמעות של קיום התנאי ושל אי-קיומו ואת הפלט המתאים במשפט ה-**if** הבא:

```
if (((x-y) % 2)  == 0) // __________________________
    Console.WriteLine("_____________________________");
else // ____________________________________________
    Console.WriteLine("_____________________________");
```

ו. הביטוי הבוליאני הבא אמור לבטא מצבים שבהם שני הכלים נמצאים על משבצות בצבעים שונים:

```
((x * y) % 2 == 0)
```

הביטוי נכון רק עבור חלק מן המקרים האפשריים. עבור אילו מקרים הביטוי נכון, ועבור אילו מקרים איננו נכון? כתבו ביטוי שיהיה נכון עבור כל המקרים האפשריים.

תנאים מורכבים מעורבים

בעזרת הקשרים וגם ו-או ניתן ליצור ביטויים מורכבים אף יותר מאלו שראינו עד כה, אשר מערבים את שני הקשרים.

שאלה 5.37

עבור כל אחד מן הביטויים הבוליאניים המורכבים הבאים, תנו דוגמה לערך של המשתנה `num` אשר עבורו יהיה ערכו של הביטוי הבוליאני **true**, ותנו דוגמה לערך אשר עבורו יהיה ערכו של הביטוי הבוליאני **false**.

ביטוי	ערכו של הביטוי false	ערכו של הביטוי true
`((num != 0) && (num >= 8)) || (num == 3)`	`num =`	`num =`
`((num<2)||(num>2))&&((num<7)||(num!=1))`	`num =`	`num =`
`(num == 0) && ((num > 0 ) ||(num == -5))`	`num =`	`num =`

בדומה לקדימות המוגדרת ביחס לפעולות חשבוניות (כגון פעולת כפל קודמת לפעולת חיבור), מוגדרת גם **קדימות** ביחס לקשרים בוליאניים: הקשר `&&` קודם לקשר `||`.

בכל זאת, לשם בהירות התוכנית ולשם קריאותה עדיף להשתמש בסוגריים כדי להבהיר את טווח הפעולה של כל קשר.

שאלה 5.38 (מבגרות 2003)

נתון הביטוי הבוליאני: `(z > x) || (x < 0) && (z - y > 9)`

מהו הערך של הביטוי עבור הנתונים: x= -2, y= 5, z= 13? פרטו את כל שלבי החישוב.

5.3 קינון של הוראה לביצוע-בתנאי

בסעיף זה נראה כי לעתים נוח ליצור הוראה מורכבת לביצוע-בתנאי. זוהי הוראה לביצוע-בתנאי שאחת (או יותר) מהוראותיה היא עצמה הוראה לביצוע-בתנאי.

בעיה 7

מטרת הבעיה ופתרונה: הצגת הוראה מקוננת לביצוע-בתנאי.

באולימפיאדת החיות מתקיימת תחרות ריצה למרחק 100 מטר. צבי אשר רץ 100 מטר בזמן של 10 **שניות** או פחות נחשב לצבי מהיר. צב שרץ 100 מטר בזמן של 10 **דקות** או פחות נחשב לצב מהיר.

פתחו אלגוריתם אשר הקלט שלו הוא סוג החיה, T (Turtle) עבור צב ו-D (Deer) עבור צבי, ומספר המציין זמן ריצה בשניות. האלגוריתם יציג הודעה אם החיה שנקלטה מהירה או לא. ישמו את האלגוריתם בשפת C#.

פירוק הבעיה לתת-משימות

1. קליטת שם החיה
2. קליטת תוצאת הריצה בשניות
3. הצגת הודעה אם החיה מהירה או לא

בחירת משתנים

animalType – משתנה מטיפוס תווי שישמור את שם החיה
scoreInSeconds – משתנה מטיפוס שלם שישמור את תוצאת הריצה בשניות

האלגוריתם

? כיצד נחליט אם החיה מהירה?

יש לבדוק תחילה אם החיה היא צבי או צב ואז להשוות את תוצאת הריצה שלה לזמן המגדיר צבי מהיר או צב מהיר, בהתאמה.

נכתוב זאת בצורה הבאה:

אם סוג החיה הוא 'D' *// צבי*
השווה את תוצאת הריצה ל-10 שניות
אחרת // צב
השווה את תוצאת הריצה ל-600 שניות

תיארנו כאן מבנה של *אם...אחרת...* שהוא מבנה של ביצוע-בתנאי. אבל כדי לדעת אם הצבי מהיר ואם הצב מהיר יש לכלול בכל אחד מחלקי ההוראה לביצוע-בתנאי הוראה נוספת לביצוע-בתנאי. המבנה המתקבל הוא של *אם...אחרת...* בתוך *אם...אחרת...*, כלומר קינון של הוראות לביצוע-בתנאי.

נציג אלגוריתם לפתרון הבעיה, תוך שימוש בתנאי שניסחנו:

1. *קלוט את שם החיה ב-*animalType
2. *קלוט את תוצאת הריצה של החיה ב-*scoreInSeconds
3. *אם* animalType == 'D' *// צבי*
3.1. *אם 10* => scoreInSeconds
3.1.1. *הצג הודעה כי הצבי מהיר*
3.2. *אחרת*
3.2.1. *הצג הודעה כי הצבי אינו מהיר*
4. *אחרת // צב*
4.1. *אם 600* => scoreInSeconds
4.1.1. *הצג הודעה כי הצב מהיר*
4.2. *אחרת*
4.2.1. *הצג הודעה כי הצב אינו מהיר*

שימו לב לאופן הזחת השורות (הזזתן פנימה, אינדנטציה) בכתיבה המקוננת של ההוראות לביצוע-בתנאי. אמנם אין השפה מחייבת זאת, אך מומלץ מאוד להקפיד על כך, משום שכך נוח לשייך כל *אחרת* ל-*אם* המתאים. בכך אנו הופכים את התוכנית לקריאה ולברורה יותר.

יישום האלגוריתם

```
/*
קלט: תו המזהה חיה ותוצאת ריצתה למאה מטרים
פלט: הודעה המציינת אם החיה מהירה או לא
*/
using System;
public class AnimalOlympics
{
     public static void Main()
     {
        // הצהרה על משתנים בתוכנית
        char animalType;
        int scoreInSeconds;
        const int DEER_LIMIT = 10;
        const int TURTLE_LIMIT = 600;
        // קליטת המשתנים
1.      Console.Write("Enter the animal type - D for a deer and " +
                                                  "T for a turtle: ");
2.     animalType = char.Parse(Console.ReadLine());
3.     Console.Write("Enter the animal score in seconds: ");
4.     scoreInSeconds = int.Parse(Console.ReadLine());
5.     if (animalType == 'D')   // אם צבי
        {
5.1.        if (scoreInSeconds <= DEER_LIMIT)//  האם מהיר לפי ההגדרה
                                             //  המתאימה לצבי
5.1.1.          Console.WriteLine("The deer is fast");
5.2.        else
5.2.1.          Console.WriteLine("The deer is not fast");
        } // if animalType
6.     else // צב
        {
6.1.        if (scoreInSeconds <= TURTLE_LIMIT) // האם מהיר לפי
                                               // ההגדרה המתאימה לצב
6.1.1.          Console.WriteLine("The turtle is fast");
6.2.        else
6.2.1.          Console.WriteLine("The turtle is not fast");
        } // else animalType
    }//Main
}// class AnimalOlympics
```

שאלה 5.39

בנו טבלת מעקב אחר ביצוע התוכנית `AnimalOlympics` עבור הקלט 700 T ועבור הקלט 5 D.
חשבו לאילו תחומים במבנה המקונן של משפטי ה-**if** בתוכנית, ינותב מהלך הביצוע עבור כל סוג קלט אפשרי?

סוף פתרון בעיה 7

לצורך פתרון בעיה 7 היה מתאים להשתמש במבנה מקונן של הוראות לביצוע-בתנאי.
ההוראה לביצוע-בתנאי שתלויה בסוג החיה (ובתוכנית תלויה במשתנה `animalType`) היא הוראה לביצוע-בתנאי **חיצונית** (בתוכנית היא יושמה במשפט **if** חיצוני). הוראה זו מכילה שתי

הוראות לביצוע-בתנאי **פנימיות** (בתוכנית – שני משפטי **if** פנימיים): האחת מוכלת בתחום ה-*אם* של ההוראה החיצונית והשנייה מוכלת בתחום ה-*אחרת* של ההוראה החיצונית. ההוראות הפנימיות הן אלו שביצוען תלוי בתוצאת הריצה (ובתוכנית תלוי בערך המשתנה `scoreInSeconds`).

לעתים, כאשר יש לנתב את מהלך הביצוע של אלגוריתם לאחת מבין שלוש או יותר אפשרויות התלויות במספר תנאים, מתאים להשתמש במבנה מקונן של הוראות לביצוע-בתנאי.

מבנה מקונן (nesting) של ביצוע-בתנאי (משפטי **if**) כולל הוראה לביצוע-בתנאי, אשר אחת או יותר מבין הוראותיה הפנימיות היא בעצמה הוראה לביצוע-בתנאי. למשל המבנה הבא הוא מבנה מקונן של ביצוע-בתנאי:

בכתיבה אלגוריתמית:	**ב-#C:**
אם...	`if(. . .)`
	`{`
...	`. . .`
אם...	`if (. . .)`
...	`{. . .}`
אחרת	`else`
...	`{. . .}`
...	`. . .`
	`}`
אחרת	`else`
	`{`
...	`. . .`
אם...	`if (. . .)`
...	`{. . .}`
אחרת	`else`
...	`{. . .}`
...	`. . .`
	`}`

בכתיבת מבנה מקונן נקפיד על הזחות מתאימות. באלגוריתם ייכתב כל *אחרת* מתחת ל-*אם* המתאים לו. ב-#C ייכתב כל **else** מתחת ל-**if** המתאים לו.

שימו ♥: ההגדרה של קינון הוראות לביצוע-בתנאי היא כללית למדי, ולכן ייתכנו מבנים מקוננים בצורות שונות. כל אחת מההוראות הפנימיות של הוראה לביצוע-בתנאי יכולה להיות בעצמה הוראה לביצוע-בתנאי. לכן ייתכן למשל משפט **if** שתחום ה-**if** שלו מכיל שתי הוראות לביצוע-בתנאי.

בנוסף, הוראה פנימית לביצוע-בתנאי יכולה להיות הוראה לביצוע-בתנאי מכל סוג שהוא – במבנה *אם*... *אחרת*..., במבנה *אם*... או אפילו הוראה לביצוע-בתנאי מקוננת בעצמה.

לכן מהרגע שקינון מצטרף למשחק, המבנים המתקבלים יכולים להיות מורכבים מאוד. משום כך, חשוב לזכור ולהקפיד על הערות המתארות משמעות של קיום תנאי ושל אי-קיומו. הערות אלו מסייעות לנו להבין מבנה מקונן של הוראות לביצוע-בתנאי. הערה המתארת משמעות של **קיום** תנאי במשפט **if** מבהירה בשפה ברורה וקריאה את הסיבה לניתוב מהלך הביצוע של משפט ה-**if** אל תחום ה-**if** שלו. בדומה, הערה המתארת משמעות של **אי-קיום** תנאי מבהירה בשפה ברורה וקריאה את הסיבה לניתוב מהלך הביצוע של משפט ה-**if** אל תחום ה-**else** שלו.

חשוב לתעד באופן הזה משפטי **if** שאינם מקוננים, כפי שעשינו בתוכניות שהוצגו בפרק עד כה. חשוב עוד יותר לתעד מבנים מקוננים.

הנה שתי דוגמאות למבנים מקוננים שונים: בכל אחד משני המבנים הבאים מחושב יחס (קטן מ, גדול מ, שווה ל) בין ערכי שני משתנים (var1 ו-var2).

מבנה I:

```
if (var1 > var2)
    Console.WriteLine("var1 > var2");
else
{
    if (var1 == var2)
        Console.WriteLine("var1 = var2");
    else
        Console.WriteLine("var1 < var2");
}
```

מבנה II:

```
if (var1 >= var2)
{
    if (var1 == var2)
            Console.WriteLine("var1 = var2");
    else
        Console.WriteLine("var1 > var2");
}
else
    Console.WriteLine("var1 < var2");
```

שני המבנים כוללים הוראה אחת והיא הוראה לביצוע-בתנאי. במבנה I תחום ה-**else** של הוראה זו מכיל הוראה אחת וגם היא הוראה לביצוע-בתנאי במבנה *אם... אחרת...* גם במבנה II תחום ה-**if** מכיל הוראה לביצוע-בתנאי במבנה *אם... אחרת...* .

כאמור, הקפדה על הזחות מתאימות למשפטי **if** בתוכנית בשפת C# מיועדת לשמירה על קריאותה. המהדר של שפת C# אינו מייחס חשיבות להזחות. הוא אינו משייך תחום **else** למשפט ה-**if** המתאים לו על פי ההזחות. השיוך נקבע על פי הכלל הבא, שיודגם מיד:

> ## כלל השיוך של **else** ל-**if** מתאים:
> **else** תמיד משויך ל-**if** אשר נמצא בתוכנית לפניו וקרוב אליו ביותר בתנאים הבאים:
> 1. ל-**if** זה לא משויך **else** אחר קרוב יותר.
> 2. **if** זה איננו כלול בתחום סוגריים מסולסלים אחר שמסתיים עוד לפני ה-**else**.

כלל זה הוא מורכב למדי. הנה כמה דוגמאות שיסייעו בהבנתו:

- במבנה I שלעיל כל **else** משויך ל-**if** שנמצא לפניו וקרוב אליו ביותר.
- במבנה II שלעיל ה-**else** הראשון שייך ל-**if** אשר נמצא בתוכנית לפניו וקרוב אליו ביותר. לעומת זאת ה-**else** השני **אינו** משויך ל-**if** שנמצא לפניו וקרוב אליו ביותר, אלא משויך ל-if הראשון (החיצוני). זאת מכיוון שה-**if** הקרוב אליו ביותר כבר שויך ל-**else** אחר. לכן ה-**else** השני משויך ל-**if** הבא הקרוב ביותר, הלוא הוא ה-**if** הראשון.
- גם במבנה המקונן הבא ה-**else** איננו משויך ל-**if** שנמצא לפניו וקרוב ביותר אליו (ה-**if** השני). הפעם, מכיוון ש-**if** זה כלול בתחום סוגריים מסולסלים שמסתיים עוד לפני ה-**else**. לכן ה-**else** משויך ל-**if** הראשון (החיצוני) ולא ל-**if** השני (הפנימי).

```
if (var1 >= var2)
{
    if (var1 == var2)
        Console.WriteLine("var1 = var2");
}
else
    Console.WriteLine("var1 < var2");
```

שאלה 5.40

בכל אחד מהסעיפים הבאים נתון קטע תוכנית. כמו כן, בכל סעיף נתונות כמה אפשרויות לערכים התחלתיים של משתני הקטע. תארו את הפלטים של הקטעים הבאים, עבור כל אחד מהערכים ההתחלתיים של משתני הקטע.

א.

```
if (num > 0)
    Console.WriteLine("+");
else
{
    if (num == 0)
        Console.WriteLine("0");
    else
        Console.WriteLine("-");
}
```

ערך התחלתי של num	פלט
0	
10	
-10	

ב.

```
if (num1 > 0)
{
    if (num2 > 0)
        Console.WriteLine("+");
    else
        Console.WriteLine("+-");
}
else
    Console.WriteLine("-");
```

ערך התחלתי של num1	ערך התחלתי של num2	פלט
-1	0	
-5	5	
-1	5	

ג.

```
if (num1 < 0)
{
    if (num2 < 0)
        Console.WriteLine(num1 * num2);
}
else
{
    Console.WriteLine(num1);
    if (num2 < 0)
            Console.WriteLine(num1 + num2);
}
```

ערך התחלתי של num1	ערך התחלתי של num2	פלט
-1	1	
-1	-1	
0	0	

שאלה 5.41

הקלט בכל אחד מקטעי התוכניות הבאים הוא אות מן הא״ב האנגלי. מטרת כל אחד מקטעי התוכניות היא להציג כפלט הודעה אם האות הנקלטת היא האות N, אחת האותיות שקודמות ל-N בא״ב האנגלי או אחת האותיות שמופיעות אחרי N בא״ב האנגלי.

השלימו את קטעי התוכניות:

א. קטע תוכנית א

```
Console.Write("Enter a letter: ");
letter = char.Parse(Console.ReadLine());
if (letter == 'N')
        .
        .
        .
```

ב. קטע תוכנית ב

```
Console.Write("Enter a letter: ");
letter = char.Parse(Console.ReadLine());
if (letter != 'N')
        .
        .
        .
```

כפי שמראה התרגיל הבא, ניתן לעתים להמיר הוראה מקוננת לביצוע-בתנאי בהוראה לביצוע-בתנאי שאינה מקוננת אך כוללת תנאים מורכבים. עם זאת במקרים מסוימים השימוש בקינון הופך את התוכנית לקריאה ולברורה יותר.

שאלה 5.42

המירו את הפתרון לבעיה 7 `AnimalOlympics` לפתרון הכולל תנאים מורכבים, ואינו כולל קינון.

שאלה 5.43

קבוצת חייזרים ממאדים או מנוגה תגיע לבקר בכדור הארץ. יש לכתוב אלגוריתם המברך אותם, על פי כללי הטקס המסובכים הנהוגים בחלל.

שמה של קבוצת החייזרים ממאדים הוא תו כלשהו. לעומתה, שמה של קבוצת החייזרים מנוגה הוא מספר כלשהו. פתחו אלגוריתם הקולט מהיכן הגיעה הקבוצה: V (Venus) מנוגה ו-M (Mars) ממאדים. אם הקבוצה היא מנוגה, יש לקלוט את שמה (מספר) ואם הוא גדול מ-10 להציג את ההודעה "hello" ואחרת להציג את ההודעה "hi". אם הקבוצה היא ממאדים יש להציג את ההודעה "have a nice day".

ישמו את האלגוריתם בשפת C#.

האם ניתן לכתוב אלגוריתם לשאלה הנעזר בתנאים מורכבים ואינו משתמש בקינון של הוראות לביצוע-בתנאי, כפי שנעשה בתרגיל 5.42? אם כן, הראו כיצד. אם לא, הסבירו מדוע.

שאלה 5.44

נתון קטע התוכנית הבא, כאשר `let1`, `let2` ו-`let3` הם משתנים מטיפוס תווי השומרים אותיות מן הא״ב האנגלי:

```
if ((let1 == let2) || (let2 == let3) || (let1 == let3))
{
    if ((let1 == let2) && (let2 == let3))
        Console.WriteLine("1");
```

```
    else
        Console.WriteLine("2");
}
else
    Console.WriteLine("3");
```

א. הביאו דוגמת קלט שהפלט עבורה הוא 1, דוגמת קלט שהפלט עבורה הוא 2, ודוגמת קלט שהפלט עבורה הוא 3.

ב. צרפו תיאורי משמעות של קיום התנאים ושל אי-קיומם לתחום ה-**if** ולתחום ה-**else** שבמבנה המקונן.

ג. מהי מטרת קטע התוכנית?

שאלה 5.45

נתונים שני קטעי התוכניות הבאים, ובשניהם num הוא מטיפוס שלם:

א.

```
if(num < 0)
    Console.WriteLine("-1");

if (num == 0)
    Console.WriteLine("0");
else //1___________________
    Console.WriteLine("1");
```

ב.

```
if (num < 0)
    Console.WriteLine("-1");
else
{
    if (num == 0)
        Console.WriteLine("0");
    else //2___________________
        Console.WriteLine("1");
}
```

א. צרפו תיאורי משמעות לאי-קיום תנאי במקומות המסומנים

ב. האם שני קטעי התוכניות שקולים (כלומר, עבור כל קלט נתון יתקבל בשניהם אותו הפלט)? אם כן, הסבירו מדוע. אם לא, הביאו דוגמת קלט אשר עבורה יתקבלו פלטים שונים.

5.4 הוראת שרשרת לביצוע-בתנאי

לעתים נוח לכתוב הוראה לביצוע-בתנאי המתאימה לשרשרת של תנאים, שצריכים להיבדק זה אחר זה. בסעיף זה נתמקד בהוראות כאלו, במבנה *אם... אחרת אם ... אחרת...*

בעיה 8

מטרת הבעיה ופתרונה: הצגת הוראת שרשרת לביצוע-בתנאי במבנה *אם... אחרת אם ... אחרת...*

פתחו וישמו אלגוריתם שהקלט שלו הוא מספר המציין ציון במבחן והפלט שלו הוא הציון המילולי בתעודה, על פי המפתח הבא: כל ציון בין 90 ל-100 במבחן זוכה לציון A בתעודה. כל ציון בין 80 ל-89 במבחן זוכה לציון B בתעודה. כל ציון בין 70 ל-79 במבחן מקבל את הציון C בתעודה. כל ציון בין 60 ל-69 במבחן מקבל את הציון D בתעודה וכל ציון נמוך מ-60 במבחן מקבל את הציון F בתעודה.

פירוק הבעיה לתת-משימות

1. קליטת הציון במבחן
2. חישוב הציון המילולי בתעודה
3. הצגה כפלט של הודעה הכוללת את הציון המילולי בתעודה

בחירת משתנים

score – משתנה מטיפוס שלם לשמירת ציון המבחן

grade – משתנה מטיפוס תווי לשמירת הציון המילולי בתעודה

האלגוריתם

? בכתיבת האלגוריתם עלינו לנסח תנאי אשר יחשב את ציון התעודה המתאים לציון המבחן. מה יהיה התנאי הנחוץ?

הציון המילולי בתעודה מוכתב על ידי כמה תנאים:

1. ציון התעודה הוא A אם מתקיים התנאי: הציון במבחן גדול או שווה ל-90.
2. ציון התעודה הוא B אם מתקיים התנאי: הציון במבחן קטן מ-90 אבל גדול או שווה ל-80.
   זהו תנאי מורכב, שמשמעותו: תנאי 1 לא מתקיים וגם ציון המבחן גדול או שווה ל-80.
3. ציון התעודה הוא C אם מתקיים התנאי: הציון במבחן קטן מ-80 אבל גדול או שווה ל-70.
   זהו תנאי מורכב, שמשמעותו: תנאים 1 ו-2 לא מתקיימים וגם ציון המבחן גדול או שווה-70.
4. ציון התעודה הוא D אם מתקיים התנאי: הציון במבחן קטן מ-70 אבל גדול או שווה ל-60.
   זהו תנאי מורכב, שמשמעותו: תנאים 1, 2 ו-3 לא מתקיימים וגם ציון המבחן גדול או שווה ל-60.
5. ציון התעודה הוא D אם מתקיים התנאי: הציון במבחן קטן מ-60.

נוכל לבטא את ההתניה הזאת בשרשרת מורכבת של משפטי תנאי מקוננים, או בשימוש בתנאים מורכבים (ראו תרגילים בהמשך). לחילופין ניתן להיעזר בהוראה במבנה *אם... אחרת אם... אחרת...* שיוצרת מבנה ברור ופשוט למדי.

נציג אלגוריתם לפתרון הבעיה, שמשתמש בהוראה לביטוי שרשרת התנאים שניסחנו:

1. *קלוט את ציון המבחן ב-*score

2. *אם* score ≥ *90*

  2.1. *השם ב-*grade *את התו* 'A'

3. *אחרת אם* score ≥ *80*

  3.1. *השם ב-*grade *את התו* 'B'

4. *אחרת אם* score ≥ *70*

  4.1. *השם ב-*grade *את התו* 'C'

5. *אחרת אם* score ≥ *60*

  5.1. *השם ב-*grade *את התו* 'D'

6. *אחרת*

  6.1. *השם ב-*grade *את התו* 'F'

יישום האלגוריתם

```
/*
קלט: ציון במבחן, בין 0 ל-100
```

```
פלט: ציון תעודה מילולי מתאים
*/
using System;
public class TermGrade
{
    public static void Main()
    {
         // הצהרה על משתנים בתוכנית
        int score;
        char grade;
        // קליטת המשתנים
1.      Console.Write("Enter test score: ");
2.      score = int.Parse(Console.ReadLine());
3.      if (score >= 90)
3.1.        grade = 'A';
4.      else if (score >= 80)
4.1.        grade = 'B';
5.      else if (score >= 70)
5.1.        grade = 'C';
6.      else if (score >= 60)
6.1.        grade = 'D';
7.      else
7.1.        grade = 'F';
8.      Console.WriteLine("Grade = {0}", grade);
    } // Main
}// class TermGrade
```

מעקב

נעקוב אחר מהלך ביצוע התוכנית TermGrade עבור הקלט 76:

מספר שורה	המשפט לביצוע	score	grade	score>=?	פלט
1	`Console.Write("Enter test score: ");`	?	?		Enter test score:
2	`score = int.Parse(Console.ReadLine());`	76	?		
3	**if** (score >= 90)	76	?	**false**	
4	**else if** (score >= 80)	76	?	**false**	
5	**else if** (score >= 70)	76	?	**true**	
5.1	grade = 'C';	76	C		
8	`Console.WriteLine("Grade = {0}", grade);`				Grade = C

סוף פתרון בעיה 8

בפתרון בעיה 8 מהלך הביצוע של האלגוריתם נקבע על פי שרשרת של תנאים. לצורך כך, השתמשנו במבנה *אם..אחרת אם...אחרת* של הוראות לביצוע-בתנאי.

> **הוראת שרשרת לביצוע-בתנאי** כוללת סדרה של תנאים ושל הוראות לביצוע עבור כל תנאי. משמעות ההוראה היא כי התנאים נבדקים לפי הסדר עד שאחד מהם מתקיים. ברגע שתנאי מסוים מתקיים, מתבצעות ההוראות שהוגדרו עבורו, ושאר התנאים לא נבדקים.

הוראת שרשרת לביצוע-בתנאי (בצד ימין בכתיבה אלגוריתמית ובצד שמאל ב-C#):

```
אם <תנאי 1>
    <קבוצת הוראות 1>
אחרת אם <תנאי 2>
    <קבוצת הוראות 2>
    .
    .
אחרת אם <תנאי k>
    <קבוצת הוראות k>
אחרת
    <קבוצת הוראות k+1>
```

```
if <תנאי 1>
  {. . .}
else if <תנאי 2>
  {. . .}
  .
  .
else if <תנאי k>
  {. . .}
else
  {. . .}
```

שאלה 5.46

פתחו אלגוריתם המחשב את דרגת החוכמה של תוכים, הנקבעת באופן הבא: אם התוכי יודע לומר יותר מ-10 מילים הוא תוכי חכם מאוד. אם התוכי יודע לומר בין 6 ל-10 מילים הוא תוכי חכם. אם התוכי יודע לומר בין מילה אחת ל-5 מילים הוא תוכי ממוצע. אם התוכי לא יודע לומר אף מילה הוא תוכי שתקן.

האלגוריתם מקבל כקלט את מספר המילים השונות שתוכי יודע להגיד, ומציג הודעה המציינת את דרגת החוכמה של התוכי.

ישמו את האלגוריתם בשפת C#.

שאלה 5.47

פתחו אלגוריתם אשר הקלט שלו הוא מקדמים של משוואה ריבועית, a, b ו-c, והפלט הוא מספר הפתרונות הממשיים של המשוואה הריבועית (0 פתרונות, פתרון אחד או 2 פתרונות) והפתרונות עצמם. ישמו את האלגוריתם בשפת C#.

להזכירכם, הנה הנוסחה לחישוב פתרונותיה של משוואה ריבועית: $X_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

שאלה 5.48

נתונים שני קטעי התוכניות הבאים אשר בכל אחד מהם num הוא מטיפוס שלם.

האם שני קטעי התוכנית שקולים (כלומר, עבור כל קלט נתון יתקבל בשניהם אותו הפלט)? אם כן, הסבירו מדוע. אם לא, הביאו דוגמת קלט אשר עבורה יתקבלו פלטים שונים.

א.

```
if (num < 0)
    Console.WriteLine("-1");
else if (num == 0)
    Console.WriteLine("0");
else
    Console.WriteLine("1");
```

ב.

```
if (num < 0)
    Console.WriteLine("-1");
else
{
    if (num == 0)
        Console.WriteLine("0");
    else
        Console.WriteLine("1");
}
```

במובן מסוים, המבנה *אם... אחרת אם... אחרת* אינו חיוני. כפי שמראים התרגילים הבאים ניתן להסתדר גם בלעדיו. אבל השימוש בו מקל לעתים על הכתיבה, ויכול להפוך את התוכנית לקריאה יותר.

שאלה 5.49

המירו את האלגוריתם שבפתרון בעיה 8, לאלגוריתם שמשתמש בהוראה לביצוע-בתנאי במבנה מקונן במקום בהוראת שרשרת לביצוע-בתנאי.

שאלה 5.50

המירו את האלגוריתם שבפתרון בעיה 8, לאלגוריתם שמשתמש בתנאים מורכבים במקום בהוראת שרשרת לביצוע-בתנאי.

5.5 הוראת בחירה

במקרים רבים ערכו של ביטוי מסוים קובע את המשך המשימה. כלומר יש לבצע תת-משימה שונה עבור כל ערך אפשרי לביטוי. בכתיבת אלגוריתם מתאים למשימה כזאת נוח להשתמש בהוראת בחירה שנכיר בסעיף זה.

בעיה 9

מטרת הבעיה ופתרונה: הצגת הוראת בחירה המיושמת ב-#C במשפט **switch**.

> פתחו וישמו אלגוריתם אשר מדמה מחשבון פשוט מאוד. הקלט הוא מספר ממשי, אחריו אחד מסימני הפעולה +, -, * או /, ואחריו מספר ממשי נוסף. הפלט הוא תוצאת החישוב של הפעולה על שני המספרים הנתונים.
>
> למשל, עבור הקלט: -1.1 + 3.4 יהיה הפלט -1.1+3.4=2.3
>
> עבור הקלט: 4.8 / 1.2 יהיה הפלט 4 4.8/1.2=4

פירוק הבעיה לתת-משימות

1. קליטת נתוני הקלט
2. ביצוע החישוב
3. הצגת תוצאת החישוב כפלט.

בחירת משתנים

num1 – משתנה מטיפוס ממשי, לשמירת המספר הראשון הניתן כקלט.

num2 – משתנה מטיפוס ממשי, לשמירת המספר השני הניתן כקלט.

operator – משתנה מטיפוס תווי, לשמירת סימן הפעולה הניתן כקלט.

result – משתנה מטיפוס ממשי, לשמירת תוצאת החישוב.

האלגוריתם

התת-משימה המשמעותית היא התת-משימה השנייה. בתת-משימה זו יש לבצע אחד מארבעה חישובים: חיבור, חיסור, כפל או חילוק, לפי סימן הפעולה הנקלט. כיצד ננסח זאת?

דרך אחת לניסוח החישוב היא באמצעות המבנה המקונן הבא:

אם סימן הפעולה הוא '+'

חבר את שני המספרים

אחרת
אם סימן הפעולה הוא '-'
חסר את המספר השני מן הראשון
אחרת
אם סימן הפעולה הוא '*'
הכפל את שני המספרים
אחרת
חלק את המספר הראשון בשני

דרך אחרת לניסוח החישוב היא באמצעות הוראת שרשרת:

אם סימן הפעולה הוא '+'
חבר את שני המספרים
אחרת אם סימן הפעולה הוא '-'
חסר את המספר השני מן הראשון
אחרת אם סימן הפעולה הוא '*'
הכפל את שני המספרים
אחרת
חלק את המספר הראשון בשני

במקרה זה, שני המבנים הם מסורבלים יחסית. נוח יותר לכתוב הוראה המפרטת חלופות שונות בהתאם לערכו של סימן הפעולה, ועבור כל חלופה מנוסחת הוראת פעולה מתאימה.

ננסח את הפירוט של החלופות השונות באמצעות הוראת הבחירה הבאה:

בצע עבור הערך המתאים לסימן הפעולה:
'+': חבר את שני המספרים
'-': חסר את המספר השני מן הראשון
'*': הכפל את שני המספרים
'/': חלק את המספר הראשון בשני

שימו ♥: הפעולות חיבור, חיסור וכפל ניתנות לביצוע עבור כל זוג מספרים ממשיים, אך בפעולת חילוק יש להיזהר – אין לחלק ב-0. אם המחלק הוא 0, אין לבצע חלוקה, אלא רק להציג כפלט הודעת שגיאה על ניסיון חלוקה ב-0. נכלול זאת באלגוריתם המלא.

יישום האלגוריתם

הוראת בחירה מיושמת ב-C# במשפט **switch**, כפי שניתן לראות בתוכנית המלאה ליישום האלגוריתם שכתבנו:

```
/*
    התוכנית מדמה מחשבון לפעולות +, -, * ו-/
*/
using System;
public class Calculator
{
    public static void Main()
    {
        // הצהרה על משתנים בתוכנית
        double num1; // operand 1
        double num2; // operand 2
```

```
        char operator;     // operator
        double result;
        // קליטת המשתנים
1.      Console.Write("Enter the first number: ");
2.      num1 = double.Parse(Console.ReadLine());
3.      Console.Write("Enter the operator: ");
4.      operator = char.Parse(Console.ReadLine());
5.      Console.Write("Enter the second number: ");
6.      num2 = double.Parse(Console.ReadLine());
7.      switch (operator)
        {
7.1.        case '+':
7.1.1.            result = num1 + num2;
7.1.2.            Console.WriteLine("{0} + {1} = {2}",num1, num2,
                                                           result);
                  break;
7.2.        case '-':
7.2.1.            result = num1 - num2;
7.2.2.            Console.WriteLine("{0} - {1} = {2}",num1, num2,
                                                           result);
                  break;
7.3.        case '*':
7.3.1.            result = num1 * num2;
7.3.2.            Console.WriteLine("{0} * {1} = {2}",num1, num2,
                                                           result);
                  break;
7.4.        case '/':
7.4.1.            if (num2 != 0)
                  {
7.4.1.1.              result = num1 / num2;
7.4.1.2.              Console.WriteLine("{0} / {1} = {2}",num1, num2,
                                                           result);
                  }
7.4.2.            else
7.4.2.1.              Console.WriteLine("Division by 0");
                  break;
7.5.        default: Console.WriteLine("Illegal operator");
                  break;
        } // switch
    } // Main
} // class Calculator
```

אם כך, בהוראת בחירה בוחנים ערך של ביטוי (במקרה זה, הביטוי הפשוט operator), ומשווים אותו לחלופות השונות. ביישום בשפת C# מקדימה המילה **case** את החלופות השונות.

שימו ♥: ה-**case** המתאים לחלופה שנבחרה, מהווה למעשה את נקודת הכניסה למשפט ה-**switch**. ממנו מתחילות להתבצע **כל** ההוראות שבתוך המשפט, זו אחר זו.

בסיום כל **case** שמפורטות בו פעולות לביצוע עלינו להוסיף את ההוראה **break** אשר מורה על יציאה ממשפט ה-**switch** ועל סיום ביצועו. אם ברצוננו לבצע אותן פעולות עבור **case** שונים, ניתן לכתוב את ה-**case** השונים בזה אחר זה, ורק עבור האחרון לכתוב את הפעולות לביצוע, ואחריהן להוסיף את ההוראה **break**. תרגיל 5.52 מדגים זאת.

קבוצת ההוראות הצמודה למילה **default** מתאימה לטיפול בערכים שאינם מטופלים באף חלופה, כלומר כאשר ערכו של הביטוי אינו שווה לאף אחד מהערכים המפורטים במשפט. אין חובה לכלול אפשרות **default** במשפט ה-**switch**. במקרה כזה, אם אין התאמה בין ערך הביטוי לאף אחד מהערכים המפורטים במשפט לא מתבצע דבר. גם בסיום חלק ה-**default** עלינו להוסיף את ההוראה **break**.

המעקב

המעקב אחר ביצוע משפט **switch** דומה למעקב אחר ביצוע משפט **if**. כלומר בשורה המתאימה בטבלת המעקב נרשום את המשפט **switch case:** לביצוע, ובשורה שאחריה נרשום את המשפט הנבחר לביצוע. הנה טבלת המעקב אחר מהלך ביצוע התוכנית Calculator עבור הקלט 2 / 5 :

מספר שורה	המשפט לביצוע	num1	num2	operator	result	פלט
1	Console.Write("Enter the first number: ")	?	?	?	?	Enter the first number
2	num1 = **double**.Parse(Console.ReadLine());	5	?	?	?	
3	Console.Write("Enter the operator: ");	5	?	?	?	Enter the operator
4	op = **char**.Parse(Console.ReadLine());	5	?	'/'	?	
5	Console.Write("Enter the second number: ");	5	?	'/'	?	Enter the second number
6	num1 = **double**.Parse(Console.ReadLine());	5	2	'/'	?	
7	**switch case**: /	5	2	'/'	?	
7.4.1	**if** (num2 != 0)	5	2	'/'	?	
7.4.1.1	result = num1 / num2;	5	2	'/'	2.5	
7.4.1.2	Console.WriteLine("{0} / {1}={2}",num1, num2, result);	5	2	'/'	2.5	5/2=2.5

סוף פתרון בעיה 9

שאלה 5.51

בנו טבלת מעקב אחר ביצוע התוכנית Calculator עבור הקלט 5.3 * 0.

כאשר עלינו לנתב את מהלך הביצוע על פי בחירה באחת מכמה אפשרויות של ערכים בדידים מתאים יותר להשתמש ב**הוראת בחירה** מאשר בהוראת שרשרת לביצוע-בתנאי.

המבנה הכללי של הוראת בחירה הוא:

עבור הערך המתאים של ביטוי בצע:

ערך 1: הוראה-1

ערך 2: הוראה-2

...

ערך k: הוראה-k

בביצוע ההוראה מחושב תחילה ערכו של הביטוי ולפיו מבוצעת החלופה המתאימה (שיכולה להיות הוראה מורכבת בפני עצמה).

הוראת בחירה מיושמת ב-C# ב**משפט switch**

המבנה הכללי של משפט **switch** הוא:

```
switch (ביטוי)
{
    case ערך: משפט; break;
    case ערך: משפט; break;
    .
    .
    .
    default: משפט; break;
}
```

תפקיד **משפט ה-break** הוא לגרום לסיום ביצוע הוראת ה-**switch**.

תפקיד ה-**default** (ברירת מחדל) הוא לספק הוראות שיתבצעו במקרה שערכו של הביטוי אינו שווה לאף אחת מן החלופות. אין חובה לכלול טיפול ברירת מחדל.

שאלה 5.52

בקטע התוכנית הבא יש שימוש ב-**break** רק בחלק מהמקרים:

```
switch (month)
{
    case 1:
    case 3:
    case 5:
    case 7:
    case 8:
    case 10:
    case 12:
        numDays = 31;
        break;
    case 4:
    case 6:
    case 9:
    case 11:
        numDays = 30;
        break;
    case 2:
        if ( ((year%4 == 0) && (year%100 != 0)) || (year%400 == 0) )
            numDays = 29;
        else
            numDays = 28;
        break;
    default:
        Console.WriteLine("Error, Acceptable values for months " +
                                                      "are 1-12");
        break;
} // switch
```

א. מהי מטרת קטע התוכנית?

ב. תנו דוגמה לקלט שעבורו יגיע ביצוע משפט ה-**switch** לחלופת ברירת המחדל.

ג. תנו דוגמה לקלט שעבורו ערכו של המשתנה `numDays` בתום ביצוע משפט ה-**switch** יהיה 28 ודוגמה לערך שעבורו ערכו של המשתנה `numDays` בתום ביצוע משפט ה-**switch** יהיה 29.

שאלה 5.53

פתחו וישמו אלגוריתם אשר הקלט שלו הוא שלושה מספרים שלמים המציינים יום, חודש ושנה של תאריך. הפלט הוא תיאור התאריך בצורה יותר ברורה על ידי הצגת שם החודש, במקום מספרו.

למשל, עבור הקלט 1995 10 12 יהיה הפלט 12 באוקטובר 1995.

שאלה 5.54

במבחן יש 10 שאלות אמריקאיות, והציון עבור כל שאלה הוא 10 נקודות או 0 נקודות. כלומר ציון המבחן יכול להיות רק אחד מאחד עשר הציונים 0, 10, 20, ..., 90, 100.

פתחו אלגוריתם שהקלט שלו הוא ציון במבחן (כמספר שלם), והפלט שלו הוא הציון המילולי המתאים לו: לציון 100 במבחן מתאים הציון A, ל-90 מתאים B, ל-80 מתאים C, ל-70 מתאים D, ולכל ציון 60 ומטה מתאים F.

שאלה 5.55

כתבו תוכנית בשפת C# שתגריל קלף מתוך חפיסת קלפים ותדפיס את צורתו ואת ערכו. ערך של קלף הוא מספר בין 1 ל-13, וצורה של קלף היא: לב, תלתן, מעוין או עלה.

הדרכה: הגרילו מספר בין 1 ל-13 ומספר בין 1 ל-4. לפי התוצאה הציגו את ערך הקלף ואת צורתו (1 – לב, 2 – תלתן, 3 – מעוין, 4 – עלה).

סיכום

בפרק זה ראינו כיצד להורות על מהלכי ביצוע שונים באלגוריתם על פי קיום תנאי או על פי אי-קיומו. הדבר נעשה באמצעות הוראה לביצוע-בתנאי.

הוראה לביצוע-בתנאי מורה על בחירה לפי תנאי בין ביצוע תת-משימה אחת (פשוטה או מורכבת) ובין ביצוע תת-משימה אחרת: *אם... אחרת...*, או מורה על בחירה אם לבצע או לא לבצע תת-משימה: *אם...*.

התנאי המופיע בהוראה לביצוע-בתנאי יכול להיות תנאי פשוט או **תנאי מורכב**. תנאי מורכב בנוי מצירוף של תנאים פשוטים ושל הקשרים *וגם* ו-*או*.

ביטוי בוליאני מבטא תנאי. אם התנאי שמייצג הביטוי הבוליאני מתקיים אז ערכו של הביטוי הבוליאני הוא **true** (אמת). אם התנאי שמייצג הביטוי הבוליאני אינו מתקיים אז ערכו של הביטוי הבוליאני הוא **false** (שקר).

כאשר ביטויים בוליאניים מקושרים בקשר *או* ומרכיבים ביטוי בוליאני חדש, ערכו **true** כאשר **לפחות** אחד מערכי הביטויים הבוליאניים המקושרים הוא **true**. כאשר הם מקושרים בקשר *וגם*, ערכו של הביטוי החדש הוא **true** רק כאשר ערכי **כל** הביטויים הבוליאניים המקושרים הם **true**.

כאשר יש לנתב את מהלך הביצוע של האלגוריתם לבחירה מבין שלוש אפשרויות או יותר, ניתן לעתים להשתמש בהוראות מורכבות לביצוע-בתנאי:

מבנה מקונן של הוראות לביצוע-בתנאי הוא הוראה לביצוע-בתנאי שאחת (או יותר) מהוראותיה הפנימיות היא הוראה לביצוע-בתנאי בעצמה (פשוטה או מורכבת).

הוראת שרשרת לביצוע-בתנאי מתאימה לשרשרת של תנאים, שצריכים להיבדק זה אחר זה. התנאים נבדקים לפי הסדר עד שאחד מהם מתקיים. ברגע שתנאי מסוים מתקיים, מתבצעות ההוראות שהוגדרו בתחומו, ושאר התנאים לא נבדקים.

ב**הוראת בחירה** בוחנים ערך של ביטוי, ומשווים אותו לרשימת ערכים אפשריים. לפי תוצאת ההשוואה מתבצעת קבוצת הוראות.

כאשר מופיעה באלגוריתם הוראה לביצוע-בתנאי יש לבדוק את מהלך האלגוריתם עבור **קלטים מייצגים**, כלומר, לפחות דוגמת קלט אחת שתביא לכך שהתנאי יתקיים ולפחות דוגמת קלט אחת שתביא לכך שהתנאי לא יתקיים.

תיאורי משמעות של קיום תנאים ושל אי-קיומם הם הערות המבהירות את התנאי. הם עוזרים לנו בקריאת תוכנית. יש לצרף תיאורים כאלה לתנאים שכדאי להבהיר את משמעותם.

סיכום מרכיבי שפת C# שנלמדו בפרק 5

הוראה לביצוע-בתנאי במבנה ***אם... אחרת...*** נכתבת בשפת C# במשפט **if** באופן הבא:

```
if (ביטוי בוליאני)
{
    ההוראות אשר מבוצעות אם התנאי מתקיים
}
else
{
    ההוראות אשר מבוצעות אם התנאי אינו מתקיים
}
```

הוראה לביצוע-בתנאי במבנה ***אם...*** נכתבת בשפת C# במשפט **if** באופן הבא:

```
if (ביטוי בוליאני)
{
    ההוראות אשר מבוצעות אם התנאי מתקיים
}
```

ליצירת ביטויים בוליאניים פשוטים ב-C# ניתן להשתמש בכל אחד מ**סימני ההשוואה** של השפה. ניתן להשתמש בסימני ההשוואה כדי להשוות ערכים מכל טיפוס שערכיו ניתנים להשוואה. ניתן להשוות בין תווים.

הקשר ***או*** מסומן ב-C# בסימן הפעולה `||` **והקשר** ***וגם*** מסומן בסימן הפעולה `&&`. עדיפות הקשר `&&` גבוהה מעדיפות הקשר `||` כלומר `&&` קודם ל-`||` בסדר הפעולות. עם זאת מומלץ לסמן בבירור את טווח הפעולה של כל קשר בסוגריים, כדי ליצור ביטויים ברורים וקריאים.

כל אחת מההוראות בתוך משפט **if** ב-C# יכולה להיות בעצמה הוראה לביצוע-בתנאי מכל סוג שהוא. כך מתקבל **קינון** של הוראות לביצוע-בתנאי.

בהוראה מקוננת לביצוע-בתנאי, השיוך של **else** ל-**if** המתאים לו מתבצע על פי הכלל הבא:
else תמיד משויך ל-**if** אשר נמצא בתוכנית לפניו וקרוב אליו ביותר, בתנאים הבאים:

- ל-**if** זה לא משויך כבר **else** אחר קרוב יותר.
- **if** זה איננו כלול בתחום סוגריים מסולסלים אחר שמסתיים עוד לפני ה-**else**.

הוראת שרשרת לביצוע-בתנאי נכתבת בשפת C# באופן הבא:

```
if (ביטוי בוליאני 1)
{
    ההוראות אשר מבוצעות אם ערכו של ביטוי 1 הוא true
}
else if (ביטוי בוליאני 2)
{
    ההוראות אשר מבוצעות אם ערכו של ביטוי 2 הוא true
}
.
.
.
else
{
    ההוראות אשר מבוצעות אם ערכם של כל הביטויים הוא false
}
```

הוראת בחירה נכתבת בשפת C# במשפט **switch** באופן הבא:

```
switch (ביטוי)
{
    case ערך: <קבוצת הוראות> break;
    case ערך: <קבוצת הוראות> break;
    .
    .
    default: <קבוצת הוראות> break;
}
```

ערכו של הביטוי מושווה לכל אחד מהערכים המפורטים בחלופות השונות. **נקודת הכניסה** למשפט **switch** היא המשפט הצמוד לערך שעבורו הצליחה ההשוואה. מנקודת הכניסה מתבצעות כל ההוראות, עד להוראת **break**...

קבוצת ההוראות הצמודה ל-**default** (**ברירת המחדל**) מתבצעת אם ערכו של הביטוי המושווה אינו שווה לאף אחד מהערכים המפורטים. אין חובה לכלול טיפול ברירת מחדל. אם משפט **switch** אינו כולל חלופת ברירת מחדל, ובמהלך הביצוע ערכו של הביטוי המושווה אינו שווה לאף אחד מהערכים המפורטים, לא תבוצע אף הוראה.

שאלות נוספות

שאלות נוספות לסעיף 5.1

1. נתונים שני קטעי התוכניות הבאים, אשר המשתנים בהם הם מטיפוס שלם:

קטע תוכנית 1:

```
if (num < 0)
    num = -num;
Console.WriteLine(num);
```

קטע תוכנית 2:

```
if (num1 > num2 )
    diff = num1 - num2;
else
    diff = num2 - num1;
Console.WriteLine(diff);
```

א. עבור כל קטע תוכנית, בחרו שתי דוגמאות קלט כך שעבור האחת יתקיים התנאי המופיע בקטע ועבור השנייה לא יתקיים התנאי.

ב. מהי המטרה של כל קטע תוכנית?

ג. עבור כל קטע תוכנית, כתבו קטע תוכנית אחר ללא משפט **if**, המשיג את אותה המטרה.

2. התעריף לתשלום עבור צריכת גז ביתי לחימום הוא: 10 שקלים עבור כל אחד מחמשת הליטרים הראשונים הנצרכים ו-7 שקלים עבור כל ליטר נוסף.
נתון קטע תוכנית אשר הקלט שלו הוא כמות ליטרים שנצרכה הנתונה כמספר ממשי, והפלט שלו הוא התשלום הכולל והתשלום הממוצע עבור ליטר נצרך.
משתני קטע התוכנית הם מטיפוס ממשי: consumption מייצג את כמות הליטרים שנצרכה, payment מייצג את התשלום הכולל ו-average מייצג את התשלום הממוצע עבור ליטר נצרך.

```
if (consumption <= 5)
{
    השלימו
}
else
{
    השלימו
}
```

3. אחד השימושים של מחשב הוא הצפנת הודעות. דרך אחת להצפנת הודעות היא באמצעות החלפת תווים בתווים אחרים, ובאמצעות "ריפוד" בתווים נוספים.
פתחו וישמו אלגוריתם אשר הקלט שלו הוא זוג אותיות שונות מהא"ב האנגלי, והפלט שלו הוא שלושה תווים לפי החוקיות הבאה: אם אות הקלט הראשונה מופיעה בסדר האלף-בית האנגלי לפני אות הקלט השנייה אז הפלט הוא האות העוקבת באלף-בית לאות הקלט הראשונה, לאחריה הסימן '+' ולבסוף האות הקודמת באלף-בית לאות הקלט השנייה. אחרת הפלט הוא האות הקודמת באלף-בית לאות הקלט הראשונה, לאחריה הסימן '-' ולבסוף האות העוקבת באלף-בית לאות הקלט השנייה.
למשל, עבור הקלט ac יהיה הפלט b+b ועבור הקלט da יהיה הפלט c-b.

שאלות נוספות לסעיף 5.2

1. מהירות הנסיעה המותרת בכביש מהיר היא 55 קמ"ש לכל הפחות ו-100 קמ"ש לכל היותר. פתחו אלגוריתם אשר הקלט שלו הוא מהירות נסיעה של מכונית, והפלט שלו הוא הודעה אם נהג המכונית חרג מגבולות המהירות המותרת. ישמו את האלגוריתם בשפת C#.

2. פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי קטן מ-100 והפלט שלו הוא המילה **בום** אם המספר הנתון מתחלק ב-7 או כולל את הספרה 7.

שאלות נוספות לסעיפים 5.3, 5.4 ו- 5.5

1. פתחו אלגוריתם הקולט תו. אם התו שווה ל-'m', האלגוריתם קורא מהקלט שני מספרים שלמים, ומציג כפלט את המספר הגדול מביניהם. אם תו הקלט שווה ל-'n', האלגוריתם קורא תו נוסף. אם התו הנוסף שווה ל-'o', האלגוריתם מציג את ההודעה תחשוב חיובי, אחרת הוא קולט מספר ומציג כפלט את ההודעה אתה המספר שנקלט ואת המספר עצמו. ישמו את האלגוריתם בשפת C#.

2. בחברת ההיי-טק "הייטקי" יש 10 דרגות לעובדים. כל העובדים שהם מעל לדרגה 7 הם מנהלים, כל העובדים מעל לדרגה 4 הם ראשי צוותים, כל העובדים מעל לדרגה 2 הם עובדים קבועים וכל השאר הם סטודנטים. פתחו אלגוריתם (לשימוש מחלקת משאבי האנוש של החברה) הקולט את דרגת העובד ומציג כפלט את תיאורו של העובד (מנהל, ראש צוות, עובד קבוע או סטודנט). ישמו את האלגוריתם בשפת #C.

3. עזרו לזברה לדעת איזה יום היום. פתחו אלגוריתם הקולט מספר בין 1 ל-7 ומציג את השם של היום המתאים (...Sunday, Monday). ישמו את האלגוריתם בשפת #C.

שאלות מסכמות לפרק 5

1. בכל המשפטים שלהלן x הוא משתנה מטיפוס ממשי. כתבו עבור כל זוג ממשפטי ה-**if** הנתונים משפט **if** יחיד שקול, כלומר שביצועו שקול לביצוע שני משפטי ה-**if**, בזה אחר זה:

א.

```
if (x <= 9)
    x = x - 1;
if (x > 9)
    x = x + 1;
```

ב.

```
if (x >= 5)
    x = x * 4;
if (x >= 20)
    x = x / 2;
```

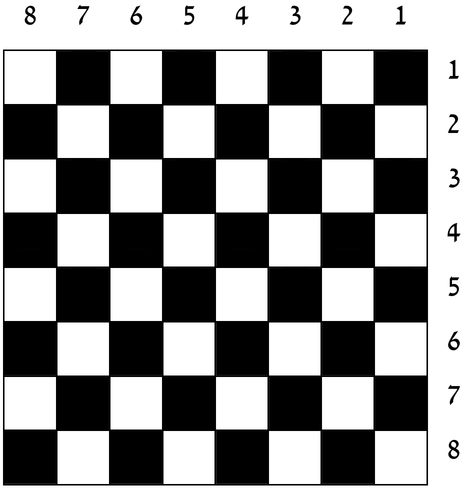
ג.

```
if (x == -1)
    x = 1;
if (x > 0)
    x = -x;
```

ד.

```
if (x < -1)
    x = -x;
if (x > 1)
    x = -x;
```

2. על לוח שחמט מוצבים שני כלים בלבד – צריח ורץ
השורות בלוח ממוספרות מ-1 עד 8, וגם העמודות ממוספרות מ-1 עד 8:



צריח (Rook באנגלית) מאיים על כלי הנמצא עמו באותה שורה או באותה עמודה.
רץ (Bishop באנגלית) מאיים על כלי הנמצא עמו על אותו אלכסון.
במשתנים rookRow ו-rookCol שמורים מספרי השורה והעמודה שהצריח מוצב עליהן.
במשתנים bishopRow ו-bishopCol שמורים מספרי השורה והעמודה שהרץ מוצב עליהן.
א. השלימו את תיאור משמעות קיום התנאי במשפט **if** הבא:

```
if((rookRow + rookCol) % 2) == 1)
      //_________________________________
```

```
Console.WriteLine("The rook is on …");
```

ב. כתבו ביטוי בוליאני שערכו **true** אם הצריח מאיים על הרץ.

ג. הביטוי הבוליאני הבא אמור לבטא מצב לוח שהרץ מאיים על הצריח:

```
(bishopRow - rookRow) == (bishopCol - rookCol)
```

הביטוי כולל רק חלק מן המקרים האפשריים. מהם המקרים הנכללים בו? מהם המקרים שאינם נכללים בו?

הרחיבו את הביטוי כך שיכלול את כל המקרים האפשריים ורק אותם.

תבניות – פרק 5

פירוט מלא של התבניות ושל שאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

מציאת מקסימום ומינימום בסדרה

שם התבנית: **מציאת מקסימום בסדרה** נקודת מוצא: שני ערכים במשתנים element1 ו-element2 מטרה: מציאת הערך הגדול ביותר מבין שני הערכים אלגוריתם: 1. *השם ב-*max *את הערך של* element1 2. *אם* element2 > max 2.1 *השם ב-*max *את הערך של* element2

שם התבנית: **מציאת מינימום בסדרה** נקודת מוצא: שני ערכים במשתנים element1 ו-element2 מטרה: מציאת הערך הקטן ביותר בין שני הערכים אלגוריתם: 1. *השם ב-*min *את הערך של* element1 2. *אם* element2 < min 2.1 *השם ב-*min *את הערך של* element2

סידור ערכים בסדרה

שם התבנית: **סידור ערכים בסדר עולה בסדרה**

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: השמת הערך הקטן יותר ב-element1 והערך הגדול יותר ב-element2

אלגוריתם:

1. *אם* element1 > element2

1.2 החלף את ערכי element1 *ו-*element2

שם התבנית: **סידור ערכים בסדר יורד בסדרה**

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: השמת הערך הגדול יותר ב-element1 והערך הקטן יותר ב-element2

אלגוריתם:

1. *אם* element1 < element2

1.2 החלף את ערכי element1 *ו-*element2

ערכים עוקבים

שם התבנית: **ערכים עוקבים?**

נקודת מוצא: שני ערכים element1 ו-element2

מטרה: קביעת הערך **true** אם element2 עוקב ל-element1, וקביעת הערך **false** אם element2 אינו עוקב ל-element1

אלגוריתם (ביטוי בוליאני):

(האם) element1 + 1 = element2

זוגיות מספר

שם התבנית: **מספר זוגי?**

נקודת מוצא: מספר שלם num

מטרה: קביעת הערך **true** אם num זוגי וקביעת הערך **false** אם num אי-זוגי

אלגוריתם (ביטוי בוליאני):

(האם) *שארית החלוקה של* num *פריטים ל-2 קבוצות שווה ל-0*

שם התבנית: **מספר אי-זוגי?**
נקודת מוצא: מספר שלם num
מטרה: קביעת הערך **true** אם num אי-זוגי וקביעת הערך **false** אם num זוגי
אלגוריתם (ביטוי בוליאני):
(האם) *שארית החלוקה של* num *פריטים ל-2 קבוצות שווה ל-1*

מחלק של מספר

שם התבנית: **מחלק של?**
נקודת מוצא: שני מספרים שלמים num1 ו-num2
מטרה: קביעת הערך **true** אם num2 מחלק את num1 וקביעת הערך **false** אם num2 אינו מחלק את num1
אלגוריתם (ביטוי בוליאני):
(האם) *שארית החלוקה של* num1 *פריטים ל-*num2 *קבוצות שווה ל-0*

פרק 6 – נכונות אלגוריתמים

לאורך ההיסטוריה הקצרה של מדעי המחשב יש אינספור דוגמאות של תוכניות שגויות, ולא מעט מהן הסתיימו בכי רע. אחת הדוגמאות היא סיפורה של ספינת חלל אמריקנית מסדרת מרינר שנשלחה אל כוכב הלכת ונוס. הספינה אבדה כתוצאה משגיאה בתוכנית המחשב שהופקדה על בקרת הטיסה, ומיליוני דולרים ירדו לטמיון.

חלק לא מבוטל מן המחקר במדעי המחשב מוקדש לתחום הנקרא "הוכחת נכונות של תוכניות". מטרתה של הוכחת נכונות של תוכנית היא אימות מלא לטענה שהתוכנית מציגה עבור **כל** קלט את הפלט הדרוש.

חוקרים פיתחו ומפתחים שיטות שונות להוכחת נכונות של תוכניות. שיטות אלה הן מתמטיות באופיין ונשענות על תיאוריות מתמטיות מורכבות למדי. חומר הלימוד של "יסודות מדעי המחשב" איננו דן בהוכחת נכונות של תוכניות, כיוון שהדבר מחייב ידע מתמטי שאיננו נרכש בבית-ספר תיכון. אך ההתייחסות לנכונות של תוכניות חשובה לאורך חומר הלימוד כולו, כבר עם פיתוח תוכניות ראשונות. לכן, אנו מציגים נושא זה כבר עתה, בצורה פשוטה ואינטואיטיבית.

אלגוריתם לפתרון בעיה אלגוריתמית נתונה הוא **נכון** אם ביצועו מביא להצגת הפלט הדרוש עבור **כל** קלט חוקי (כלומר כל קלט המתאים להגדרת הבעיה).

מספר האפשרויות השונות לקלטים חוקיים הוא בדרך כלל רב, ולעתים רבות אפילו אינסופי, ולכן לא מתקבל על הדעת להיווכח בנכונותו של אלגוריתם על ידי בדיקת הפלט עבור כל קלט אפשרי. בפרקי הלימוד של "יסודות מדעי המחשב" אנו בודקים את נכונותו של אלגוריתם על ידי בדיקת הפלט עבור דוגמאות שונות של קלט.

בפרק 4 הצגנו בדיקת נכונות עבור **דוגמאות קלט מגוונות**; כלומר דוגמאות קלט אשר מאפייניהן מבטאים את מגוון הקלטים האפשריים. עתה נחדד בנקודה של בדיקת נכונות, ונשתמש בדרך כלל במושג **דוגמאות קלט מייצגות**.

דוגמאות קלט מייצגות הן דוגמאות קלט אשר כל אחת מהן מייצגת קבוצת קלטים. **בחירה ממצה** של דוגמאות קלט מייצגות היא בחירה המבטאת חלוקה ממצה של הקלטים האפשריים לקבוצות מיוצגות.

בפתרון הבעיה הבאה נמחיש בחירה ממצה של דוגמאות קלט מייצגות, ונראה שימוש בדוגמאות הקלט המייצגות כדי לזהות שגיאה בתוכנית ולתקנה.

בעיה 1

מטרת הבעיה ופתרונה: הדגמה של חלוקת הקלטים לקבוצות אשר מאפייניהן שונים, בחירת דוגמת קלט מייצגת לכל קבוצה, ושימוש בדוגמאות הקלט המייצגות כדי לאמת את נכונותה של תוכנית וכדי לתקנה במידת הצורך.

נתונה התוכנית הבאה:

```
/*
קלט: שתי אותיות אנגליות גדולות
פלט: הודעה אם האותיות עוקבות זו לזו
```

```
*/
using System;
public class Letters
{
    public static void Main()
    {
       char letter1, letter2;
       Console.Write("Enter a capital letter: ");
       letter1 = char.Parse(Console.ReadLine());
       Console.Write("Enter another capital letter: ");
       letter2 = char.Parse(Console.ReadLine());
       if (letter2 == letter1 + 1)
            Console.WriteLine("The letters are consecutive");
       else
            Console.WriteLine("The letters are not consecutive");
    }// Main
} // class Letters
```

ביצוע התוכנית יביא להצגת הודעת פלט נכונה רק עבור חלק מן הקלטים האפשריים.

חלקו את הקלטים האפשריים לקבוצות, בחרו דוגמת קלט מייצגת לכל קבוצה, ותארו עבור אילו דוגמאות תוצג הודעה נכונה ועבור אילו דוגמאות תוצג הודעה שגויה. אחר-כך, תקנו את התוכנית כך שעבור כל קלט תוצג הודעה נכונה.

? מכותרת התוכנית וממשפטי הפלט ניתן לראות שמטרת התוכנית היא לבדוק אם שתי האותיות הנתונות כקלט הן אותיות אנגליות עוקבות. מהי חלוקה מתאימה של הקלטים האפשריים לקבוצות?

מתאים לחלק את הקלטים האפשריים לשתי הקבוצות הבאות:

1. קלטים שהאותיות הנתונות בהם הן אותיות אנגליות עוקבות.
2. קלטים שהאותיות הנתונות בהם אינן אותיות אנגליות עוקבות.

? החלוקה המתוארת אכן מבטאת אבחנה בין קלטים אשר יש להם מאפיינים שונים. האם חלוקה זו מספיקה? האם לא קיימים קלטים בעלי מאפיינים שונים בקבוצות אלו?

למשל, הקלט B C שייך לקבוצה הראשונה, שהאותיות בה הן אנגליות עוקבות. גם הקלט C B שייך לקבוצה הראשונה, אך לשני קלטים אלה מאפיינים שונים. בקלט B C האות השנייה עוקבת לאות הראשונה, ואילו בקלט C B האות הראשונה עוקבת לאות השנייה.

שימו ♥: ניתן גם לחלק את קבוצת הקלטים השנייה באופן דומה, אך חלוקה זו איננה נחוצה: כיוון שהאותיות אינן עוקבות, הרי סדר הופעתן בקלט איננו משנה.

אם כך, נקבל את החלוקה הבאה של הקלטים האפשריים לשלוש קבוצות:

1א. האות השנייה עוקבת לאות הראשונה.

1ב. האות הראשונה עוקבת לאות השנייה.

2. שתי האותיות אינן עוקבות.

? חילקנו את הקלטים האפשריים חלוקה ממצה לקבוצות. מה תהיה בחירה ממצה של דוגמאות קלט מייצגות עבור הבעיה הנתונה?

בחירה ממצה של דוגמאות קלט מייצגות תהיה בחירת דוגמת קלט מכל קבוצה. למשל, קבוצת דוגמאות הקלט הבאות מבטאת בחירה ממצה: X Y (מייצגת את קבוצת הקלטים 1א); E D (מייצגת את קבוצת הקלטים 1ב); D O (מייצגת את קבוצת הקלטים 2)

? עבור אלו מדוגמאות הקלט המתוארות תוצג הודעת פלט נכונה?

עבור דוגמת הקלט הראשונה (X Y) והשלישית (D O) תוצג הודעת פלט נכונה. אך עבור דוגמת הקלט השנייה (E D) תוצג הודעת פלט שגויה! עבור דוגמה זו תוצג ההודעה:

```
The letters are not consecutive
```

הודעה שגויה תוצג, בעצם, עבור כל קלט מהקבוצה 1ב, המיוצגת בדוגמת הקלט השנייה.

? כיצד ניתן לתקן את התוכנית, כך שתציג פלט דרוש עבור **כל** קלט חוקי? כלומר, כך שגם עבור קלטים מן הקבוצה השנייה תוצג הודעה מתאימה?

יש להרחיב את הביטוי הבוליאני שבמשפט ה-**if**. הביטוי הבוליאני שבתוכנית הנתונה כולל רק את האפשרות שהאות השנייה בקלט היא אות עוקבת לאות הראשונה בקלט. יש להרחיב ביטוי זה כך שיכלול גם את האפשרות שהאות הראשונה בקלט היא אות עוקבת לאות השנייה.

לכן, הביטוי הבוליאני שבתוכנית המתוקנת יהיה:

```
(letter2 == letter1 + 1) || (letter1 == letter2 + 1)
```

התוכנית המתוקנת תהיה:

```
/*
קלט: שתי אותיות אנגליות גדולות
פלט: הודעה אם האותיות עוקבות
*/
using System;
public class Letters
{
    public static void Main()
    {
        char letter1, letter2;
        // קלט
        Console.Write("Enter a capital letter: ");
        letter1 = char.Parse(Console.ReadLine());
        Console.Write("Enter another capital letter: ");
        letter2 = char.Parse(Console.ReadLine());
        if ( (letter2 == letter1 + 1) || (letter1 == letter2 + 1) )
            Console.WriteLine("The letters are consecutive");
        else
            Console.WriteLine("The letters are not consecutive");
    }// Main
} // class Letters
```

סוף פתרון בעיה 1

ניתן ללמוד לקח מן הפתרון לבעיה 1:
השגיאה שהופיעה בתוכנית הנתונה נובעת מכך שמפתח התוכנית לא ביצע ניתוח מלא של כל אפשרויות הקלט השונות. מפתח התוכנית לא הבחין בכך שהמשמעות של אותיות אנגליות עוקבות, איננה רק האפשרות ש"האות השנייה עוקבת לאות הראשונה", אלא גם האפשרות ש"האות הראשונה עוקבת לאות השנייה". זיהוי השגיאה ותיקונה התאפשר הודות לחלוקה ממצה של הקלטים האפשריים לקבוצות, והיא הביאה לבחירה ממצה של דוגמאות קלט מייצגות.

שימו ♥: הבדיקה בעזרת דוגמאות קלט מייצגות אינה **מוכיחה** נכונות, אלא מסייעת באיתור שגיאות, ומקטינה מאוד את ההסתברות לטעות, אך תמיד ייתכן (בייחוד בתוכניות גדולות מאוד) שנפספס תת-מקרה מסוים, משום שאין בידינו מתכון לקביעת חלוקה ממצה לקבוצות.

כיום בכל חברת תוכנה יש צוות בודקים אשר כל תפקידו הוא לוודא כי התוכנה עובדת כשורה עבור כל קלט אפשרי. חשיבות בדיקות אלה גבוהה ונועדה על מנת להימנע מהפסדים ולעתים גם מנזקים משמעותיים הרבה יותר, כמו פגיעה בחיי אדם (למשל בתוכנות רפואיות).

במרבית חברות התוכנה מערך הבדיקות מבוסס על סימולציה, כלומר על בחירה מתוחכמת של דוגמאות קלט מייצגות ובדיקת התוכניות עליהן, בחירה שיכולה להיות מורכבת מאוד ומסובכת מאוד בתוכניות גדולות ומורכבות. עם זאת, יש גם חברות (בעיקר חברות לתכנון ולפיתוח חומרה) המשתמשות בכלֵי **הוכחה** (הנשענים על תורות מתמטיות). הסיבה שממעטים להשתמש בכלי ההוכחה היא שלכלים כאלה הקיימים כיום קשה להתמודד בצורה נוחה עם תוכניות גדולות. הסיבה שמשתמשים בהם יותר בתעשיית החומרה היא שהרבה יותר יקר לייצר מחדש רכיב חומרה אם מתגלית בו טעות אחרי שלב הייצור, מאשר לתקן ולהפיץ גרסה חדשה של תוכנה שהתגלתה בה טעות.

שאלה 6.1
נתון הלוח הבא, ובכל משבצת בו מופיע מספר שלם:

2	1
4	3

קטע התוכנית הבא, אשר הקלט שלו הוא שניים מן המספרים המופיעים בלוח, יציג כפלט הודעה.

```
Console.Write("Enter first number: ");
num1 = int.Parse(Console.ReadLine());
Console.Write("Enter second number: ");
num2 = int.Parse(Console.ReadLine());
if (num1 == num2 + 2)
    Console.WriteLine("The numbers are in the same column");
else
    Console.WriteLine("The numbers are not in the same column");
```

א. בחרו שתי דוגמאות קלט, אשר עבור כל אחת מהן תוצג הודעה נכונה אחרת. תארו את המאפיינים של שתי קבוצות הקלטים שהדוגמאות שבחרתם שייכות אליהן.
ב. בחרו דוגמת קלט שעבורה תוצג הודעה שגויה, ותארו את המאפיין של קבוצת הקלטים שעבורם תוצג הודעה שגויה.
ג. שנו את הביטוי הבוליאני שבקטע התוכנית לביטוי בוליאני אחר, שעבורו תוצג תמיד הודעה מתאימה. נסחו את הביטוי החדש כביטוי פשוט.

שאלה 6.2

הקלט לקטע התוכנית הבא הוא מספר שלם כלשהו. מטרת קטע התוכנית הבא היא להציג כפלט ערך שסימָנו כסימן המספר שנקלט וגודלו הוא ריבוע המספר שנקלט. למשל עבור הקלט 10 הפלט הנדרש הוא 100, ועבור הקלט 10- הפלט הנדרש הוא 100-.

שימו ♥: בקטע התוכנית נעשה שימוש בפעולת החזקה `Pow` של המחלקה המתמטית `Math`. הפעולות המתמטיות הוצגו בפרק 4.

```
num = int.Parse(Console.ReadLine());
s = Math.Pow(num, 2);
Console.WriteLine(s);
```

קטע התוכנית שגוי.

א. תנו דוגמת קלט מייצגת שעבורה יתקבל הפלט הדרוש.

ב. תנו דוגמת קלט מייצגת שעבורה לא יתקבל הפלט הדרוש.

ג. תארו את קבוצת הקלטים שעבורם לא יתקבל הפלט הדרוש.

ד. תקנו את קטע התוכנית, כך שעבור כל קלט יתקבל הפלט הדרוש.

שאלה 6.3

נתון קטע התוכנית הבא אשר הקלט שלו הוא ארבע אותיות אנגליות גדולות.

```
letter1 = char.Parse(Console.ReadLine());
letter2 = char.Parse(Console.ReadLine());
letter3 = char.Parse(Console.ReadLine());
letter4 = char.Parse(Console.ReadLine());
if ((letter1 == letter2) && (letter3 == letter4))
    Console.WriteLine("All letters are equal");
else
    Console.WriteLine("Not all letters are equal");
```

א. תנו שתי דוגמאות קלט שונות אשר עבור כל אחת מהן תתקבל הודעה שגויה.

ב. תארו את קבוצת הקלטים שעבורם תתקבל הודעה שגויה.

ג. שנו את הביטוי הבוליאני שבקטע התוכנית כך שעבור כל קלט תתקבל הודעה נכונה.

שאלה 6.4

לפניכם תוכנית בשפת C#. התוכנית קולטת תו כקלט. מטרת התוכנית היא לבדוק האם התו מייצג מספר. אם כן, יוצג המספר הבא אחריו כפלט. אחרת תוצג הודעה כי התו אינו מספר.

הראו **שלוש** דוגמאות קלט מייצגות לתוכנית זו והסבירו עבור כל דוגמה איזו קבוצה היא מייצגת. מצאו את השגיאה בתוכנית ותקנו אותה.

```
/*
התוכנית קולטת תו
אם התו מייצג מספר, התוכנית מציגה את המספר הבא אחריו
בכל מקרה אחר מוצגת ההודעה "לא מספר"
*/
using System;
public class NextNumber
{
    public static void Main ()
    {
        // הגדרת משתנים
        char character;  //תו הקלט
        // קלט
```

```
        Console.Write("Enter a character: ");
        character = char.Parse(Console.ReadLine());
        // פלט
        if (character >= '0' && character <= '9')
            Console.WriteLine( (char)(character + 1) );
        else
            Console.WriteLine("Not a number");
    } // Main
} // class NextNumber
```

ראינו עד כה ניתוח של תוכנית נתונה שמטרתה ברורה. אך בפיתוח תוכנה קורה לא פעם שיש לשלב תוכניות אשר אינן מתועדות כראוי ומטרתן איננה ברורה. במקרים כאלו יש לזהות קודם כל את מטרת התוכנית הנתונה.

זיהוי מטרת תוכנית נתונה מתבצע על ידי כך שנבחן את פלט התוכנית עבור דוגמאות קלט שונות, ונקבע את היחס בין הקלט לפלט.

נדגים זאת בעזרת שתי השאלות הבאות.

שאלה 6.5

נתון קטע התוכנית הבא שהקלט שלו הוא מספר לא שלילי, והמשתנים בו הם מטיפוס ממשי.

שימו ♥: בקטע התוכנית נעשה שימוש בפעולות `Sqrt` ו-`Floor` של המחלקה המתמטית `Math` שהוצגה בפרק 4. הפעולה `Floor` מקבלת מספר ממשי ומחזירה את החלק השלם שלו (למשל, ערך הביטוי `Math.Floor(5.8)` שווה ל-5.0)

```
num = double.Parse(Console.ReadLine());
sqrtNum = Math.Sqrt(num);
fraction = sqrtNum - Math.Floor(sqrtNum);
if (fraction > 0)
    Console.WriteLine("1");
else
    Console.WriteLine("0");
```

א. תנו דוגמת קלט שעבורה יהיה הפלט 1.

ב. תנו דוגמת קלט שעבורה יהיה הפלט 0.

ג. תארו את מטרת קטע התוכנית וחלקו את הקלטים האפשריים לשתי קבוצות. תארו את הפלט עבור הקלט בכל קבוצה.

שאלה 6.6

נתון קטע התוכנית הבא, שהקלט שלו הוא ארבעה מספרים שלמים שונים. כל המשתנים בקטע התוכנית הם מטיפוס שלם.

```
num1 = int.Parse(Console.ReadLine());
num2 = int.Parse(Console.ReadLine());
num3 = int.Parse(Console.ReadLine());
num4 = int.Parse(Console.ReadLine());
if (num1 > num2)
    max1 = num1;
else
    max1 = num2;
if (num3 > num4)
    max2 = num3;
```

```
else
    max2 = num4;
if (max1 > max2)
    Console.WriteLine(max1);
else
    Console.WriteLine(max2);
```

א. מהו הפלט עבור כל אחת מדוגמאות הקלט הבאות: 10 20 30 40 ו-50 40 30 20.

ב. תנו שלוש דוגמאות קלט שונות של מספרים חיוביים שהפלט עבורן הוא 5.

ג. מהי מטרת קטע התוכנית?

שאלה 6.7

לפניכם תוכנית בשפת C#:

```
/*  התוכנית קולטת שלושה מספרים שלמים  */
                                              /* התוכנית */
using System;
public class What
{
    public static void Main ()
    {
        // הגדרת משתנים
        int num1, num2, num3;  // משתני הקלט
        int temp;              // משתנה עזר
        //קלט
        Console.Write("Enter first number: ");
        num1 = int.Parse(Console.ReadLine());
        Console.Write("Enter second number: ");
        num2 = int.Parse(Console.ReadLine());
        Console.Write("Enter third number: ");
        num3 = int.Parse(Console.ReadLine());
        if (num1 > num2)
        {
            temp = num1;
            num1 = num2;
            num2 = temp;
        }
        if (num2 > num3)
        {
            temp = num2;
            num2 = num3;
            num3 = temp;
        }
        // פלט
        Console.WriteLine("{0},{1},{2}", num1, num2, num3);
    } // Main
} // class What
```

א. כתבו **במשפט אחד** מהי לדעתכם מטרת התוכנית.

ב. הציעו חלוקה ממצה של הקלטים האפשריים לקבוצות.

ג. בחרו דוגמאות קלט מייצגות על פי החלוקה שהצעתם בסעיף א', וציינו את הפלט עבור כל אחת מהן.

ד. האם התוכנית משיגה את מטרתה? אם כן, הסבירו מדוע; אם לא, תקנו אותה.

סיכום

בפרק זה הרחבנו והעמקנו בנושא נכונות של אלגוריתם.

אלגוריתם נכון הוא אלגוריתם אשר ביצועו מביא להשגת המטרה עבור **כל** קלט חוקי (המתאים להגדרת הבעיה).

לא מתקבל על הדעת להיווכח בנכונות של אלגוריתם על ידי בניית טבלת מעקב עבור כל קלט אפשרי.

לכן אנו בוחרים בצורה ממצה דוגמאות מייצגות של קלט ובודקים את מהלך ביצוע האלגוריתם עבור דוגמאות קלט אלה.

דוגמאות קלט מייצגות הן דוגמאות קלט אשר כל אחת מהן מייצגת קבוצת קלטים בעלת מאפיינים שונים.

בחירה ממצה של דוגמאות קלט מייצגות היא בחירה המבטאת חלוקה ממצה של הקלטים האפשריים לקבוצות מייצגות.

לפעמים נתונים תוכניות או קטעי תוכניות אשר מטרתם אינה ברורה. **זיהוי מטרת תוכנית נתונה** נעשה לפי בחינת פלט התוכנית עבור דוגמאות קלט שונות, ולפי הכללת היחס בין הקלט לפלט.

פרק 7 – ביצוע-חוזר

עד כה הכרנו בעיות אשר לשם פתרונן ביצענו מספר תת-משימות שונות, באופן סדרתי. כלומר כל תת-משימה בסדרה בוצעה פעם אחת (ואם זו משימה שביצועה תלוי בתנאי, ייתכן שלא בוצעה אפילו פעם אחת). אולם יש בעיות אשר לצורך פתרונן יש לבצע תת-משימה אחת, או כמה תת-משימות, יותר מפעם אחת, ואולי אף מספר רב של פעמים. בפרק זה נכיר אלגוריתמים אשר מורים על חזרה שוב ושוב על ביצוע של תת-משימה (או תת-משימות). אלגוריתמים אלה כוללים **הוראה לביצוע-חוזר** של קבוצת הוראות.

בסעיף 7.1 נכיר אלגוריתמים שמבנה הביצוע-החוזר בהם הוא פשוט. באלגוריתמים אלה מספר הפעמים של הביצוע-החוזר נקבע **לפני** תחילת ביצועו.

בסעיף 7.4 נכיר אלגוריתמים שמבנה הביצוע-החוזר בהם מורכב יותר. באלגוריתמים אלה מספר הפעמים של הביצוע-החוזר לא נקבע מראש, אלא תלוי בתנאי אשר נבדק שוב ושוב במהלך הביצוע-החוזר.

7.1 ביצוע-חוזר מספר פעמים ידוע מראש

בעיה 1

מטרת הבעיה ופתרונה: הצגת אלגוריתם הכולל הוראה לביצוע-חוזר מספר פעמים ידוע מראש.

עלינו להמיר רשימת מחירים מייצוג בדולרים לייצוג בשקלים. פתחו וישמו אלגוריתם אשר הקלט שלו הוא שער ההמרה מדולרים לשקלים ואחריו רשימה של עשרה מחירים הנתונים בדולרים. הפלט שלו הוא הערך בשקלים של כל אחד מעשרת המחירים. הפלט עבור כל מחיר צריך להינתן מיד אחרי קליטתו ולפני קליטת המחיר הבא.

פירוק הבעיה לתת-משימות

1. קליטת שער ההמרה מדולרים לשקלים
2. קליטת כל אחד מעשרת המחירים בדולרים, חישוב ערכו בשקלים והצגת הערך המחושב

? ניתן לפרט את התת-משימה השנייה. כיצד?

התת-משימה השנייה מורכבת בעצם מביצוע-חוזר, עשר פעמים, של התת-משימות הבאות:

2.1 קליטת מחיר בדולרים
2.2 חישוב ערכו של המחיר בשקלים
2.3 הצגה של הערך המחושב

דרך אחת להורות על ביצוע החוזר עשר פעמים על התת-משימות שניסחנו, היא כמובן לכתוב אותן עשר פעמים, כך:

2.1. קליטת מחיר בדולרים
2.2. חישוב ערכו של המחיר בשקלים
2.3. הצגה של הערך המחושב
2.4. קליטת מחיר בדולרים
2.5. חישוב ערכו של המחיר בשקלים
2.6. הצגה של הערך המחושב

.
.
.
2.28. קליטת מחיר בדולרים
2.29. חישוב ערכו של המחיר בשקלים
2.30. הצגה של הערך המחושב

זהו ניסוח מסורבל כמובן. עבור רשימה של עשרה מחירים נכתבות שלושים הוראות. עבור רשימה של מאה מחירים ייכתבו 300 הוראות. ובעצם, עבור רשימות מחירים באורכים שונים ייכתבו אלגוריתמים שבהם מספר הוראות שונה. כלומר, לא רק שמדובר באלגוריתמים ארוכים מאוד, אלא שעבור שינוי קטן בהגדרת הבעיה (מספר המחירים), יש צורך לבצע שינוי משמעותי באלגוריתם.

האם ניתן להימנע מן הסרבול המתואר? האם ניתן לכתוב אלגוריתם שבו יהיה אותו מספר הוראות עבור רשימות מחירים באורכים שונים?

אכן ניתן באמצעות הוראה לביצוע-חוזר לקבוצת הוראות. בפתרון הבעיה הנוכחית שבה יש להמיר עשרה מחירים ניתן להשתמש בהוראה הבאה לביצוע-חוזר:

בצע 10 פעמים:
קלוט מחיר בדולרים
חשב את ערכו של המחיר בשקלים
הצג את הערך המחושב

עבור רשימה של 100 מחירים ניתן לכתוב אלגוריתם הכולל הוראה לביצוע-חוזר במבנה זהה, אלא שמספר הפעמים המצוין בו בכותרת ההוראה הוא 100 במקום 10.
שימו ♥ להזחה בהוראה לביצוע-חוזר. בהוראה זו (כמו בהוראה לביצוע-בתנאי) אנו מזיחים פנימה את קבוצת ההוראה לביצוע-חוזר.

בחירת משתנים

נשתמש במשתנים הבאים מטיפוס ממשי:
rate – ישמור את שער ההמרה מדולרים לשקלים.
dollarPrice – ישמור מחיר בדולרים
shekelPrice – ישמור את ערכו בשקלים של המחיר השמור ב-dollarPrice

האלגוריתם

1. קלוט שער המרה ב-rate
2. **בצע 10 פעמים:**
   2.1. קלוט מחיר ב-dollarPrice
   2.2. חשב את ערכו בשקלים של המחיר השמור ב-dollarPrice והשם ב-shekelPrice
   2.3. הצג את ערכו של shekelPrice

יישום האלגוריתם

הוראה לביצוע-חוזר במבנה של *בצע מספר פעמים*... מיושמת ב-#C במשפט **for**. משפט **for** משתמש ב**משתנה בקרה**, אשר שולט בביצוע הלולאה. למשל אם ברצוננו לבצע קבוצת הוראות 10 פעמים, נכתוב משפט **for** בצורה הבאה:

```
for (i = 1; i <= 10; i++)
{
    ההוראות לביצוע
}
```

משתנה הבקרה במשפט זה הוא `i`. ערכו מאותחל ב-1 (כפי שמורה משפט ההשמה `i = 1`, המהווה את הרכיב הראשון בסוגריים). אחרי שסדרת ההוראות לביצוע מתבצעת פעם אחת ערכו של משתנה הבקרה גדל ב-1 (על כך מורה הרכיב השלישי בסוגריים: `i++`), והביצוע-החוזר יימשך כל עוד ערכו של `i` קטן או שווה ל-10 (כפי שמורה התנאי `i <= 10`, הרכיב השני בסוגריים).

אם כך בתחילת הביצוע של משפט ה-**for**, `i` יאותחל ב-1. אחרי שקבוצת ההוראות לביצוע תבוצע פעם אחת, ערכו יגדל ל-2. אחרי שקבוצת ההוראות תבוצע פעם שנייה, ערכו יגדל ל-3. אחרי שקבוצת ההוראות תבוצע פעם עשירית, ערכו כבר יהיה 11, ואז יסתיים הביצוע-החוזר, משום שערכו של התנאי `i <= 10` יהיה **false**.

באופן כללי, בביצוע משפט **for** מושם ערך התחלתי במשתנה הבקרה לפי הרכיב הראשון במשפט. לאחר מכן מתבצעת בדיקת התנאי, המתואר ברכיב השני. אם ערכו של התנאי הוא **true** מתבצעות ההוראות לביצוע. בתום ביצוע קבוצת ההוראות גדל ערכו של משתנה הבקרה לפי הרכיב השלישי. כעת מתבצעת שוב בדיקת התנאי. אם ערכו של התנאי הוא **true** מתבצעת שוב קבוצת ההוראות וכך הלאה, עד אשר ערכו של התנאי הוא **false** ואז מסתיים הביצוע.

באופן דומה, ניתן היה לבחור גם במשפט ה-**for** הבא ליישום ההוראה לביצוע-חוזר באלגוריתם:

```
for (i = 0; i < 10; i++)
{
    הוראות לביצוע
}
```

גם במקרה זה ההוראות לביצוע מתבצעות 10 פעמים: פעם אחת כאשר ערכו של `i` שווה ל-0, פעם שנייה כאשר ערכו שווה ל-1, פעם שלישית כאשר ערכו שווה ל-2, ובפעם העשירית ואחרונה כאשר ערכו של `i` שווה ל-9. כאשר ערכו של `i` גדל שוב, ומגיע ל-10, התנאי להמשך הביצוע כבר לא מתקיים, והביצוע-החוזר מסתיים.

שימו ♥: ההוראה `++i` היא למעשה הוראת השמה מקוצרת. היא שקולה להוראת ההשמה: `i = i + 1`. ניתן להשתמש בהוראה זו בכל מקום בתוכנית, לאו דווקא במשפט **for**.

התוכנית המלאה

```
/*
הקלט: שער ההמרה מדולרים לשקלים ורשימה של 10 מחירים בדולרים
הפלט: הערכים השקליים של 10 המחירים הנתונים בדולרים
*/
using System;
public class Convertor
{
    public static void Main ()
    {
        const int HOW_MANY=10; //קבוע: מספר המחירים הנקראים מהקלט
```

```
        double rate;            //שער ההמרה
        double dollarPrice;     //מחיר בדולרים
        double shekelPrice;     //מחיר בשקלים
        int i;                  //משתנה בקרה
        // קלט
 1.     Console.Write("Enter the rate: ");
 2.     rate = double.Parse(Console.ReadLine());
       //   ההוראה לביצוע-חוזר
 3.     for (i = 1; i <= HOW_MANY; i++)
        {
 3.1.       Console.Write("Enter price in Dollars: ");
 3.2.       dollarPrice = double.Parse(Console.ReadLine());
 3.3.       shekelPrice = dollarPrice * rate;
 3.4.       Console.WriteLine("Price in Shekels is {0}",
                                                      shekelPrice);
        }// for
    }// Main
}// Convertor
```

מעקב

בטבלת מעקב הכוללת משפט **for** נכלול עמודה עבור משתנה הבקרה של המשפט. בתוכנית לפתרון בעיה 1, ערכו של משתנה הבקרה i גדל ב-1 אחרי כל ביצוע של קבוצת ההוראות הכלולה במשפט ה-**for**. ערכו בביצוע-החוזר הראשון הוא 1 וערכו בביצוע-החוזר האחרון הוא 10. בנוסף לכך, טבלת המעקב תכלול עמודה עבור התנאי הבוליאני שבכותרת המשפט.

נעקוב אחר ביצוע התוכנית עבור הקלט הבא: שער ההמרה הוא 3, ועשרת המחירים להמרה הם: 100 61.03 71.05 60 50.3 200.01 120.3 18.2 5 10.1.

כדי להימנע מהצגת טבלה ארוכה מדי, תכלול הטבלה הבאה מעקב רק אחרי עיבוד שני המחירים הראשונים והמחיר האחרון שבקלט.

	המשפט לביצוע	i	i<=10	rate	dollar	shekel	**פלט**
1	Console.Write("Enter the rate:");	?		?	?	?	Enter the rate
2	rate = **double**.Parse(Console. ReadLine());	?		3	?	?	
3	**for** (i = 1; i <= 10; i++)	1	**true**	3	?	?	
3.1	Console.Write("Enter price in Dollars: ");	1		3	?	?	Enter price …
3.2	dollarPrice = **double**.Parse(Console.ReadLine());	1		3	10.1	?	
3.3	shekelPrice = dollarPrice * rate;	1		3	10.1	30.3	
3.4	Console.WriteLine("Price in Shekels is {0}",shekelPrice);	1		3	10.1	30.3	Price in Shekels is 30.3
3	**for** (i = 1; i <= 10; i++)	2	**true**	3	10.1	30.3	
3.1	Console.Write("Enter price in Dollars: ");	2		3	10.1	30.3	Enter price …
3.2	dollarPrice = **double**.Parse(Console.ReadLine());	2		3	5	30.3	
3.3	shekelPrice = dollarPrice *	2		3	5	15	

	rate;						
3.4	Console.WriteLine("Price in Shekels is {0}",shekelPrice);	2		3	5	15	Price in Shekels is 15
. . .							
3	**for** (i = 1; i <= 10; i++)	10	**true**	3	61.03	183.09	
3.1	Console.Write("Enter price in Dollars: ");	10		3	61.03	183.09	Enter price …
3.2	dollarPrice = **double**.Parse(Console.ReadLine());	10		3	100	183.09	
3.3	shekelPrice = dollarPrice * rate;	10		3	100	300	
3.4	Console.WriteLine("Price in Shekels is {0}",shekelPrice);	10		3	100	300	Price in Shekels is 300
3	**for** (i = 1; i <= 10; i++)	11	**false**	3	100	300	

שימו לב לשינויים שחלים בערכו של משתנה הבקרה i ולבדיקת התנאי של משתנה הבקרה.

סוף פתרון בעיה 1

נציג את המושגים החדשים שהכרנו בפתרון לבעיה 1.

באלגוריתם לפתרון הבעיה כללנו **הוראה לביצוע-חוזר במבנה *כך וכך מספר פעמים*...** כדי לציין ביצוע-חוזר של תת-משימה. הוראה זו מורה על ביצוע-חוזר של קבוצת הוראות מספר פעמים. הוראה לביצוע-חוזר היא הוראת בקרה.

הוראה לביצוע-חוזר נקראת גם **לולאה** (loop), וקבוצת ההוראות לביצוע הכלולות בה נקראת **גוף הלולאה**.

בדומה לכתיבה של הוראה לביצוע-בתנאי גם כתיבה של הוראה לביצוע-חוזר נעשית תוך הקפדה על **הזחה** מתאימה: קבוצת ההוראות שיש לחזור על ביצוען מוזחת פנימה.

בשפת C# מיושמת הוראה לביצוע-חוזר במבנה זה במשפט **for**.

זהו **המבנה הכללי של משפט for** בשפת C#:

```
for (אתחול משתנה הבקרה; התנאי להמשך הביצוע; שינוי משתנה הבקרה)
{
    הוראות לביצוע
}
```

אתחול משתנה הבקרה: הוראת השמה הקובעת ערך התחלתי למשתנה הבקרה.

התנאי להמשך הביצוע: ביטוי בוליאני שמהווה את התנאי השולט בביצוע-החוזר. התנאי נבדק אחרי אתחול משתנה הבקרה. כמו כן הוא נבדק שוב בכל פעם שמסתיים ביצוע של קבוצת ההוראות-לביצוע. כל עוד התנאי מתקיים הביצוע-החוזר ממשיך. כאשר ערכו של התנאי הוא **false** הביצוע-החוזר מסתיים.

שינוי משתנה הבקרה: השינוי שחל במשתנה הבקרה בכל פעם שמסתיים שלב ביצוע נוסף.

גוף הלולאה תחום בסוגריים מסולסלים. במקרה שגוף הלולאה מכיל משפט בודד אפשר להשמיט את הסוגריים.

למשל כך:

```
for (i = 0; i < 10; i++)
    Console.Write("*");
```

בטבלת מעקב אחר תוכנית הכוללת משפט **for** אנו כוללים עמודה עבור משתנה הבקרה ועמודה עבור התנאי להמשך הביצוע.

משתנה הבקרה במשפט **for** הוא בדרך כלל מטיפוס שלם. מאחר שבמקרים רבים תפקידו של משתנה הבקרה הוא רק לשלוט במשפט ה-**for**, ואין בו שימוש בחלקי התוכנית האחרים, שפת C# מאפשרת להצהיר על משתנה הבקרה בתוך הוראת ה- **for**, כלומר, נוכל לכתוב:

```
for (int i = 1; i <= 10; i++ )
```

כאשר מצהירים על משתנה הבקרה בתוך משפט ה-**for**, אין אליו גישה מחוץ לתחום משפט ה-**for**.

שאלה 7.1

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `Convertor` (לפתרון בעיה 1) עבור הקלט שבו שער ההמרה הוא 4.5 ועשרת המחירים להמרה הם:

5.05 18.01 17.03 20.9 101 105 213.05 16.1 17.2 18.3

פרטו בטבלה רק את השורות המתאימות לעיבוד שני הקלטים הראשונים ולעיבוד הקלט האחרון (בדומה לנעשה בפתרון בעיה 1).

שאלה 7.2

נסחו עבור כל אחת מן הבעיות האלגוריתמיות הבאות קבוצת תת-משימות לביצוע-חוזר:

א. הקלט הוא 20 מרחקים הנתונים במיילים, והפלט הוא 20 המרחקים בקילומטרים (1 מייל = 1.6 קילומטר).

ב. הקלט הוא עשר אותיות מן הא״ב האנגלי השונות מהאות Z, והפלט הוא עשר האותיות שעוקבות לאותיות הנתונות.

ג. הקלט הוא 40 זוגות של ציונים (זוג ציונים עבור כל תלמיד), והפלט הוא רשימה של ארבעים מספרים: כל מספר הוא הממוצע של זוג הציונים המתאים לו.

שאלה 7.3

לפניכם קטע תוכנית:

```
Console.Write('X');
for (i = 0; i < 10; i++)
    Console.Write("*");
Console.Write('X');
```

מהו פלט קטע התוכנית?

שאלה 7.4

פתחו וישמו אלגוריתם אשר הקלט שלו הוא תו, והפלט שלו הוא שכפול של התו הנקלט חמישים פעמים. למשל, עבור הקלט A יהיה הפלט AAA...AA (חמישים פעמים).

בשלב החלוקה לתת-משימות הקפידו על ניסוח תת-משימה לביצוע-חוזר.

שאלה 7.5

פתחו וישמו אלגוריתם אשר הקלט שלו הוא 20 מספרים שלמים חיוביים דו-ספרתיים, והפלט שלו הוא סכום הספרות לכל אחד מהמספרים הנתונים.

למשל, אם הקלט הוא:

11 17 99 10 20 30 10 20 30 10 20 30 10 10 20 20 30 30 88 15

הפלט המתאים הוא:

2 8 18 1 2 3 1 2 3 1 2 3 1 1 2 2 3 3 16 6

בשלב החלוקה לתת-משימות הקפידו על ניסוח תת-משימה לביצוע-חוזר, ובשלב הבדיקה הקפידו על בדיקה מסודרת באמצעות טבלת מעקב.

בפתרון בעיה 1 מספר הפעמים לביצוע-חוזר נקבע עוד לפני תחילת ביצוע התוכנית. במקרים רבים, ייתכן כי מספר הפעמים לביצוע-חוזר ידוע לפני שמתחיל ביצועה של ההוראה לביצוע-חוזר, אך לא לפני תחילת ביצוע התוכנית. כלומר הוא תלוי בקלט. מקרה כזה מודגם בבעיה הבאה:

בעיה 2

מטרת הבעיה ופתרונה: הצגת ביצוע-חוזר שאורכו נקבע על פי נתון קלט.

פתחו אלגוריתם שיקבל כקלט מספר שלם N, ויציג על המסך שורה של כוכביות באורך N.

פירוק הבעיה לתת-משימות

1. קליטת אורך לרשימת כוכביות
2. הדפסת כוכביות לפי המספר הנקלט

רשימת המשתנים

עד עתה השתמשנו במספרים קבועים בתנאי לסיום הביצוע-החוזר. בבעיה זו אנו נדרשים לקבל כקלט את מספר הפעמים שתתבצע הלולאה, ולכן התנאי להמשך הביצוע יהיה תלוי בערכו של משתנה.

numOfTimes – מספר הפעמים שיש להציג כוכבית

i – משתנה הבקרה של הלולאה

האלגוריתם

1. קלוט מספר שלם ב-numOfTimes
2. בצע numOfTimes פעמים:
   2.1. הצג '*'

יישום האלגוריתם

בפתרון בעיה 1, כאשר רצינו לבצע קבוצת משימות 10 פעמים יישמנו זאת באמצעות משפט **for** שכותרתו:

```
for(i = 1; i <= 10; i++)
```

(או באמצעות משפט שכותרתו `for(i = 1; i <= HOW_MANY; i++)`, ו-`HOW_MANY` הוגדר כקבוע שערכו 10).

כעת אנחנו רוצים לבצע את הוראה 2.1 numOfTimes פעמים. לכן ניישם את ההוראה לביצוע-חוזר במשפט **for** שכותרתו:

```
for(i = 1; i <= numOfTimes; i++)
```

התוכנית המלאה

```
/*
הקלט: מספר שלם
הפלט: שורה של כוכביות באורך הנתון כקלט
*/
using System;
public class Stars
{
    public static void Main ()
    {
           int numOfTimes;
           Console.Write("Enter a number please: ");
           numOfTimes = int.Parse(Console.ReadLine());
           for (int i = 1; i <= numOfTimes; i++)
                Console.Write('*');
    } // Main
} // Stars
```

סוף פתרון בעיה 2

בתוכנית `Stars` ראינו שימוש בנתון קלט לקביעת מספר הפעמים לביצוע לולאה. בפיתוח אלגוריתמים בהמשך נשתמש בכך פעמים רבות.

באמצעות הבעיה הבאה נכיר שני סוגים של משתנים המשמשים בפתרון בעיות רבות ותבנית העבודה איתם היא שימושית מאוד. כפי שנראה, תבנית העבודה עם משתנים כאלה משתמשת בהוראות לביצוע-חוזר.

בעיה 3

מטרת הבעיה ופתרונה: הצגת **מונה** ו**צובר** ואופן העבודה איתם.

פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר חיובי שלם, ולאחריו רשימה של מספרים ממשיים, שאורכה שווה לערך הקלט הראשון. הפלט שלו הוא ממוצע המספרים ברשימה ומספר המספרים ברשימה שערכם עולה על 50.

למשל, עבור הקלט:

10 23.4 100 95 78 64.15 75 90.3 54.2 67 20

הפלט המתאים הוא: 8 66.7

משום שממוצע המספרים הוא 66.7, ו-8 מתוכם הם גדולים מ-50.

פירוק הבעיה לתת-משימות

1. קליטת אורך הרשימה
2. קליטת רשימת הערכים, סיכומם, ומניית מספר הערכים הגבוהים מ-50
3. חישוב ממוצע הערכים
4. הצגה כפלט של הממוצע שחושב ושל מספר הערכים הגבוהים מ-50

? כדי לחשב את הממוצע יש לסכם את הערכים הנתונים בקלט, ובנוסף יש למנות כמה מהערכים הנתונים בקלט הם גבוהים מ-50. ניתן לבצע הן את פעולת הסיכום והן את פעולת המנייה תוך כדי קריאת נתוני הקלט. כיצד ניתן לבטא זאת כהוראה לביצוע-חוזר?

ניתן לבצע את פעולת **הסיכום** בהוספת כל ערך שנקלט לסכום מצטבר, עוד לפני קריאת הערך הבא. בדומה, ניתן לבצע את פעולת **המנייה** בהשוואת כל ערך שנקלט ל-50, ובהגדלת מונה מתאים בכל פעם שהערך שנקלט עולה על 50. לכן, את תת-משימה 2 נוכל לבצע באמצעות ביצוע-חוזר של קבוצת ההוראות הבאה:

2.1. קליטת ערך
2.2. הוספת הערך שנקלט לסכום המצטבר
2.3. השוואת הערך שנקלט ל-50. אם גבוה מ-50, הגדלת ערכו של מונה למניית מספר הערכים הגבוהים מ-50.

בחירת משתנים

מן התת-משימות לביצוע-חוזר ניתן להסיק שיש להשתמש במשתנה אשר יישמר בו ערך תורן שנקרא מהקלט, במשתנה שיצבור את **סכום** הערכים, ובמשתנה **שימנה** את מספר הערכים הגבוהים מ-50. לכן ניעזר במשתנים הבאים:

length – מטיפוס שלם, ישמור את אורך רשימת המספרים
num – מטיפוס ממשי, ישמור ערך תורן הנקרא מהקלט
sum – מטיפוס ממשי, **צובר** שישמור את סכום הערכים
average – מטיפוס ממשי, ישמור את ממוצע הערכים
counterLarge – מטיפוס שלם, **מונה** שישמור את מספר הערכים הגבוהים מ-50

שימו ♥: המשתנים average, num, sum הם מטיפוס ממשי, כיוון שערכי הקלט בבעיה הם ממשיים. לעומתם counterLarge הוא מטיפוס שלם כיוון שהוא משמש למנייה.

יישום האלגוריתם

את ההוראה לביצוע-חוזר נוכל לנסח באופן הבא:

בצע length *פעמים:*
1. *קלוט ערך ממשי ב-*num
2. *הוסף את ערכו של* num *לסכום המצטבר השמור ב-*sum
3. *אם ערכו של* num *גדול מ-50*
3.1. *הגדל ב-1 את ערכו של* counterLarge

? אילו הוראות יש להוסיף **לפני** ההוראה לביצוע-חוזר?

שימו ♥: כדי שהצבירה והמנייה יתבצעו באופן נכון, עלינו לאתחל נכונה את הערכים של sum ושל counterLarge. על פי תפקידם של שני המשתנים באלגוריתם יש לאתחל את ערכיהם ב-0.

התוכנית המלאה

```
/*
קלט: רשימת ערכים ממשיים
פלט: ממוצע הערכים ומספר הערכים הגבוהים מ-50
*/
using System;
public class CalcAvgAndCountLargerThan50
{
    public static void Main ()
    {
        // הצהרה על קבוע בתוכנית
        const int LIMIT = 50;
```

```
        // הצהרה על משתנים בתוכנית
        int length;      //אורך רשימת הקלט
        double num;      //ערך קלט תורן
        double sum = 0;  // צובר
        double average;  // ממוצע
        int  counterLarge = 0;       //מונה למספר הערכים הגבוהים מ-50
        Console.Write("Enter length of input list: ");
        length = int.Parse(Console.ReadLine());
        //   ההוראה לביצוע-חוזר //
        for (int i = 1; i <= length; i++)
        {
            Console.Write("Enter number: ");
            num = double.Parse(Console.ReadLine());
            sum = sum + num;
            if (num > LIMIT)
                counterLarge++;  // counterLarge=counterLarge+1 -שקול ל
        }// for
        average = sum / length;
        Console.WriteLine("Average is {0}", average);
        Console.WriteLine("{0} numbers are larger than {1}",
                                          counterLarge, LIMIT);
    }// Main
}// CalcAvgAndCountLargerThan50
```

סוף פתרון בעיה 3

באלגוריתם לפתרון בעיה 3 משמש המשתנה sum **כצובר** של הערכים הנתונים.

צובר הוא משתנה אשר תפקידו לצבור ערכים. למשל ניתן להשתמש בצובר לסכימת ערכים.

המשתנה counterLarge משמש **כמונה** של מספר הערכים הגבוהים מ-50 מבין הערכים הנתונים.

מונה הינו משתנה אשר תפקידו למנות את מספר הפעמים שהתרחש אירוע מסוים (למשל מספר הפעמים שנקרא נתון קלט). כיוון שהשימוש במונה הוא לספירה, מונה הוא משתנה מ**טיפוס שלם**.

בתוכנית לפתרון בעיה 3 מופיעים המשפטים המבטאים את פעולות הצבירה והמנייה בגוף הלולאה לאחר משפט הקלט.

המשפט המבטא את **פעולת הצבירה** בגוף הלולאה הוא:

```
sum = sum + num;
```

המשפט המבטא את **פעולת המנייה** בגוף הלולאה הוא:

```
counterLarge++;
```

מאחר שתחזוקה של מונה או של צובר כוללת ביצוע של פעולות עדכון חוזרות ונשנות, נבצע בדרך-כלל פעולות צבירה ומנייה בגוף הלולאה.

שימו ♥ בשני המקרים גדל המשתנה המשמש כצובר או כמונה בערך כלשהו. במקרה של צובר הוא גדל בערך ששייך לקבוצת הערכים המצטברים. במקרה של מונה הוא גדל ב-1.

באלגוריתם שמשתמשים בו בצובר או במונה יש לאתחל את הצובר או את המונה. בתוכנית לפתרון הבעיה מאותחלים **הצובר** `sum` **והמונה** `counterLarge` **ב-0**.

אִתְחוּל של צובר או של מונה הוא השמת ערך המתאים לתחילת תהליך הצבירה או המנייה. מונה מאותחל בדרך כלל ב-0. הערך ההתחלתי המתאים לצובר תלוי במהות הצבירה המתבצעת בו. למשל, צובר השומר סכום מצטבר יאותחל ב-0.

שאלה 7.6

שנו את המשפטים הנמצאים בגוף הלולאה שבתוכנית לפתרון בעיה 3, כך שיחושב ממוצע הערכים הגבוהים מ-50.

שאלה 7.7

לפעמים ניתן להשתמש בערכו של מונה לחישוב מספר הנתונים המאופיינים בצורה הפוכה לנתונים שמנינו. למשל, נניח שבבעיה 3 יש להציג גם את מספר הערכים **הקטנים או שווים** ל-50.

דרך אחת לחישוב מספר זה היא באמצעות שימוש במונה נוסף `counterSmall` (נוסף ל-`counterLarge`), אשר ערכו יוגדל ב-1 בכל פעם שנקלוט ערך הקטן או שווה ל-50.

אך בעצם אין צורך במונה נוסף. ניתן לבצע את החישוב בתום ביצוע הלולאה, באמצעות המונה `counterLarge` המופיע כבר בתוכנית. כיצד? הוסיפו לתוכנית את ההוראה או את ההוראות המתאימות.

שאלה 7.8

ציינו עבור כל אחת מן הבעיות האלגוריתמיות הבאות אם נחוץ לפתרונה צובר, מונה או אף אחד מהשניים:

א. קלט: רשימת מחירים, פלט: סך כל המחירים.

ב. קלט: רשימת מחירים, פלט: מספר המחירים הגבוהים מ-100.

ג. קלט: רשימת מחירים, פלט: מספר המחירים שמרכיב האגורות בהם שונה מ-0.

ד. קלט: רשימת מספרים, פלט: הערך המוחלט של כל אחד מהמספרים.

ה. קלט: רשימת מספרים, פלט: הממוצע של הערכים המוחלטים של המספרים.

שאלה 7.9

יש לקלוט סדרה של תווים, אשר אורכה שמור במשתנה `listSize`, ולמנות את מספר התווים שהם אותיות גדולות (capital letters) בא״ב האנגלי.

בחרו משתנים, כתבו הוראה לביצוע-חוזר אשר לפניה אתחול מתאים, וישמו אותה במשפט **`for`**.

שאלה 7.10

מטרת קטע התוכנית הבא היא מניית מספר תווי קלט השונים מן האות A. המשתנים `counter` ו-`listSize` הם מטיפוס שלם והמשתנה `letter` הוא מטיפוס תווי.

```
int counter = __________ ;
for (int i = 1; i <= listSize; i++)
{
    Console.Write("Enter a char: ");
    letter  = char.Parse(Console.ReadLine());
    if (__________________)

         __________________
}// for
Console.WriteLine("There are {0} letters different than A", counter);
```

קטע התוכנית כולל לולאה.

א. כמה פעמים תתבצע הלולאה עבור הערך 10 ב-`listSize`?

ב. כמה פעמים תתבצע הלולאה עבור הערך 1 ב-`listSize`?
ג. השלימו את קטע התוכנית.

שאלה 7.11

נתונה התוכנית הבאה:

```
/*
קלט: טור של 16 תווים מטופס ספורטוטו
פלט: ________________________________________
*/
using System;
public class Toto
{
    public static void Main ()
    {
        const int NUM_OF_GAMES = 16;
        int d = 0;
        char score;
        for (int i = 0; i < NUM_OF_GAMES; i++)
        {
            Console.Write("Enter the score: ");
            score = char.Parse(Console.ReadLine());
            if (score == 'X')
                d++;
        } // for
        Console.WriteLine(d);
    } // Main
} // Toto
```

קלט התוכנית הוא טור בטופס הספורטוטו. כלומר, 16 תווים שכל אחד מהם מציין תוצאת משחק (1, 2 או X).

א. מהו הפלט עבור הקלט: 1 2 X 1 2 X 1 2 X 1 2 X 1 2 X 2?
ב. האם התוכנית כוללת מונה או צובר? אם כן, מהו או מהם?
ג. כמה פעמים תתבצע לולאת התוכנית?
ד. הביאו דוגמת קלט אשר הפלט עבורה הוא 0.
ה. הביאו דוגמת קלט אשר הפלט עבורה הוא 15.
ו. מה מאפיין את הקלטים אשר הפלט עבורם הוא 1?
ז. מהם ערכי הפלט האפשריים?
ח. מהי מטרת התוכנית? בחרו שם משמעותי למשתנה d.

שאלה 7.12

פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר המציין אורך של רשימה ואחריו רשימת מספרים ממשיים באורך הנתון. הפלט שלו הוא סכום החלקים השלמים של המספרים הממשיים הנתונים.

שאלה 7.13

כתבו קטע תוכנית אשר מקבל כקלט מספר שלם num. התוכנית תגריל num מספרים בתחום 10-50 ותבדוק כמה מספרים מבין המספרים שהוגרלו הם זוגיים.

שאלה 7.14

כתבו קטע תוכנית שיגריל 10 מספרים תלת-ספרתיים. התוכנית תחשב ותדפיס את סכום המספרים האי-זוגיים בלבד מבין המספרים שהוגרלו.

בדוגמאות שראינו עד כה השתמשנו בצובר לצבירת סכום. ניתן להשתמש בצובר גם לצבירת מכפלה. כאשר צובר משמש לצבירת מכפלה הוא מאותחל בערך שונה מ-0. שימו לב לכך בשאלה הבאה:

שאלה 7.15

פתחו וישמו אלגוריתם שיקבל כקלט רשימה של 50 מספרים ויציג כפלט את **מכפלתם** של המספרים הקטנים מ-10.

להעמקה בתבניות ***מנייה*** ו***צבירה*** פנו לסעיף התבניות המופיע בסוף הפרק.

בעיה 3 השתמשה גם בתבנית ***ממוצע***. להעמקה בתבנית ***ממוצע*** עבור סדרה שיכולה להכיל יותר מ-3 ערכים פנו לסעיף התבניות המופיע בסוף הפרק.

בעיה 4

מטרת הבעיה ופתרונה: עוד על הוראה לביצוע-חוזר מספר פעמים ידוע מראש: הצגת הוראה לביצוע-חוזר המטפלת בתחום של מספרים, והדגמת שימוש במשתנה הבקרה בתוך גוף הלולאה.

פתחו וישמו אלגוריתם שיקבל כקלט מספר שלם, ויציג כפלט את כל המספרים השלמים החיוביים הקטנים מהמספר הנתון. למשל עבור הקלט 5 הפלט המתאים הוא: 4 3 2 1.

פירוק הבעיה לתת-משימות

ברצוננו לקלוט מספר שלם num ולהציג num-1 ערכים, מ-1 עד num-1. אם כך, החלוקה לתת-משימות היא ברורה למדי:

1. קליטת מספר מהקלט
2. הצגה כפלט של כל הערכים החיוביים והשלמים שקטנים מהמספר שנקלט

ברור כי לשם ביצוע תת-משימה 2 נזדקק להוראה לביצוע-חוזר, שתתבצע num-1 פעמים.

בחירת משתנים

ברור כי נזדקק למשתנה עבור הערך הנקרא מהקלט. סביר כי נהיה גם זקוקים למשתנה כלשהו, עבור האיבר התורן להצגה. משתנה זה יאותחל ב-1 ויקודם בסיום כל סיבוב בלולאה ב-1, עד אשר ערכו יגיע ל-num.

אבל איננו זקוקים למשתנה **נוסף** כי תיאור זה מתאים בדיוק למשתנה הבקרה של הלולאה!

לכן נקבל את רשימת המשתנים הבאה:

num – מטיפוס שלם, לשמירת הערך הנקרא מהקלט

i – משתנה הבקרה

האלגוריתם

כאמור תת-משימה 2 תתבצע באמצעות הוראה לביצוע-חוזר. למעשה בהוראה זו אנו מעוניינים לעבור על תחום של ערכים (מהערך 1 ועד הערך num-1) ולהציג כל ערך בתחום. כדי לבטא זאת, ננסח את ההוראה לביצוע-חוזר בצורה שונה מעט מזו שראינו באלגוריתמים קודמים בפרק.

1. *קלוט מספר שלם ב-*num
2. *עבור כל* i *שלם מ-1 עד* num-1 ***בצע:***
   2.1. *הצג את ערכו של* i

יישום האלגוריתם

את הוראה 2 באלגוריתם שכתבנו נישם במשפט **for**. זהו משפט **for** הדומה מאוד לאלה שראינו בתוכניות קודמות בפרק, רק שמשתנה הבקרה בו אינו משמש רק כדי לשלוט בביצוע הלולאה. יש התייחסות למשתנה הבקרה גם בתוך גוף הלולאה, ולא רק בשלושת הרכיבים שבכותרת הלולאה.

התוכנית המלאה

```
/*
קלט: מספר שלם
פלט: כל המספרים השלמים והחיוביים הקטנים מהמספר הנתון
*/
using System;
public class PrintNumbers
{
    public static void Main()
    {
        int num;        // ערך הנקרא מהקלט
        Console.Write("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        for (int i = 1; i < num; i++)
            Console.WriteLine(i);
    } // Main
} // PrintNumbers
```

סוף פתרון בעיה 4

שאלה 7.16

על פי הגדרת בעיה 4, יכול להינתן כנתון ראשון בקלט כל מספר שלם. תארו את מהלך ביצוע התוכנית `PrintNumbers` אם הערך הראשון בקלט הוא המספר השלם 0. תארו את מהלך הביצוע של התוכנית אם הערך הראשון בקלט הוא המספר השלם 2-.

שאלה 7.17

פתחו אלגוריתם שמקבל כקלט מספר חיובי שלם n, מחשב את !n, ומציג את הערך שחושב כפלט. ישמו את האלגוריתם בשפת #C.

להזכירכם: !n היא מכפלת המספרים מ-1 עד n, כלומר: $n! = 1\cdot 2\cdot \ldots \cdot n$.

שאלה 7.18

פתחו אלגוריתם שמקבל כקלט מספר חיובי שלם n, ומחשב את הסכום: $\frac{1}{1}+\frac{1}{2}+\ldots+\frac{1}{n}$.

ישמו את האלגוריתם בשפת #C.

כזכור הרכיב השלישי במשפט **for** הוא משפט השמה המתאר את השינוי של משתנה הבקרה בתום כל סיבוב בלולאה. עד עתה כתבנו תוכניות שבהן משתנה הבקרה גדל ב-1 בתום כל סיבוב של הלולאה. אבל לעתים נרצה לקדם את משתנה הבקרה ב-2, 3 או אפילו פי 2. שפת #C מאפשרת לנו לתאר שינויים כאלה, פשוט בכתיבת משפט השמה מתאים. הנה מספר דוגמאות לכותרות משפטי **for** כאלה:

```
for(int i = small; i < big; i = i + 2)
for(int i = small; i < big; i = i * 2)
for(int i = big; i > small; i--)
```

(כפי שוודאי הבנתם, `--i` היא דרך מקוצרת לכתוב: `i = i - 1`).
במקרים רבים, אכן מתאים לבצע עדכונים שונים של משתנה הבקרה, כפי שמדגימות השאלות הבאות.

שאלה 7.19

לפניכם קטע תוכנית הכתוב ב-#C:

```
for (int i = 1; i <= 50; i = i * 2)
    Console.WriteLine(i);
```

א. עקבו בעזרת טבלת מעקב אחר קטע התוכנית? מה יודפס?
ב. נניח כי במקום הערך 50 מופיע ערך חיובי שלם כלשהו N. תארו מה מבצע קטע התוכנית כתלות בערך N.

שאלה 7.20

פתחו אלגוריתם אשר מקבל כקלט מספר שלם ומציג כפלט את כל המספרים **הזוגיים** החיוביים הקטנים מהמספר שנקרא.

7.2 מציאת מקסימום או מינימום

שתיים מהבעיות הבסיסיות ביותר במדעי המחשב הן חישוב הערך הגדול ביותר או הקטן ביותר ברשימה של ערכים נתונים. פתרונן מהווה תבנית שימושית מאוד. בסעיף זה נערוך היכרות עם בעיות אלו ונראה כי כאשר אורך הרשימה ידוע ניתן לפתור אותן בעזרת הוראה לביצוע-חוזר מספר פעמים ידוע מראש.

בעיה 5

מטרת הבעיה ופתרונה: הצגת אופן החישוב של הערך הגדול ביותר ברשימת ערכים נתונים שאורכה נתון אף הוא.

> מנהל סניף הבנק החליט לבדוק מהו סכום הכסף הגדול ביותר השמור בחשבונות של לקוחות הסניף. פתחו אלגוריתם אשר קולט את מספר החשבונות בסניף, ולאחר מכן את סכום הכסף הנמצא בכל חשבון וחשבון (כמספר שלם). פלט האלגוריתם יהיה הסכום הגבוה ביותר. ישמו את האלגוריתם בשפת #C.

ניתוח הבעיה בעזרת דוגמאות

הרעיון המרכזי בפתרון הבעיה הוא לזכור **בכל זמן** מהלך הביצוע, מהו הערך הגדול ביותר מבין אלה שכבר נקלטו. ערך זה יושווה תמיד לערך הבא שנקרא מהקלט, ויתעדכן בהתאם לתוצאת ההשוואה.

פירוק הבעיה לתת-משימות

את הרעיון שתיארנו נוכל לבטא באמצעות התת-משימה הבאה ועל ביצועה יש לחזור לאחר כל קליטת ערך נוסף:

השוואת הסכום האחרון שנקלט לסכום הגדול ביותר שנקלט עד כה, ועדכון הסכום הגדול ביותר שנקרא עד כה על פי תוצאת ההשוואה

בחירת משתנים

נזדקק למשתנה שיזכור את מספר ערכי הקלט (מספר החשבונות), ולמשתנה לשמירת הערך התורן בקלט. בנוסף נזדקק למשתנה כדי לזכור את הערך הגדול ביותר מבין אלה שנקלטו. נבחר את המשתנים הבאים מטיפוס שלם:

howMany – מספר הערכים ברשימת הנתונים, מספר החשבונות בסניף

balance – הערך התורן ברשימת ערכי הקלט

max – לשמירת הסכום הגדול ביותר מבין אלה שנקלטו

האלגוריתם

max יאותחל בסכום הראשון ברשימה, כיוון שמיד אחרי שנקרא הסכום הראשון, הוא בוודאי הגדול ביותר מבין כל הסכומים שנקראו. כעת יש לקרוא את כל הסכומים האחרים בעזרת לולאת `for`. לאחר כל קליטה של סכום נוסף, יש להשוות את הסכום החדש שנקלט לסכום הגדול ביותר שנקרא עד כה (השמור ב-max). אם הסכום החדש גדול מזה השמור ב-max, יישמר ב-max הסכום החדש.

? כמה פעמים צריך להתבצע גוף הלולאה?

הסכום הראשון נקרא מהקלט עוד לפני הלולאה לצורך האתחול של המשתנה max. לכן הלולאה צריכה להתבצע רק howMany-1 פעמים כדי לקלוט ולעבד את ערכי הקלט הנותרים.

יישום האלגוריתם

```
/*
קלט: מספר חשבונות בנק, ורשימת הסכומים בחשבונות אלה
פלט: הסכום הגבוה ביותר ברשימה
*/
using System;
public class FindMax
{
    public static void Main ()
    {
        int howMany;      // מספר הערכים ברשימה
        int balance;      // הסכום התורן
        int max;          // הסכום הגבוה ביותר מבין אלה שנקלטו
1.      Console.Write("Enter the amount of accounts: ");
2.      howMany = int.Parse(Console.ReadLine());
```

```
3.      Console.Write("Enter the first balance: ");
4.      balance = int.Parse(Console.ReadLine());
5.      max = balance;        //אתחול המקסימום לסכום הראשון
6.      for(int i = 2; i <= howMany; i++)
        {
6.1.        Console.Write("Insert balance of account {0}: ", i);
6.2.        balance = int.Parse(Console.ReadLine());
6.3.        if (balance > max) //הסכום הנוכחי גדול מהמקסימום הנוכחי
6.3.1.          max = balance;
        } // for
7.      Console.WriteLine("The maximum is {0}", max);
    }// Main
}// FindMax
```

שימו ♥ בתוך הלולאה אפשר להשתמש במשתנה הבקרה i כמשתנה לכל דבר.

סוף פתרון בעיה 5

שאלה 7.21

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `FindMax` לפתרון בעיה 5 עבור הקלט:

5 3 2 4 6 -9

כמה פעמים במהלך ביצוע התוכנית מושם ערך במשתנה `max`?

שאלה 7.22

בפתרון בעיה 5 אותחל `max` לערכו של הנתון הראשון ברשימה. האם הפתרון היה נכון לו `max` היה מאותחל בערך קבוע כלשהו, למשל 0? הערה: זכרו כי ייתכן שזהו בנק לא מוצלח במיוחד וכל חשבונות הבנק בו במשיכת יתר, כלומר בעלי ערך שלילי.

שאלה 7.23

נתון קטע התוכנית החלקי הבא לחישוב המספר **הקטן** ביותר ברשימת מספרים **חיוביים** נתונה, אשר אורכה שמור במשתנה `len`.

```
min = _____________
Console.Write("Enter the list size: ");
len = int.Parse(Console.ReadLine());
for (int i = 1; i <= len; i++)
{

      _____________________

      _____________________

      _____________________
}
Console.WriteLine("The minimum is: {0}", min);
```

השלימו את קטע התוכנית (הוסיפו משתנה או משתנים במידת הצורך).

במשפט הראשון אתחלו את `min` לערך קבוע כלשהו (ולא לערך הראשון ברשימה).

שאלה 7.24

פתחו וישמו אלגוריתם שמקבל כקלט רשימה של ציונים במדעי המחשב של 40 תלמידים. הפלט הוא הציון הגבוה ביותר מבין התלמידים **שנכשלו** במבחן (ציון נכשל הוא ציון הנמוך מ-55).

שימו לב: כאן הציון התורן הנקרא מהקלט אינו מושווה בכל מקרה למקסימום הנוכחי, אלא רק כאשר הוא קטן מ-55.

כפי שראינו בתרגילים הקודמים, יש דרכים שונות לאתחול בתבנית מציאת מקסימום (וכמובן, כך גם לגבי תבנית מציאת מינימום). המשתנה ששומר את המקסימום התורן, צריך בכל שלב לשמור את האיבר הגדול ביותר מבין האיברים שהושוו עד כה. בדרך כלל נעדיף לאתחל את המשתנה הזה בערך האיבר הראשון ברשימה. אכן אחרי שנקלט איבר אחד בלבד, ודאי שהוא הגדול מבין כל אלה שנקלטו. אך ישנם מקרים שעובדה זו אינה מספיקה ולא נוכל להשתמש באתחול כזה. כך למשל בשאלה 7.24: בשאלה זו אנו מעוניינים למצוא מקסימום רק מבין הציונים הנכשלים ולא מבין כל הציונים שבקלט. כלומר לא כל איבר שנקלט צריך להיות מושווה לצורך מציאת מקסימום. בפרט לא בטוח שהאיבר הראשון הוא ציון נכשל ולכן בכלל לא יהיה מועמד למקסימום. לכן יהיה שגוי לקבוע אותו כמקסימום התחלתי.

במקרים כאלה נוכל להשתמש בקצוות של טווח הערכים האפשריים. למשל אם מדובר בציונים, ערכם יכול לנוע מ-0 עד 100. במקרה כזה, יהיה נכון לאתחל את המקסימום התורן ב-0 (ואת המינימום התורן ב-100). כיוון שברור שהערך הראשון שנשווה למקסימום התורן ערכו הוא 0 לפחות, הרי מיד אחרי ההשוואה הראשונה המקסימום התורן יכיל את האיבר הראשון שהושווה.

שימו ♥ : לא תמיד ידועים ערכי קצה לטווח הערכים האפשריים! (ראו את שאלה 7.22)

להעמקה בתבניות ***מציאת מקסימום*** ו***מציאת מינימום*** פנו לסעיף התבניות המופיע בסוף הפרק

7.3 מציאת ערך נלווה למקסימום או למינימום

בסעיף זה נכיר בעיות הדומות לבעיית מציאת מקסימום או מינימום, אך מעט יותר מורכבות. גם פתרונותיהן מהווים תבניות שימושיות.

בעיה 6

מטרת הבעיה ופתרונה: הצגת אופן חישוב **מיקומו** של הערך הגדול ביותר והקטן ביותר ברשימת ערכים נתונים.

בנגן MP3 שמורים 100 שירים. לכל שיר מספר ייחודי משלו בין 1 ל-100. על צג המכשיר מוצג **מספרו** של השיר הארוך ביותר ושל השיר הקצר ביותר שנמצאים בו. עליכם לפתח אלגוריתם שיאפשר את הצגת המידע הזה. כלומר האלגוריתם יקבל כקלט את רשימת אורכי 100 השירים במכשיר, ופלט האלגוריתם יהיה **מספרו** של השיר הארוך ביותר **ומספרו** של השיר הקצר ביותר.

כמובן שלצורך פתרון הבעיה יש למצוא את השיר הארוך ביותר ואת השיר הקצר ביותר. כלומר יש למצוא ברשימת אורכי השירים ערך מקסימלי וערך מינימלי. אבל בבעיה זו עלינו לשמור מלבד הערך המקסימלי ומלבד הערך המינימלי גם את **מיקומם** של ערכים אלה ברשימה.

פירוק הבעיה לתת-משימות

את הרעיון המרכזי בפתרון נוכל לבטא באמצעות התת-משימה הבאה אשר על ביצועה יש לחזור לאחר כל קליטת ערך נוסף:

השוואת אורך השיר האחרון שנקלט לאורך השיר הגדול ביותר שנקלט עד כה ולאורך השיר הקצר ביותר שנקלט עד כה, ובהתאם לתוצאת ההשוואה עדכון ערך המקסימום ומיקום המקסימום ועדכון ערך המינימום ומיקום המינימום.

בחירת משתנים

currentSongLength – מטיפוס ממשי, ישמור את אורך השיר הנוכחי.

longest – מטיפוס ממשי, ישמור את אורך השיר הארוך ביותר שנקלט עד כה.

shortest – מטיפוס ממשי, ישמור את אורך השיר הקצר ביותר שנקלט עד כה.

placeLongest – מטיפוס שלם, ישמור את **מיקומו** של השיר הארוך ביותר שנקלט עד כה.

placeShortest – מטיפוס שלם, ישמור את **מיקומו** של השיר הקצר ביותר שנקלט עד כה.

שימו ♥ לבחירת טיפוסי המשתנים: המשתנים longest ו-shortest אשר שומרים את האורך המקסימלי ואת האורך המינימלי חייבים להיות מאותו טיפוס כמו הערכים ברשימת הקלט. כיוון שכאן מדובר באורכי שירים (שיכולים להיות לא שלמים), בחרנו בטיפוס ממשי. לעומתם המשתנים placeLongest ו-placeShortest שומרים מיקום ברשימה, ולכן הם מטיפוס שלם.

התוכנית המלאה

? כיצד נדע מהו מיקומו של שיר נתון?

לשם כך נוכל להשתמש במשתנה הבקרה של משפט ה-`for`. כדי שהלולאה תתבצע 99 פעמים משתנה הבקרה יתקדם מ-2 עד 100, וכך ערכו של משתנה הבקרה יהיה בדיוק מספרו של השיר התורן.

שימו ♥: בדיוק כפי שנאתחל את longest ואת shortest, נאתחל גם את placeLongest ואת placeShortest באופן עקבי. Longest יאותחל באורכו של השיר הראשון ולכן בהתאם placeLongest יאותחל במיקום הראשון (כלומר ב-1). כך גם לגבי האתחול של shortest ושל placeShortest.

```
/*
קלט: רשימת אורכי 100 השירים במכשיר MP3
פלט: מספרו של השיר הארוך היותר, ומספרו של השיר הקצר ביותר
*/
using System;
public class Mp3
{
    public static void Main ()
    {
        const int NUM_OF_SONGS = 100; // הגדרת קבוע - מספר השירים בנגן
        double currentSongLength;    //אורך השיר הנוכחי
        double shortest;             // אורך השיר הקצר ביותר
        double longest;              //אורך השיר הארוך ביותר
        int placeLongest, placeShortest; // מיקומם של השיר הקצר והארוך
        Console.Write("Enter the length of the first song: ");
        currentSongLength = double.Parse(Console.ReadLine());
        // אתחולם של המקסימום, המינימום ומקומם
        longest = currentSongLength;
        shortest = currentSongLength;
        placeLongest = 1;
        placeShortest = 1;
        // הלולאה מתחילה מ-2 כיוון שהשיר הראשון כבר נקלט
        for(int i = 2; i <= NUM_OF_SONGS; i++)
        {
```

```
            Console.Write("Enter the length of song {0}: ", i);
            currentSongLength = double.Parse(Console.ReadLine());
            if (currentSongLength > longest)
            {
                longest = currentSongLength;
                placeLongest = i;
            } // if
            if (currentSongLength < shortest)
            {
                shortest = currentSongLength;
                placeShortest = i;
            } // if
        } // for
        Console.WriteLine("The number of the longest song is {0}",
                                                placeLongest);
        Console.WriteLine("The number of the shortest song is {0}",
                                                placeShortest);
    } // Main
}// Mp3
```

סוף פתרון בעיה 6

שאלה 7.25

פתחו אלגוריתם שיקבל כקלט מספר חיובי שלם המציין את מספר שחקני מכבי ת״א, ואחר כך יקלוט רשימה של נתוני קליעות של השחקנים: עבור כל שחקן ייקלט מספרו (המספר שעל החולצה שלו), ומספר הנקודות שקלע במהלך העונה. פלט האלגוריתם יהיה **מספרו** של השחקן שקלע הכי הרבה נקודות במהלך העונה.

שימו ♥: במקרה זה, אנחנו לא מתבקשים להציג את מיקומו של השחקן שקלע הכי הרבה נקודות אלא את מספרו, לכן משתנה הבקרה המצביע על ה״מיקום״ של השחקן התורן בקלט אינו מתאים כי הוא אינו מספרו הסידורי של השחקן. את מספרו של השחקן יש לקרוא מהקלט.

להעמקה בתבניות ***מציאת ערך נלווה למקסימום*** ו***מציאת ערך נלווה למינימום*** פנו לסעיף התבניות המופיע בסוף הפרק.

7.4 ביצוע-חוזר-בתנאי

לצורך פתרון הבעיות שראינו עד כה אפשר היה להשתמש בהוראה לביצוע-חוזר אשר מספר הסיבובים בה נקבע **לפני** תחילת ביצוע הלולאה.

בסעיף זה נכיר בעיות אשר בפתרונן אי אפשר לקבוע לפני תחילת הלולאה את מספר הפעמים לביצוע-חוזר. בפתרון בעיות אלה מהלך הלולאה נקבע בהתאם לתנאי. הלולאה מתבצעת שוב ושוב כל עוד התנאי מתקיים. הלולאה מסתיימת כאשר התנאי לא מתקיים.

ביצוע-חוזר בשימוש בזקיף

בסעיף זה נראה בעיות דומות לאלו שראינו בסעיפים הקודמים. הדמיון מתבטא בכך שגם בבעיות אלו מתבצע עיבוד של רשימת נתוני קלט. ההבדל נעוץ באורך הרשימה המעובדת. בבעיות שהוצגו עד כה אורך הרשימה היה קבוע של התוכנית או נקרא מהקלט. בבעיות שיוצגו בסעיף זה אורך

הרשימה אינו ידוע כלל. במקום זאת האיבר האחרון בקלט הוא איבר מיוחד המסמן את סופה של רשימת ערכי הקלט.

בעיה 7

מטרת הבעיה ופתרונה: הצגת הוראה לביצוע-חוזר-בתנאי בשימוש בזקיף.

צבי עובד בשעות אחר הצהריים בחלוקת משלוחי פרחים. הוא מעוניין להיעזר במחשב הנייד שהוא נושא עמו כדי לחשב בסיום יום העבודה את סכום הטיפים שהרוויח במהלך היום. בכל פעם שצבי מקבל טיפ הוא מקיש את הסכום שקיבל (בשקלים שלמים). בסיום יום העבודה הוא מקיש 1-. פתחו אלגוריתם המקבל כקלט את רשימת הטיפים שקיבל צבי, המסתיימת בערך 1-, ומחשב את סכום הטיפים הכולל. ישמו את האלגוריתם בשפת C#.

ניתוח הבעיה בעזרת דוגמאות

שאלה 7.26

תארו את הפלט עבור כל אחד מן הקלטים הבאים (משמאל לימין):

א. 1- 9 7 10 12

ב. 1- 20

פירוק הבעיה לתת-משימות

בניתוח ראשוני של הבעיה ניתן לראות שיש להשתמש בביצוע-חוזר של שתי התת-משימות:

1. קליטת מספר
2. הוספת המספר לסכום (בשימוש בצובר)

לו הקלט היה כולל בתחילתו את מספר הערכים בסדרה הנתונה, היינו מפתחים אלגוריתם דומה לאלגוריתמים שבסעיפים הקודמים, בשימוש בהוראה במבנה *בצע מספר פעמים*. אבל מספר הערכים בסדרה לא נתון בתחילת הקלט (כלומר אין אנו יודעים מראש כמה שליחויות עשה צבי באותו היום). במקום זה מופיע בסוף הקלט הערך 1- המציין "סוף קלט". כיצד נשתמש בסימון זה בהוראה לביצוע-חוזר?

נשתמש בערך 1- כדי להחליט על סיום ביצוע-חוזר של התת-משימות שתיארנו. כלומר לאחר קליטה של טיפ תורן, יושווה ערך הטיפ לערך 1-. אם הוא שונה מ-1- הרי הוא נתון קלט רגיל ולכן יש להוסיפו לסכום המצטבר. אחרת יש לסיים את הביצוע-החוזר.

בחירת משתנים

tip – מטיפוס שלם, ישמור את הטיפ למשלוח הנוכחי.
sum – מטיפוס שלם, ישמור את סכום הטיפים.

האלגוריתם

ננסח את הרעיון שתיארנו:

כל עוד הערך האחרון שנקלט איננו 1- בצע:
הוסף את הערך האחרון שנקלט לסכום המצטבר
הצג את ערכו של הסכום המצטבר

בהוראה לביצוע-חוזר מסוג זה נבדק תנאי לפני כל סיבוב בלולאה. במקרה זה התנאי הוא *הערך האחרון שנקלט אינו 1-*. אם התנאי מתקיים מתבצע סיבוב נוסף בלולאה. כאשר התנאי לא מתקיים, כלומר הערך שנקלט הוא אכן 1-, מסתיים הביצוע-החוזר.

? בין ההוראות האלו לא נמצאת עדיין הוראה לקליטת הטיפ התורן. היכן נמקם את הוראת הקלט?

יש לבצע את קליטת הטיפ התורן לפני השוואתו לסימן 1-. לכן את הערך הראשון יש לקרוא מהקלט עוד לפני הלולאה. קליטת הערכים הבאים (עד לקליטת סימן סוף הקלט) תהיה ההוראה האחרונה בגוף הלולאה. כלומר מיד כאשר מסתיים עיבוד איבר קלט תורן, ולפני שנבדק קיום התנאי ביחס לאיבר הקלט החדש, מתבצעת קליטה של איבר הקלט החדש.

הנה האלגוריתם המלא:

1. *אתחל את המשתנה* sum *ב-0*
2. *קלוט מספר שלם ב-*tip
3. *כל עוד* $tip \neq -1$ ***בצע:***
   - 3.1. *הוסף ל-*sum *את הערך* tip
   - 3.2. *קלוט מספר שלם ב-*tip
4. *הצג את ערכו של* sum

יישום האלגוריתם

את ההוראה במבנה ***כל עוד... בצע***, נממש ב-#C במשפט **while**. משפט **while** כולל תנאי בוליאני שקיומו קובע את המשך ביצוע הלולאה.

```
/*(-1)-קלט: סדרת מספרים שלמים חיוביים המסתיימת ב
פלט: סכומם של המספרים שנקלטו */
using System;
public class SumOfTips
{
    public static void Main ()
    {
        int tip;          // הטיפ מהמשלוח הנוכחי
        int sum;          // סכום הטיפים המצטבר
1.      sum = 0;
2.      Console.Write("Enter your first tip for today." +
                            " End the list of tips with -1: ");
3.      tip = int.Parse(Console.ReadLine());
4.      while (tip != -1)
        {
4.1.        sum = sum + tip;
4.2.        Console.Write("Enter the next tip." +
                            " End the list of tips with -1: ");
4.3.        tip = int.Parse(Console.ReadLine());
        } // while
5.      Console.WriteLine("You have earned {0} shekels", sum);
    }// Main
}// SumOfTips
```

מעקב

נעקוב אחר מהלך ביצוע התוכנית `SumOfTips` עבור הקלט 1- 3 6 12:

בטבלת מעקב הכוללת משפט **while** אנו כוללים עמודה עבור התנאי הבוליאני שבכותרת המשפט.

	המשפט לביצוע	tip	sum	tip!=-1	**פלט**
1	sum = 0;	?	0		
2	Console.Write("Enter your first tip…");	?	0		Enter your first tip …
3	tip = **int**.Parse(Console.ReadLine());	12	0		
4	**while** (tip!= -1)	12	0	**true**	
4.1	sum = sum + tip;	12	12		
4.2	Console.Write("Enter the next tip.…");	12	12		Enter the next tip. …
4.3	tip = **int**.Parse(Console.ReadLine());	6	12		
4	**while** (tip != -1)	6	12	**true**	
4.1	sum = sum + tip;	6	18		
4.2	Console.Write("Enter the next tip.…");	6	18		Enter the next tip. …
4.3	tip = **int**.Parse(Console.ReadLine());	3	18		
4	**while** (tip!= -1)	3	18	**true**	
4.1	sum = sum + tip;	3	21		
4.2	Console.Write("Enter the next tip.…");	3	21		Enter the next tip. …
4.3	tip = **int**.Parse(Console.ReadLine());	-1	21		
4	**while** (tip!= -1)	-1	21	**false**	
5	Console.WriteLine("You have earned {0} shekels", sum);	-1	21		You have earned 21 Shekels

סוף פתרון בעיה 7

נציג את המבנה החדש שהכרנו בפתרון לבעיה 7:

הוראה לביצוע-חוזר-בתנאי מבטאת ביצוע-חוזר של תת-משימה התלוי בקיום תנאי.
הוראה כזאת נכתבת במבנה *כל עוד... בצע...*

גם הוראה כזאת נקראת **לולאה**, והתנאי להמשך הביצוע-החוזר נקרא **תנאי הכניסה ללולאה**.

הוראה לביצוע-חוזר-בתנאי מיושמת ב-#C במשפט **while**.

המבנה הכללי של משפט while הוא:

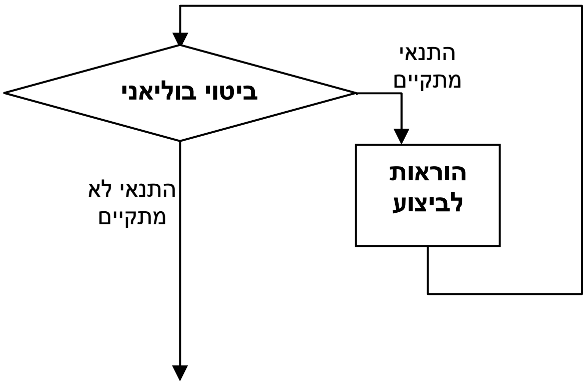
```
while (ביטוי בוליאני)
{
      גוף הלולאה: משפט או משפטים לביצוע
}
```

ביצוע משפט while מתחיל בחישוב ערכו של הביטוי הבוליאני. אם ערכו **true** מתבצע גוף הלולאה. בתום ביצוע גוף הלולאה מחושב הביטוי הבוליאני שוב. אם ערכו **true** מתבצע גוף הלולאה שוב. תהליך זה נמשך כל עוד ערכו של הביטוי הבוליאני הוא **true**. כאשר ערכו הוא **false** מסתיים ביצוע משפט ה-**while**.

גוף הלולאה תחום בסוגריים מסולסלים. במקרה שגוף הלולאה מכיל משפט בודד אפשר להשמיט את הסוגריים.

בטבלת מעקב אחר מהלך ביצוע תוכנית הכוללת משפט **while** אנו כוללים עמודה עבור תנאי הכניסה ללולאה.

ניתן לתאר משפט **while** באמצעות תרשים הזרימה הבא:



כאמור השוני בין בעיה 7 לבעיות הקודמות לה בפרק הוא בהגדרת הקלט. בבעיות הקודמות ניתן מראש אורך רשימת איברי הקלט, ואילו בבעיה 7 סיום רשימת הקלט מסומן בסימן מיוחד (1-). סימן כזה נקרא **זקיף**.

זקיף הוא נתון קלט חריג שתפקידו לסמן את סוף סדרת ערכי הקלט. הזקיף אינו חלק מסדרת הנתונים, ואין לעבד אותו כמו שאר איברי הסדרה.

למשל בפתרון בעיה 7 לא הוספנו את הערך 1- לסכום איברי הסדרה כיוון שהזקיף אינו נחשב כחלק מהסדרה!

השוני בהגדרת הקלט משפיע גם על מבנה הלולאה. הבדל ברור אחד הוא שבפתרון בעיה 7 השתמשנו בהוראה לביצוע-חוזר-בתנאי שיושמה במשפט **while**, בעוד שבפתרון הבעיות הקודמות השתמשנו בהוראות לביצוע-חוזר מספר פעמים ידוע מראש שיושמו במשפטי **for**. אך יש הבדל נוסף – האופן שמשולבות הוראות הקלט בתוך הלולאה:

בלולאות שבהן השתמשנו בפתרון בעיות קודמות, גוף הלולאה כלל הוראת קלט ומיד לאחריה הוראה לעיבוד הקלט. מבנה זה של גוף הלולאה התאים לבעיות שהיה צריך לעבד בהן כל אחד ואחד מאיברי הקלט באופן אחיד.

בבעיות דוגמת בעיה 7 עיבוד איברי הקלט אינו אחיד: העיבוד של הזקיף שונה מהעיבוד של איברים אחרים בקלט. דרוש מבנה המאפשר לקרוא איבר קלט ולעבד אותו רק אחרי שבוצעה בדיקה כי הוא אינו הזקיף. לכן בפתרון בעיות שהגדרת הקלט בהן כוללת זקיף, מתבצעת הוראת קלט ראשונה לפני הלולאה. גוף הלולאה כולל קודם כל הוראה לעיבוד הקלט, ורק אחר כך הוראת קלט.

המבנה הכללי של הוראה לביצוע-חוזר לעיבוד רשימת קלט המסתיימת בזקיף:

```
קלוט נתון
כל עוד הנתון הנקלט אינו הזקיף בצע:
    עבד את הנתון הנקלט
    קלוט נתון
```

באופן זה, הנתון החדש ייבדק תמיד **מיד** לאחר קליטתו.

שאלה 7.27

נתון קטע התוכנית הבא. הקלט הוא רשימת תווים המהווים מילה באנגלית ובסופה הזקיף '*'.

```
int i = 0;
char letter;
Console.Write("Enter the first letter of the word: ");
letter = char.Parse(Console.ReadLine());
while (letter != '*')
{
    i = i + 1;
    Console.Write("Enter the next letter of the word: ");
    letter = char.Parse(Console.ReadLine());
}
Console.WriteLine(i);
```

מהי מטרת קטע התוכנית?

שאלה 7.28

פתחו אלגוריתם אשר הקלט שלו הוא רשימת ציונים (נתונים כמספרים שלמים) בין 0 ל-100 אשר בסופה הזקיף 101, והפלט שלו הוא מספר הציונים ברשימה הגדולים או שווים ל-60. ישמו את האלגוריתם בשפת C#.

שאלה 7.29

אלון ובני מתחרים על תפקיד יושב ראש ועדת קישוט של הכיתה, מועמד זוכה אם **יותר** מחצי מן הבוחרים הצביעו עבורו. פתחו וישמו אלגוריתם אשר הקלט שלו הוא סדרה של התווים A ו-B (A עבור אלון, B עבור בני), המבטאת את קולות הבוחרים, ומסתיימת בזקיף '#'. הפלט שלו הוא הודעה אם אלון זכה או לא זכה ברוב קולות (כלומר ביותר מחצי מקולות הבוחרים).

למשל, עבור הקלט ABBAABBAA# הפלט המתאים הוא: Alon wins, ועבור הקלט ABBABBAA# הפלט המתאים הוא: Alon didn't win.

להעמקה בתבנית ***איסוף בקיזוז*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 7.30

בבחירות לוועד חיות היער מועמדות שלוש חיות: העכבר שמספרו 1, האריה שמספרו 2, והנמלה שמספרה 3. פתחו אלגוריתם אשר הקלט שלו הוא רשימת קולות הבוחרים (כל קול הוא אחד מן המספרים 1, 2 או 3) ובסופה הזקיף 0. הפלט הוא הודעה המתארת עד כמה צריכות חיות היער להיזהר: אם האריה קיבל פחות מ-30% מהקולות עליהן להיזהר מאוד, אם האריה קיבל בין 31% ל-70% מהקולות עליהן להיזהר קצת, ואם האריה קיבל יותר מ-71% מהקולות הן יכולות להיות רגועות ואינן צריכות להיזהר כלל. ישמו את האלגוריתם בשפת C#.

שאלה 7.31

פתחו אלגוריתם אשר הקלט שלו הוא סדרת תווים המהווה מילה באנגלית, ומסתיימת בזקיף '*'. הפלט שלו הוא מילת הקלט והיא מוצפנת באופן הבא: כל אות במילה מוחלפת באות העוקבת לה "בצורה מעגלית" ב-א"ב האנגלי (כלומר, כל אות מלבד האות Z מוחלפת באות העוקבת לה, והאות Z מוחלפת באות A). למשל עבור הקלט *ZEBRA הפלט הוא AFCSB.
ישמו את האלגוריתם בשפת C#.

שאלה 7.32

טורניר השש-בש מתחיל, אך איבדתם את הקוביות! פתחו אלגוריתם אשר מדמה הטלת **שתי** קוביות ומציג את תוצאות ההטלה. האלגוריתם מדמה את ההטלות עד שתוצאת ההטלה היא "שש-בש", כלומר צמד המספרים 5 ו-6 או 6 ו-5. לסיום האלגוריתם מציג כפלט את מספר ההטלות שהתבצעו עד שהתקבלה התוצאה שש-בש. ישמו את האלגוריתם בשפת C#.

ביצוע-חוזר עם תנאי כניסה כלשהו

בעיה 8

מטרת הבעיה ופתרונה: הצגת שימוש בהוראה לביצוע-חוזר-בתנאי כניסה כלשהו שאינו תלוי בזקיף.

בתוכנית "הנוסע המתמיד" של חברת התעופה "שחקים" ניתן לצבור מרחקי טיסות. נוסע אשר צובר למעלה מ-3000 קילומטרים זוכה בכרטיס טיסה חינם להונולולו.

פתחו וישמו אלגוריתם אשר הקלט שלו הוא רשימת מרחקי הטיסות של נוסע אשר זכה בכרטיס חינם (כלומר ידוע שכבר צבר יותר מ-3000 ק"מ). הפלט שלו הוא: מספר המרחקים המופיעים ברצף מתחילת הרשימה אשר סכומם המצטבר עולה על 3000, ומספר הקילומטרים שנשארו לנוסע מעבר ל-3000 כְּיִתְרָה לצבירות הבאות.

למשל עבור הקלט: 500 150 700 1000 800 הפלט המתאים הוא 150 5, משום שסכומם של כל חמשת המרחק ים עולה על 3000, ואחרי שמופחת מסכום המרחקים הערך 3000 נשארת יתרה של 150.

ניתוח הבעיה בעזרת דוגמאות

שאלה 7.33

מהו הפלט המתאים עבור כל אחד מן הקלטים הבאים:

א. 200 1000 800 600 600.

ב. 1000 1200 800 900 .

? ברור שבפתרון הבעיה יש לצבור את המרחקים הנתונים. כלומר יש לבצע לולאה שבה ייקלט מרחק נתון ויתווסף לצובר. אך כמה פעמים יש לבצע זאת?

בניגוד לבעיות בסעיפים הקודמים שאורך רשימת איברי הקלט בהם היה נתון, או שניתן לנו סימן מיוחד לסיום הקלט (זקיף), בבעיה זו יש לקלוט מרחקים ולהוסיפם לצובר כל עוד ערכו של הצובר איננו גדול מ-3000. ברגע שערכו של הצובר גדול מ-3000 אין צורך להמשיך בפעולות הקליטה והצבירה. מכאן נובע שיש לבצע את פעולות הקליטה והצבירה רק כל עוד מתקיים התנאי הבא:

המרחק הנצבר קטן או שווה ל-3000

פירוק הבעיה לתת-משימות

כל עוד המרחק הנצבר קטן או שווה ל-3000 יש לחזור על ביצוע התת-משימות הבאות:

קליטת מרחק נתון

הוספת המרחק הנתון לצובר

הגדלה ב-1 של "מונה המרחקים"

המונה המוזכר בתת-משימה האחרונה ימנה את מספר המרחקים שנצברו בצובר. כאשר יסתיים ביצוע הלולאה, ישמור מונה זה את מספר המרחקים שנצברו עד שהסכום המצטבר עלה על 3000.

בחירת משתנים

dist – מטיפוס שלם, ישמור מרחק תורן בקלט
sum – מטיפוס שלם, צובר שישמור את סכום המרחקים המצטבר
counter – מטיפוס שלם, מונה שישמור את מספר המרחקים שנצברו

האלגוריתם

1. *אתחל את* sum *ב-0*
2. *אתחל את* counter *ב-0*
3. ***כל עוד*** sum *קטן או שווה ל-3000* ***בצע:***
   3.1 *קלוט את המרחק ב-*dist
   3.2 *הוסף את* dist *ל-*sum
   3.3 *הגדל ב-1 את* counter
4. *הצג את ערכו של* counter
5. *הצג את ההפרש* sum-*3000*

שאלה 7.34

א. ישמו את האלגוריתם בשפת C#.

ב. בנו טבלת מעקב אחר ביצוע התוכנית (לפתרון בעיה 8) עבור הקלט:

500 1200 800 300 300

כמה פעמים יתבצע גוף הלולאה?

שימו ♥: בכל ביצוע-חוזר של הלולאה נקלט מרחק טיסה אחד, ולכן הערך השמור במשתנה counter הוא בעצם מספר הפעמים שמתבצעת הלולאה.

סוף פתרון בעיה 8

שאלה 7.35

נסחו תנאי כניסה בוליאני מתאים עבור כל אחד מן התיאורים הבאים של ביצוע-חוזר:

א. סכימת משקלי מכוניות (לצורך מעבר במעבורת) **כל עוד** הסכום אינו עולה על 100 טון.

ב. סכימת המספרים החיוביים השלמים המתחילים ב-1 **עד אשר** הסכום גדול מהערך הנתון כקלט.

ג. קריאת אותיות **עד אשר** נקלטות 10 אותיות A.

שימו ♥: בסעיף א מתואר הביצוע-החוזר במתכונת של **כל עוד** ובסעיפים ב ו-ג מתואר הביצוע-החוזר במתכונת של **עד אשר**.

שאלה 7.36

בכל אחד מן הסעיפים הבאים מופיע קטע תוכנית הכולל משפט **while** ובו תנאי הכניסה חסר. השלימו את תנאי הכניסה לפי מטרת קטע התוכנית, ובנו טבלת מעקב אחר ביצוע הקטע השלם.

א. מטרת הקטע: הצגה של המספר הזוגי הקטן ביותר אשר גדול מנתון הקלט בהינתן שהקלט הוא מספר חיובי.

```
Console.Write("Enter a number: ");
num = int.Parse(Console.ReadLine());
i = 0;
```

```
while (_____________)
    i = i + 2;
Console.WriteLine(i);
```

בנו טבלת מעקב אחר ביצוע קטע התוכנית השלם עבור הקלט 9.

ב. מטרת הקטע: הצגה של מכפלת שני נתוני הקלט, בהינתן שהם מספרים שלמים חיוביים.

```
Console.Write("Enter a number: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter a number: ");
y = int.Parse(Console.ReadLine());
counter = 0;
sum = 0;
while (______________)
{
    sum = sum + x;
    counter = counter + 1;
}
Console.WriteLine(sum);
```

בנו טבלת מעקב אחר ביצוע קטע התוכנית השלם עבור הקלט 3 4.

שאלה 7.37

נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר חיובי שלם:

```
Console.Write("Enter a number: ");
limit = int.Parse(Console.ReadLine());
s = 0;
c = 0;
while (s < limit)
{
    c = c + 1;
    s = s + c;
}
Console.WriteLine(c);
```

קטע התוכנית כולל מונה `c` וצובר `s`. הצובר צובר את ערכיו של המונה במהלך הביצוע-החוזר.

א. מהו הפלט עבור הקלט 1? מהו הפלט עבור הקלט 11?

ב. ציינו שני קלטים שונים שעבורם יהיה הפלט 5. מהו מספר הפעמים לביצוע-החוזר עבור קלטים אלה?

ג. מהי מטרת קטע התוכנית?

היעזרו בטבלת מעקב כדי לענות על סעיפים א ו-ב.

שאלה 7.38

נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר חיובי שלם, ו-`TOP_LIMIT` הוא קבוע שערכו 100:

```
Console.Write("Enter a number: ");
num = int.Parse(Console.ReadLine());
mult  = 1;
i = 0;
while ((i < num) && (mult < TOP_LIMIT))
{
    i = i + 1;
    mult = mult * i;
```

```
}
Console.WriteLine(mult);
```

קטע התוכנית הנתון כולל צובר `mult` אשר צובר ערך של מכפלה, והוא מאותחל בערך 1.

א. מהו הפלט עבור הקלט 2 ? ומהו הפלט עבור הקלט 5?

ב. מהו הקלט שעבורו יהיה הפלט 24? ומהו מספר הפעמים של הביצוע-החוזר עבור קלט זה?

ג. מהי מטרת קטע התוכנית?

היעזרו בטבלת מעקב כדי לענות על סעיפים א ו-ב.

שאלה 7.39

פתחו אלגוריתם אשר הקלט שלו הוא מספר חיובי שלם, והפלט שלו הוא החזקה הקטנה ביותר של 2 אשר גדולה מנתון הקלט. למשל: עבור הקלט 7 הפלט הדרוש הוא 8 (כי $2^3=8$), ועבור הקלט 8 הפלט הדרוש הוא 16 (כי $2^4=16$). ישמו את האלגוריתם בשפת C#.

במהלך הפיתוח הקפידו על ניסוח תת-משימות לביצוע-חוזר ועל ניסוח תנאי לביצוע-חוזר.

שימו ♥: באלגוריתם זה יש להשתמש בצובר של מכפלה, ולא בפעולת החזקה Pow המוגדרת במחלקה Math.

בלולאות **`while`** שראינו עד כה תנאי הכניסה היו דומים למדי זה לזה. הם כללו תמיד השוואה של משתנה, אשר ערכו גדל במהלך הביצוע-חוזר של הלולאה, לחסם אשר נשמר במשתנה או נקבע מפורשות. הביצוע-החוזר הסתיים כאשר ערכו של המשתנה גדל ועבר את החסם (או השתווה לו). אבל תנאי כניסה כאלה הם רק סוג אחד של תנאי כניסה ללולאה. בפתרון הבעיה הבאה נראה דוגמה לתנאי כניסה מסוג אחר.

בעיה 9

מטרת הבעיה ופתרונה: הצגת תבנית פירוק מספר שלם חיובי כלשהו לספרותיו.

> פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי והפלט שלו הוא מספר ספרות המספר. למשל: עבור ה קלט 31 הפלט הדרוש הוא 2 ועבור הקלט 15568 הפלט הדרוש הוא 5.

בפרקים קודמים כבר עסקנו בפירוק של מספר שלם לספרותיו, אך שם מספר ספרותיו היה קטן וידוע. בבעיה הנתונה מספר הספרות איננו ידוע ויכול להיות גדול מאוד.

פירוק הבעיה לתת-משימות

מה יהיו התת-משימות לביצוע-חוזר עבור מניית ספרות המספר? ומה יהיה התנאי לחזרה על ביצוע תת-משימות אלה?

נוכל "לקצץ" את ספרות המספר אחת אחת ולמנות את מספר הספרות הנקצצות. נעשה זאת באמצעות תת-משימות לביצוע-חוזר הכוללות קיצוץ ספרה מן המספר והגדלת מונה ב-1. קיצוץ ספרה יתבצע באמצעות חלוקה בשלמים של המספר ב-10. חלוקה זו תביא לקיצוץ ספרת האחדות (הספרה הימנית ביותר). למשל במקום המספר 534 יתקבל המספר 53.

אם כך, ננסח את התת-משימות הבאות לביצוע-חוזר:

1. קיצוץ ספרת האחדות
2. הגדלה ב-1 של מונה הספרות

יש לבצע תת-משימות אלו כל עוד "יש מה לקצץ", כלומר כל עוד המספר הנותר במהלך הביצוע-החוזר גדול מ-0. לכן התנאי לביצוע-חוזר יהיה: *המספר הנותר גדול מ-0.*

בחירת משתנים

נבחר שני משתנים מטיפוס שלם:

num – ישמור את המספר הניתן כקלט, וספרותיו נקצצות

digits – מונה שישמור את מספר הספרות שנקצצו

יישום האלגוריתם

```
/*
קלט: מספר חיובי שלם
פלט: מספר הספרות במספר הנתון
*/
using System;
public class DigitCount
{
    public static void Main ()
    {
        int digits = 0;  // מספר הספרות
        int num;         // המספר המעובד
        Console.Write("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        while (num > 0)
        {
            num = num / 10;
            digits++;
        }
        Console.WriteLine("The number of digits is {0}", digits);
    }// Main
}//DigitCount
```

סוף פתרון בעיה 9

להעמקה בתבניות ***פירוק מספר לספרותיו*** ו***בניית מספר*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 7.40

א. שנו את התוכנית `DigitCount` לפתרון בעיה 9, כך שהפלט יכלול את **סכום** ספרות המספר, למשל עבור הקלט 153 הפלט המתאים הוא 9.

ב. שנו את התוכנית `DigitCount` לפתרון בעיה 9, כך שהפלט יכלול רק את **מספר** הספרות **האי-זוגיות** במספר הנתון. למשל עבור הקלט 150 הפלט המתאים הוא 2.

ג. שנו את התוכנית `DigitCount` לפתרון בעיה 9, כך שהפלט יכלול רק את **מכפלת** ספרות המספר. שימו לב לאתחול נכון של המשתנה השומר את המכפלה.

שאלה 7.41

נתון קטע התוכנית החלקי הבא, אשר הקלט שלו הוא שני מספרים שלמים חיוביים `x` ו-`y` ומטרתו היא הצגת שארית החלוקה בשלמים של `x` ב-`y` (כלומר תוצאת החישוב `x%y`). בקטע התוכנית מתבצע החישוב הדרוש באמצעות פעולת **חיסור**.

השלימו את קטע התוכנית.

```
Console.Write("Enter a number: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter a number: ");
y = int.Parse(Console.ReadLine());
while (____________________)
    x = x - y;
Console.WriteLine(__________________________);
```

שאלה 7.42

שנו את הקטע הנתון בשאלה הקודמת כך שיציג גם את מנת החלוקה של x ב-y, (כלומר את תוצאת החישוב x/y). יש לבצע את החישובים הדרושים באמצעות פעולות חיבור וחיסור בלבד!

שאלה 7.43

שני תלמידים המשחקים זה נגד זה מותחים בתחילת המשחק קו באורך N סנטימטרים ($N > 1$). השחקנים מחליפים תורות לסירוגין. כל שחקן מקצר בתורו את הקו לחצי מאורכו. השחקן אשר מקצר בתורו את הקו לאורך של פחות מסנטימטר אחד מנצח במשחק.

למשל אם אורכו של הקו הוא 8 ס"מ, השחקן הראשון יקצר את הקו ל-4 ס"מ והשחקן השני יקצר את הקו ל-2 ס"מ, השחקן הראשון יקצר את הקו לס"מ אחד והשחקן השני יקצר את הקו ל-0.5 ס"מ וינצח במשחק.

פתחו וישמו אלגוריתם אשר הקלט שלו הוא אורכו ההתחלתי של הקו, והפלט שלו הוא הודעה מיהו השחקן המנצח (הראשון או השני).

שאלה 7.44

במשחק אסימונים שחקן מניח 2 אסימונים בתור הראשון, 4 אסימונים בתור השני, 8 אסימונים בתור השלישי, וכך הלאה – בכל תור מוכפל מספר האסימונים.

נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר האסימונים ההתחלתי של השחקן, והפלט שלו צריך להיות המספר הסידורי של התור אשר בו לא ניתן להמשיך לשחק לפי השיטה המתוארת. למשל, עבור הקלט המייצג מספר אסימונים התחלתי 9, הפלט הדרוש הוא 3, כיוון שאחרי שהניח 2 אסימונים בתור הראשון ו-4 אסימונים נוספים בתור השני יוותרו לשחקן רק 3 אסימונים (ולא 8 אסימונים, כפי שנדרש לתור השלישי). קטע התוכנית שגוי.

```
int turn = 0;                   //המספר הסידורי של התור הנוכחי
int tokensInCurrentTurn = 0;  //מספר אסימונים למשחק בתור הנוכחי
int totalPlayedTokens = 0; // סכום אסימונים כולל בתורות שהתבצעו עד כה
int startTokens;              //מספר אסימונים התחלתי
Console.Write("Enter number of tokens to start with: ");
startTokens = int.Parse(Console.ReadLine());
while (totalPlayedTokens <= startTokens)
{
    turn = turn + 1;
    tokensInCurrentTurn = tokensInCurrentTurn * 2;
    totalPlayedTokens = totalPlayedTokens + tokensInCurrentTurn;
}
Console.WriteLine("The game cannot go on after {0} turns", turn);
```

ציינו מהי השגיאה ותקנו אותה.

שאלה 7.45

בצלחת פֶּטְרִי החיידקים מכפילים את עצמם פי 5 בכל דקה עד אשר מספרם עובר סף מסוים הנקרא "סף ההכפלה".

פתחו אלגוריתם אשר הקלט שלו הוא שני נתונים: מספר חיידקים התחלתי בצלחת וסף ההכפלה. הפלט שלו הוא מספר החיידקים שיהיו בצלחת בדקה שבה יעבור מספרם את סף ההכפלה. ישמו את האלגוריתם בשפת C#.

ראינו עד כה את המבנים *בצע מספר פעמים* (או *עבור כל... בצע*)... ו-*כל עוד... בצע*... לביצוע-חוזר של תת-משימה. לעתים נוהגים לכנות את המבנה הראשון בשם "לולאת **for**" ואת המבנה השני בשם "לולאת **while**". מתי נבחר להשתמש במבנה הראשון ומתי נעדיף את המבנה השני?

- כאשר אפשר לחשב לפני ביצוע לולאה את מספר הפעמים שתתבצע, נעדיף לשם הנוחות ולשם הבהירות להשתמש בלולאת **for**.
- כאשר אי אפשר לחשב לפני ביצוע לולאה את מספר הפעמים שתתבצע, בין שהסיום נשלט בידי זקיף ובין שנשלט בידי תנאי אחר, נשתמש בלולאת **while**. המבנה של לולאת **while** מבהיר כי ביצועה תלוי בתנאי, והתנאי עצמו קל לזיהוי.

בחירה נכונה בהוראה לביצוע-חוזר מבהירה לקורא התוכנית אם מספר הפעמים לביצוע ידוע מראש, ואם לא, על פי מה הוא נקבע. לכן בחירה נכונה של הוראה לביצוע-חוזר מסייעת לקריאות ולבהירות התוכנית.

שאלה 7.46

כתבו לולאה להצגת פלט של 50 כוכביות.

א. כתבו לולאת **for** להצגת הפלט הדרוש.

ב. כתבו לולאת **while** להצגת הפלט הדרוש.

ג. איזה מן הפתרונות פשוט יותר?

שאלה 7.47

א. כתבו לולאת **for** המציגה כפלט את 20 הכפולות החיוביות הראשונות של 5 (כלומר הפלט הוא: 100 ... 15 10 5).

ב. כתבו לולאת **for** נוספת המבצעת אותו דבר אבל בעלת כותרת שונה וגוף לולאה שונה.

ג. כתבו לולאת **while** המבצעת אותו דבר.

שאלה 7.48

ציינו עבור כל אחת מן הבעיות הבאות אם לפתרונה מתאים יותר להשתמש בלולאת **for** או בלולאת **while**. נמקו את תשובותיכם.

א. הקלט: מספר שלם חיובי num. הפלט: רשימת המספרים השלמים **החיוביים** מ-1 עד num.

ב. הקלט: מספר שלם חיובי num. הפלט: רשימת המספרים השלמים **השליליים** מ-1- עד num-.

ג. הקלט: מספר שלם חיובי num. הפלט: רשימת המספרים השלמים החיוביים מ-1 עד המספר הקטן ביותר k אשר עבורו מתקיים $1+2+3+...+k > num$.

שתי השאלות הבאות מתייחסות להשוואת אותיות באלף-בית האנגלי. נגדיר סדר מילוני של אותיות באופן הבא: אות אחת קטנה מאות אחרת אם האחת מופיעה לפני האחרת בסדר הא"ב. אם האות מופיעה אחרי האחרת בסדר הא"ב אז היא גדולה ממנה. (למשל B קטנה מ-E, ו-F גדולה מ-C).

שאלה 7.49

עבור כל אחת מן הבעיות הבאות ציינו אם לפתרונה מתאים יותר להשתמש בלולאת **for** או בלולאת **while**. נמקו בקצרה את תשובותיכם.

שימו ♥: בעיות אלו מתאימות לתבניות מקסימום, מינימום, מקום המקסימום ומקום המינימום, אך לא תמיד גודל התחום שמופעלת בו התבנית ידוע מראש.

א. הקלט הוא סדרת אותיות מהאלף-בית ובסופה '*', והפלט הוא האות הגדולה ביותר שמופיעה בקלט.

ב. הקלט הוא מספר שלם חיובי המציין אורך סדרה ואחריו סדרת אותיות מהאלף-בית באורך המצוין, והפלט הוא האות הקטנה ביותר שמופיעה בקלט.

ג. הקלט הוא כמו הקלט לסעיף ב, והפלט הוא המקום הסידורי של ההופעה הראשונה (אולי יש יותר מאחת) של האות הגדולה ביותר שמופיעה בקלט.

ד. הקלט הוא כמו הקלט לסעיף ב, והפלט הוא המקום הסידורי של ההופעה האחרונה (אולי יש יותר מאחת) של האות הקטנה ביותר שמופיעה בקלט.

ה. מהם ההבדלים ביישום האלגוריתם בין סעיף ג ל-ד?

שאלה 7.50

פתחו אלגוריתם אשר הקלט שלו הוא סדרת אותיות מן האלף-בית האנגלי שמסתיימת ב-'*', והפלט שלו הוא האות **הגדולה** ביותר מבין האותיות B עד I המופיעות בקלט.

למשל עבור הקלט *SCHOOL הפלט המתאים הוא H. אמנם יש בקלט אותיות גדולות מ-H, למשל S, אך אותיות אלו לא כלולות בסדרת האותיות B עד I.

הניחו שבקלט מופיעה לפחות אות אחת בתחום B עד I.

ישמו את האלגוריתם בשפת #C.

ביצוע-חוזר אינסופי

כאמור, ההוראות לביצוע-חוזר שהוצגו בשלושת הסעיפים הראשונים היו תמיד באורך ידוע מראש. כמובן עבור לולאות כאלו התשובה לשאלה "כמה פעמים יתבצע גוף הלולאה" היא פשוטה מאוד. אבל כאשר מדובר בהוראה לביצוע-חוזר-בתנאי, התשובה לשאלה זו אינה תמיד פשוטה, כפי שמדגימה הבעיה הבאה.

בעיה 10

מטרת הבעיה ופתרונה: דיון בחישוב מספר הפעמים שמתבצעת לולאה, והצגת לולאה אינסופית.

נתונה התוכנית הבאה (מטרתה מתוארת בהערה בתחילת התוכנית):

```
/*
קלט: מספר חיובי שלם
פלט: כל המספרים האי-זוגיים החיוביים הקטנים מן המספר הנתון
*/
using System;
public class OddNumbers
{
     public static void Main ()
     {
          int limit; // המספר הנתון
          int oddN = 1; //המספר האי-זוגי החיובי הראשון
```

```
 1.      Console.Write("Enter a number: ");
 2.      limit = int.Parse(Console.ReadLine());
 3.      Console.WriteLine("The odd numbers that are smaller " +
                                "than the given number are: ");
 4.      while (oddN != limit)
         {
 4.1.        Console.Write("{0} ",oddN);
 4.2.        oddN = oddN + 2;
         }
     } //Main
}// OddNumbers
```

חשבו את מספר הפעמים שתתבצע הלולאה שבתוכנית עבור קלט נתון L.

ניתוח הבעיה בעזרת דוגמאות

כדי לבטא בצורה כללית (עבור קלט L) את מספר הפעמים שתתבצע לולאת התוכנית נחשב את מספר הפעמים לביצוע הלולאה עבור דוגמאות קלט שונות:

עבור הקלט 1 לא יתבצע גוף הלולאה אפילו פעם אחת, כיוון שערכי `limit` ו-`oddN` שווים כבר בפעם הראשונה שמחושב תנאי הכניסה ללולאה.

עבור הקלט 5, נבחן את מספר הפעמים של ביצוע הלולאה באמצעות טבלת מעקב:

	המשפט הבא לביצוע	oddN	limit	oddN!=limit	**פלט**
3	Console.WriteLine("The odd Numbers …");	1	5		The odd Numbers…
4	**while** (oddN != limit)	1	5	**true**	
4.1	Console.Write("{0} ",oddN);	1	5		1
4.2	oddN = oddN + 2;	3	5		
4	**while** (oddN != limit)	3	5	**true**	
4.1	Console.Write("{0} ",oddN);	3	5		3
4.2	oddN = oddN + 2;	5	5		
4	**while** (oddN != limit)	5	5	**false**	

עבור הקלט 5 יתבצע גוף הלולאה פעמיים ויוצג הפלט: 3 1.

באופן דומה נוכל לראות שעבור הקלט 15 יתבצע גוף הלולאה 7 פעמים, ויוצג הפלט:

1 3 5 7 9 11 13

? ננסה להכליל על פי הדוגמאות שבחנו את מספר הפעמים שהלולאה תתבצע עבור קלט אי-זוגי שערכו L. מהו הביטוי הכללי?

עבור קלט אי-זוגי שערכו L הלולאה תתבצע (L-1)/2 פעמים.

? ניסחנו ביטוי כללי של מספר הפעמים לביצוע הלולאה עבור קלטים אי-זוגיים. האם קיים ביטוי כללי דומה עבור קלטים זוגיים?

נבחן את מספר הפעמים לביצוע הלולאה עבור הקלט 2 באמצעות טבלת מעקב:

	המשפט הבא לביצוע	oddN	limit	oddN!=limit	**פלט**
3	Console.WriteLine("The odd Numbers …");	1	2		The odd Numbers…
4	**while** (oddN != limit)	1	2	**true**	

4.1	`Console.Write("{0} ",oddN);`	1	2		1
4.2	`oddN = oddN + 2;`	3	2		
4	**while** (oddN != limit)	3	2	**true**	
4.1	`Console.Write("{0} ",oddN);`	3	2		3
4.2	`oddN = oddN + 2;`	5	2		
4	**while** (oddN != limit)	5	2	**true**	

קטענו את מילוי הטבלה לפני סיום ביצוע המעקב המלא. ניתן לראות שעבור הקלט 2 מוצג הפלט 3 אשר אין להציגו, כיוון שהוא מספר אי-זוגי שאיננו קטן מן הקלט!

כאשר מחושב הביטוי הבוליאני `oddN != limit` (תנאי הכניסה ללולאה) בפעם השנייה, ערכו של `oddN` הוא 3, וערכו של `limit` הוא 2, ולכן ערכו של הביטוי הבוליאני הוא **true** ומוצג הערך 3 כפלט. ערכו של `OddN` גדל ב-2 בכל סיבוב בלולאה, ולכן הלולאה תתבצע גם פעם שלישית ויוצג הפלט 5. בעצם, הלולאה תתבצע שוב ושוב וערכו של הביטוי הבוליאני יהיה **תמיד true**, כיוון שערכו של `oddN` רק ילך ויגדל. מכאן נובע שהלולאה תתבצע **אינסוף** פעמים.

הלולאה תתבצע אינסוף פעמים גם עבור הקלט 4 וגם עבור הקלט 6, ובעצם עבור **כל** קלט זוגי. זאת משום שעבור קלט זוגי יתקיים תנאי הכניסה ללולאה שוב ושוב ואף פעם לא יהיה מצב של שוויון של ערכו של `limit` (שהוא מספר זוגי) וערכו של `oddN` (שיהיה תמיד מספר אי-זוגי).

כיוון שעבור קלט זוגי הלולאה תתבצע אינסוף פעמים יוצגו כפלט מספרים אי-זוגיים שאין להציגם, ולכן לא רק שהתוכנית אינה מסתיימת, היא גם מציגה פלט שגוי!

? כיצד ניתן לתקן את התוכנית ולטפל בכך שהלולאה תתבצע מספר מתאים של פעמים גם עבור קלט זוגי?

אפשר לשנות את תנאי הכניסה ללולאה ל-`oddN < limit`, וכך ביצוע הלולאה יסתיים לאחר הצגת המספר האי-זוגי הגדול ביותר שעדיין קטן מהקלט.

שאלה 7.51

תקנו את התוכנית `OddNumbers` כך שתסתיים לאחר הצגת כל המספרים האי-זוגיים החיוביים שקטנים מן המספר הנתון.

סוף פתרון בעיה 10

התוכנית שהוצגה בבעיה 10 כללה לולאה אשר התבצעה כדרוש עבור כל קלט אי-זוגי, אך ביצועה נמשך אינסוף פעמים עבור כל קלט זוגי.

> לולאה אשר מתבצעת אינסוף פעמים עבור קלט כלשהו נקראת **לולאה אינסופית**. בחומר הלימוד של "יסודות מדעי המחשב" תוכניות הכוללות לולאה אינסופית נחשבות כתוכניות שגויות.

בפיתוח אלגוריתם קורה לעתים שנכתבת לולאה אינסופית. לולאה כזו עלולה להתבצע אינסוף פעמים רק עבור חלק מן הקלטים. לכן חשוב להקפיד לבדוק לולאת אלגוריתם עבור בחירה ממצה של **דוגמאות קלט מייצגות**, כפי שעשינו בפתרון בעיה 10. בפתרון זה בדקנו תחילה את הלולאה עבור דוגמה של קלט אי-זוגי, וראינו שעבורה מושגת המטרה. אחר כך בדקנו עבור דוגמה של קלט זוגי ונוכחנו שהלולאה אינסופית.

חשבו: מה היה קורה אילו הקלדנו ערך שלילי כקלט לתוכנית?

שאלה 7.52

ציינו עבור כל אחת מן הלולאות הבאות אם היא לולאה אינסופית.

אם הלולאה סופית ציינו את מספר הפעמים שהיא תתבצע. היעזרו בטבלת מעקב לביצוע החישובים הדרושים.

א.

```
int i = 0;
while (i < 30)
    i = i + 4;
```

ב.

```
int j = 0;
while (j < 50)
    j = j - 10;
```

ג.

```
int j = 0;
while (Math.Abs(j) < 30)
    j = j - 10;
```

ד.

```
int k = 0;
for (int j = 1; j < 10; j++)
    k = k + 11;
```

ה. num הוא מספר שלם חיובי כלשהו

```
while (num != 10)
{
    Console.WriteLine(num);
    num = num + 1;
}
```

ו. num הוא מספר שלם חיובי כלשהו

```
while (num != 0)
{
    Console.WriteLine(num);
    num = num - 1;
}
```

שאלה 7.53

נתון קטע התוכנית הבא אשר הקלט שלו הוא מספר שלם חיובי:

```
Console.Write("Enter a number: ");
num = int.Parse(Console.ReadLine());
while (num != 0)
{
    num = num % 3;
    Console.WriteLine(num);
}
```

הלולאה בקטע התוכנית תתבצע מספר סופי של פעמים עבור חלק מן הקלטים, ומספר אינסופי של פעמים עבור שאר הקלטים. לכן הלולאה שבקטע התוכנית היא לולאה אינסופית.

א. תנו דוגמת קלט שעבורה תתבצע הלולאה מספר סופי של פעמים.
מה מאפיין את הקלטים שעבורם תתבצע הלולאה מספר סופי של פעמים?
כמה פעמים תתבצע הלולאה עבור קלטים אלה?

ב. תנו דוגמת קלט שעבורה תתבצע הלולאה מספר אינסופי של פעמים.
מה מאפיין את הקלטים שעבורם תתבצע הלולאה מספר אינסופי של פעמים?

שאלה 7.54

נתון קטע התוכנית הבא אשר הקלט שלו הוא שני מספרים שלמים:

```
Console.Write("Enter a number: ");
x = int.Parse(Console.ReadLine());
Console.Write("Enter a number: ");
y = int.Parse(Console.ReadLine());
while (x != y)
{
    y = y - 1;
    x = x + 1;
}
Console.WriteLine(x);
```

א. תנו שתי דוגמאות קלט שונות שעבורן **לא יתבצע** גוף הלולאה.

ב. תנו שתי דוגמאות קלט שונות שעבורן יתבצע גוף הלולאה **בדיוק פעם אחת**.

ג. תנו שתי דוגמאות קלט שונות שעבורן יתבצע גוף הלולאה **בדיוק חמש פעמים**.
ד. תנו שתי דוגמאות קלט שונות שעבורן יתבצע גוף הלולאה **אינסוף** פעמים.
ה. נניח שערכו של x הוא 0. עבור אילו ערכים של y יתבצע גוף הלולאה **אינסוף** פעמים?

7.5 משתנים מטיפוס בוליאני

בסעיף זה נכיר משתנים מטיפוס בוליאני, כלומר משתנים היכולים לשמור בתוכם אחד משני הערכים **true** או **false**. משתנים כאלה יכולים להשתלב בביטויים בוליאניים ולכן הם שימושיים מאוד בהוראות לביצוע-בתנאי ובהוראות לביצוע-חוזר-בתנאי, כפי שנדגים בהמשך הסעיף.

בעיה 11

מטרת הבעיה הבאה: הצגת משתנה מטיפוס בוליאני

פתחו אלגוריתם שהקלט שלו הוא מספר שלם גדול מ-1 ולאחריו רשימה של מספרים שלמים נוספים הגדולים מ-1. סדרת המספרים מסתיימת במספר 0. אם המספר הראשון שנקרא הוא זוגי, אז הפלט הוא כל המספרים מרשימת המספרים כך שכל מספר מוצג פעמיים. אם המספר הראשון שנקרא הוא אי-זוגי אז הפלט הוא כל המספרים מרשימת המספרים וכל מספר מוצג פעם אחת בלבד.
למשל עבור הקלט: 0 7 4 2 8 הפלט המתאים הוא 7 7 4 4 2 2.
ועבור הקלט: 0 7 4 2 9 הפלט המתאים הוא 7 4 2.

פירוק הבעיה לתת-משימות

ננסח רעיון לפירוק ראשוני לתת-משימות לפתרון הבעיה:

1. קליטת מספר ב-num
2. עבור כל מספר ברשימה בדיקה אם num זוגי. אם כן, הצגת המספר הנקלט פעמיים, אחרת הצגת המספר הנקלט פעם אחת

נסתכל על התת-משימה השנייה: כפי שהיא מנוסחת, עבור כל מספר ברשימה אנו בודקים שוב אם num זוגי או לא, אף על פי שבעצם התשובה לשאלה זו תהיה זהה בכל פעם.
לכן, כדאי לבדוק רק פעם אחת אם num זוגי או לא.

הנה פירוק שני לתת-משימות לפתרון הבעיה:

1. קליטת מספר ב-num
2. אם num זוגי
   2.1. עבור כל מספר ברשימה הצגת המספר הנקלט פעמיים
3. אחרת (אם num אי-זוגי)
   3.1. עבור כל מספר ברשימה הצגת המספר הנקלט פעם אחת

אמנם כעת אנו בודקים רק פעם אחת אם num זוגי או לא, אבל אם נסתכל על התת-משימה השנייה והתת-משימה השלישית נראה ששתיהן כוללות קליטה של רשימת המספרים. כלומר, באלגוריתם שהתקבל מהפירוק הזה ולאחר מכן גם בתוכנית ליישום האלגוריתם, הוראות הקלט יופיעו פעמיים.

פירוק בהיר יותר לתת-משימות הוא הפירוק הבא:

1. קליטת מספר
2. בדיקה אם המספר שנקלט זוגי ושמירת תוצאת הבדיקה
    2.1. עבור כל מספר ברשימה:
        2.1.1. אם תוצאת הבדיקה חיובית (כלומר המספר הראשון שנקלט זוגי)
            2.1.1.1. הצגת המספר הנקלט פעמיים
        2.1.2. אחרת (כלומר המספר הראשון שנקלט אי-זוגי)
            2.1.2.1. הצגת המספר הנקלט פעם אחת

בפירוק זה נבדקת הזוגיות של ערך הקלט הראשון רק פעם אחת. גם המעבר על רשימת הערכים מתואר רק פעם אחת. עבור כל ערך קלט ברשימה, מספר המופעים בפלט ייקבע לפי תוצאת הבדיקה שבוצעה קודם לכן.

בחירת משתנים

מהו טיפוס הערך המתאים לשמירת התוצאה של בדיקת הזוגיות לערך הראשון בקלט? הערכים האפשריים לתשובה לשאלה אם הערך הוא זוגי הם **כן** **ולא**. טיפוס המתאים לערכים מסוג זה הוא טיפוס **בוליאני**. כזכור מפרקים קודמים, ערך מטיפוס בוליאני יכול להיות אחד משני הערכים: **true** או **false**.

עד כה ראינו ביטויים מטיפוס בוליאני ששימשו אותנו לתיאור תנאים. אבל ניתן גם להצהיר על משתנים מטיפוס בוליאני, ולשלב גם אותם בביטויים בוליאניים. אם נשמור את התוצאה של בדיקת הזוגיות לערך הקלט הראשון במשתנה בוליאני, נוכל להשתמש במשתנה זה כתנאי בהוראה לביצוע-בתנאי שעוברת על רשימת איברי הקלט ומעבדת אותם.

אם כך נשתמש במשתנים הבאים:

num – מטיפוס שלם, ישמור את ערך הקלט הראשון
numInList – מטיפוס שלם, ישמור את ערך הקלט התורן מהרשימה
isEven – מטיפוס בוליאני, ישמור את התוצאה של בדיקת הזוגיות ל-num

האלגוריתם

מאחר שהרשימה מסתיימת בזקיף, נשתמש בהוראה לביצוע-חוזר-בתנאי התלוי בזקיף.

1. *קלוט מספר ב-*num
2. *בדוק אם* num *זוגי ושמור את תוצאת הבדיקה ב-*isEven
3. *קלוט את המספר הראשון ברשימה ב-*numInList
4. ***כל עוד*** *0* $\neq$ numList ***בצע***:
    4.1. ***אם*** *ערכו של* isEven *הוא* **true**
        4.1.1. *הצג פעמיים את* numInList
    4.2. ***אחרת***
        4.2.1. *הצג פעם אחת את* numInList
    4.3. *קלוט את המספר הבא ברשימה ב-*numInList

יישום האלגוריתם

הצהרה על משתנה מטיפוס בוליאני נעשית באמצעות שם הטיפוס **bool**. לכן ניתן להצהיר על המשתנה isEven באופן הבא:

```
bool isEven;
```

כמו בכל משתנה בתוכנית, גם במשתנים מטיפוס בוליאני אפשר לבצע השמה. הביטוי שבצד ימין של משפט ההשמה צריך להיות ביטוי בוליאני. אנו מעוניינים שהמשתנה isEven ישמור את תוצאת בדיקת הזוגיות של ערך הקלט הראשון, ולכן נוכל לכתוב את ההוראה הבאה לביצוע-בתנאי:

```
if (num % 2 == 0)
    isEven = true;
else
    isEven = false;
```

במקרה האחד אנחנו משימים במשתנה isEven את הערך הבוליאני **true** ובמקרה האחר את הערך הבוליאני **false**.

אבל מאחר שבצד ימין של משפט השמה למשתנה בוליאני אפשר לכתוב כל ביטוי בוליאני, ולאו דווקא את הביטויים הפשוטים **true** או **false**, נוכל גם לכתוב את משפט ההשמה הבא:

```
isEven = (num % 2 == 0);
```

בעקבות ביצוע ההשמה שלעיל, יושם במשתנה isEven הערך **true** אם num זוגי, ואחרת יושם בו הערך **false**.

אנו מעוניינים לשלב את המשתנה isEven בביטוי הבוליאני בהוראה לביצוע-בתנאי. נוכל לכתוב כך:

```
if (isEven == true)
```

אבל מאחר שערכו של ביטוי בוליאני הוא **true** או **false**, וכך גם ערכו של משתנה בוליאני, הרי משתנה בוליאני הוא כבר ביטוי בוליאני בעצמו (בדיוק כמו שמשתנה שלם הוא כבר ביטוי חשבוני בעצמו). לכן נוכל גם לכתוב:

```
if (isEven)
```

התוכנית המלאה

```
/* קלט: מספר שלם ולאחריו רשימת מספרים שלמים גדולים מ-1 שמסתיימת ב-0
פלט: אם המספר הראשון זוגי יוצגו המספרים ברשימה פעמיים, אחרת יוצגו
המספרים ברשימה פעם אחת בלבד */
using System;
public class IfEven
{
    public static void Main ()
    {
        int num;               //המספר הראשון בקלט
        int numInList;         //מספר תורן ברשימת הקלט
        bool isEven;           //האם המספר הראשון זוגי
        // קליטת המספר הראשון ובדיקה האם הוא זוגי
1.      Console.Write("Enter the first number: ");
2.      num = int.Parse(Console.ReadLine());
```

```
3.        isEven = (num % 2 == 0);
4.        Console.Write("Enter a number. Enter 0 to end the list: ");
5.        numInList = int.Parse(Console.ReadLine());
6.        while (numInList != 0)
          {
6.1.          if (isEven)
6.1.1.            Console.Write("{0} {1} ", numInList, numInList);
6.2.          else
6.2.1.            Console.WriteLine(numInList);
6.3.          Console.Write("Enter the next number. Enter 0 to " +
                                                 "end the list: ");
6.4.          numInList = int.Parse(Console.ReadLine());
          } // while
    }// Main
}// IfEven
```

שאלה 7.55

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `IfEven` לפתרון בעיה 11 עבור הקלט: 0 7 5 4.

סוף פתרון בעיה 11

בפתרון הבעיה השתמשנו במשתנה מטיפוס **בוליאני**:

> **משתנה בוליאני** שומר ערכים מטיפוס בוליאני, כלומר אחד משני הערכים **true** או **false**.
>
> במשתנה בוליאני ניתן להשים ערך של ביטוי בוליאני כלשהו.
>
> ניתן לשלב משתנה בוליאני בתוך תנאי בוליאני.
>
> מתאים להשתמש במשתנה בוליאני כדי לזכור תוצאה של בדיקה.
>
> ב-C# משמשת המילה השמורה **bool** **להצהרה על משתנה בוליאני.**
>
> בדומה למשתנים מטיפוסים אחרים, ניתן לבצע **אתחול למשתנה בוליאני בזמן ההצהרה**, למשל:
>
> ```
> bool boolVariable = true;
> ```

למשתנה בוליאני נוהגים לעתים לקרוא **דגל** (flag). כאשר ערכו של המשתנה **true** הוא משול לדגל מורם המסמן מעבר אפשרי. כאשר ערך המשתנה **false** הוא משול לדגל מורד המסמן כי אין אפשרות מעבר.

שאלה 7.56

בקטע התוכנית הבא המשתנה `length` שומר מספר חיובי שלם המבטא אורך רשימה. הקטע קולט רשימת תוצאות הטלה של קובייה (תוצאת הטלה של קובייה היא מספר שלם בין 1 ל-6). רשימת התוצאות עלולה לכלול מספרים שגויים (כלומר שאינם בין 1 ל-6). המשתנה `valid` הוא מטיפוס בוליאני ושאר המשתנים הם מטיפוס שלם.

```
i = 1;
s = 0;
valid = true;
while(valid && (i <= length))
{
    Console.Write("Enter the number on the die: ");
    die = int.Parse(Console.ReadLine());
```

```
    if((die >= 1) && (die <= 6))
        s = s + die;
    else
        valid = false;
    i = i + 1;
} // while
if (valid)
    Console.WriteLine(s);
else
    Console.WriteLine(die);
```

א. תארו את הפלט עבור הקלט: 3 2 1 3.
ב. תארו את הפלט עבור הקלט: 3 0 1 3.
ג. מהו תפקיד המשתנה הבוליאני valid?
ד. מהי מטרת קטע התוכנית?

שאלה 7.57

פתחו אלגוריתם שהקלט שלו הוא רשימה של 20 מספרים חיוביים. הקלט נחשב חוקי אם **כל** המספרים בו הם זוגיים. האלגוריתם בודק אם הקלט חוקי. אם הקלט חוקי האלגוריתם מציג כפלט את סכום המספרים שבקלט, ואחרת תוצג הודעה כי הקלט איננו חוקי. ישמו את האלגוריתם בשפת C#.

הדרכה: השתמשו במשתנה בוליאני שיאותחל ב-**true**. אם יימצא בקלט מספר אי-זוגי יושם במשתנה זה הערך **false**.

> הבעיה המוצגת בשאלה 7.57 מתאימה לתבנית ***האם כל הערכים בסדרה מקיימים תנאי?*** להעמקה בתבנית פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 7.58

פתחו אלגוריתם שהקלט שלו הוא סיסמה, המורכבת מרשימת אותיות אנגליות המסתיימת ב-'*'. הפלט הוא הודעה המציינת אם הסיסמה תקינה. סיסמה תקינה כוללת לפחות 6 תווים, ובהם לפחות אות אחת קטנה (בתחום a..z) ולפחות אות אחת גדולה (בתחום A..Z). ישמו את האלגוריתם בשפת C#.

שאלה 7.59

בקטע התוכנית הבא length מכיל מספר שלם חיובי. הקלט לקטע התוכנית הוא סדרה באורך length של מילים בנות שלוש אותיות כל אחת.

מטרת קטע התוכנית היא להציג כל מילה בסדרה בסדר אותיות הפוך כל עוד אותיות המילה שונות זו מזו. ברגע שנקלטת מילה שבה לפחות שתי אותיות זהות יש לעצור את מהלך הביצוע.

המשתנה diff הוא מטיפוס **בוליאני**, המשתנים length ו-i מטיפוס שלם והמשתנים let1, let2 ו-let3 הם מטיפוס תווי.

```
diff = __________
i = 1;
while(_________________)
{
    Console.Write("Enter the first letter of the word: ");
    let1 = char.Parse(Console.ReadLine());
    Console.Write("Enter the second letter of the word: ");
    let2 = char.Parse(Console.ReadLine());
```

```
    Console.Write("Enter the third letter of the word: ");
    let3 = char.Parse(Console.ReadLine());
    if (_________________________________)
        Console.WriteLine("{0} {1} {2}", let3, let2, let1);
    else
        diff = ________
} // while
```

השלימו את קטע התוכנית.

7.6 הקשר הלוגי לא (not)

הביטויים הבוליאניים שהכרנו עד כה הם ביטויים פשוטים וביטויים מורכבים. הביטויים הבוליאניים הפשוטים שהכרנו כוללים קבועים (**true** או **false**), משתנים בוליאניים, או ביטויי השוואת ערכים בעזרת פעולות השוואה. ביטויים בוליאניים-מורכבים מתקבלים על ידי קישור של ביטויים פשוטים באמצעות קשרים לוגיים. עד כה הכרנו שני קְשָׁרים לוגיים: או (or) ו-וגם (and).

נכיר כעת קשָר לוגי נוסף, הוא הקשר לא (not). כשמו כן הוא – הוא מבטא שלילה, היפוך התנאי. בניגוד לקשרים שהכרנו עד כה, הקשר לא מופעל על ביטוי בוליאני אחד, ולא על שניים.

הנה **טבלת האמת של הקשר לא**, כאשר b הוא ביטוי בוליאני:

b	לא b
false	**true**
true	**false**

סדר העדיפות של הקשר לא גבוה מסדר עדיפותם של הקשרים וגם ו-או. כלומר הוא קודם להם בחישוב ביטוי בוליאני.

בשפת C# הקשר לא נכתב באמצעות סימן קריאה (!).

דוגמאות

נניח ש-a הוא משתנה מטיפוס שלם ו-b הוא משתנה מטיפוס בוליאני

- ערכו של הביטוי הבוליאני `!true` הוא `false`.
- ערכו של הביטוי הבוליאני `!(a == 1)` הוא `false` כאשר ערכו של a שווה ל-1, ו-`true` אחרת.
- ערכו של הביטוי הבוליאני `!b` הוא `false` כאשר ערכו של b הוא `true`. כאשר ערכו של b הוא `false`, ערכו של הביטוי `!b` הוא `true`.

שאלה 7.60

נניח שהמשתנים a ו-b הם מטיפוס שלם. עבור כל ביטוי מן הביטויים הבוליאניים הבאים, תנו דוגמה לערכים למשתנים שעבורם יהיה ערך הביטוי **true**, ודוגמה לערכי המשתנים שעבורם יהיה ערך הביטוי **false**.

	ערך הביטוי **true**	ערך הביטוי **false**
א. `!(a != b)`	`a =       b =`	`a =      b =`
ב. `!(Math.Abs(a) == a)`	`a =`	`a =`

שאלה 7.61

נניח שהמשתנים `a` ו-`b` הם מטיפוס שלם והמשתנה `c` הוא מטיפוס בוליאני. עבור כל ביטוי מן הביטויים שבמשפטי ההשמה הבאים, תנו דוגמה לערכי המשתנים שעבורם יושם ב-`c` הערך **`true`**, ודוגמה לערכי המשתנים שעבורם יושם ב-`c` הערך **`false`**.

	ערך c הוא **true**	ערך c הוא **false**
א. `c =(x == Math.Sqrt(x))`	`x =`	`x =`
ב. `c =(c && (a == b))`	`a =     b =     c =`	`a =     b =     c =`
ג. `c =(!((a + b)>= 5))`	`a =     b =`	`a =     b =`

שאלה 7.62

במשחק הניחושים מגריל המחשב מספר בתחום 1-100, והשחקן צריך לנחש אותו. בתום המשחק יודיע המחשב לשחקן כמה ניסיונות ניסה עד שהצליח לנחש.

לפניכם קטע תוכנית ב-C#. השלימו את הקטע כך שיבצע את הנדרש:

```
Random rnd = new Random();
int num = ___________________________;      //המספר שהמחשב יגריל
int numOfGuesses = _____________;           //מונה לספירת מספר ניחושים
bool found = ____________;
while (!found)
{
    numOfGuesses++;
    Console.Write("Enter your guess: ");
    guess = int.Parse(Console.ReadLine());
    if (_____________)
    {
        Console.WriteLine("Correct!! ");
        found = true;
    }
    else if (_______________)
        Console.WriteLine("Your guess is too big");
    else
        Console.WriteLine("Your guess is too small");
} // while
Console.WriteLine("It took you {0} guesses", ________________ );
```

שאלה 7.63

הבעיה עוסקת בהצפנת הודעות. הודעה היא סדרת מילים, כך שכל מילה היא רצף אותיות מן הא״ב האנגלי ובין כל שתי מילים מופיע סימן קריאה (!). הצפנת הודעה מתבצעת באופן הבא: בכל מילה אי-זוגית (כלומר המילה הראשונה, השלישית, החמישית וכו׳) מוחלפת כל אות באות **העוקבת** לה בא״ב. בכל מילה זוגית (כלומר המילה השנייה, הרביעית, השישית וכו׳) מוחלפת כל אות באות ה**קודמת** לה בא״ב. סימני הקריאה נותרים במקומם ללא שינוי.

כלומר תחילה יש להצפין כל אות נתונה לאות העוקבת לה בא״ב ולאחר כל קריאה של סימן קריאה יש להפוך את ״כיוון ההצפנה״ (מעוקבת בא״ב לקודמת, או מקודמת בא״ב לעוקבת).

למשל, הצפנת ההודעה DING!DING!DONG! תהיה : EJOH!CHMF!EPOH.
לשם הפשטות נניח שהאותיות A ו-Z אינן נכללות בהודעה.

פתחו וישמו אלגוריתם אשר הקלט שלו הוא מספר המציין אורך של הודעה (כלומר מספר התווים בהודעה) ואחריו ההודעה עצמה (הנקלטת תו אחר תו). הפלט שלו הוא ההודעה כשהיא מוצפנת באופן שתואר לעיל – הצפנת אות תופיע בפלט מיד לאחר קליטתה.

הדרכה: השתמשו באלגוריתם במשתנה מטיפוס בוליאני לשמירת "כיוון ההצפנה". אתחלו את ערכו של משתנה זה ל-**true**, ולאחר כל סימן קריאה הפכו את ערכו באמצעות הקשר *לא*.

הקשר הבוליאני *לא* משמש בתבנית ***האם קיים ערך בסדרה המקיים תנאי?*** להעמקה בתבנית זו פנו לסעיף התבניות המופיע בסוף הפרק.

7.7 קינון הוראות לביצוע-חוזר

בהוראה לביצוע-חוזר, גוף הלולאה יכול להכיל הוראות שונות. ייתכן כי בין הוראות אלו יש גם הוראות לביצוע-חוזר. בכך מתקבל מבנה מקונן, הכולל הוראה לביצוע-חוזר בתוך הוראה לביצוע-חוזר. הדבר דומה לקינון של הוראות לביצוע-בתנאי, שהכרנו בפרק 5. בסעיף זה נראה כי מבנה מקונן של הוראות לביצוע-חוזר הוא שימושי מאוד בפתרון בעיות מסוימות.

בעיה 12

מטרת הבעיה ופתרונה: הצגת הוראה מקוננת לביצוע-חוזר.

פתחו וישמו אלגוריתם אשר הפלט שלו הוא לוח הכפל, מוצג באופן הבא :

```
1  2  3 .........  10
2  4  6 .........  20
3  6  9 .........  30
.        .          .
.        .          .
10  20 ........ 100
```

בבעיה הנתונה אין קלט. מוגדר פלט בלבד. הפלט הוא טבלת המספרים של לוח הכפל. טבלה זו מכילה 100 מספרים, המסודרים בעשר שורות ובעשרה טורים.

פירוק הבעיה לתת-משימות

דרך אחת להצגת לוח הכפל היא על ידי אלגוריתם ובו 10 לולאות, אשר כל אחת מהן תציג כפלט שורה של 10 מספרים.
כלומר, נקבל את הפירוק הבא לתת-משימות :

1. הצגה כפלט בשורה אחת של המספרים 1 עד 10
2. הצגה כפלט בשורה אחת של המספרים 1 עד 10, מוכפלים ב-2
3. הצגה כפלט בשורה אחת של המספרים 1 עד 10, מוכפלים ב-3

.
.

10. הצגה כפלט בשורה אחת של המספרים 1 עד 10, מוכפלים ב-10

זהו ניסוח מסורבל וארוך. בנוסף ניסוח זה אינו כללי: כדי להציג טבלה בעריכה שונה או טבלה חלקית (למשל רק את חמש השורות הראשונות) יש לכתוב אלגוריתם שונה. בנוסף קשה להכליל את האלגוריתם הזה לבעיה מעט שונה, שגודל הטבלה בה (מספר השורות ומספר העמודות) מתקבל כקלט ואינו ידוע בעת כתיבת האלגוריתם.

האם ניתן לכתוב אלגוריתם פשוט, קצר יותר, ובצורת ניסוח כללית יותר?
כן!
בכל אחת מהלולאות המפורטות מתואר ביצוע-חוזר של תת-משימה. אך בעצם ניתן להתייחס לכל לולאה כאל תת-משימה כוללת יותר, אשר גם על ביצועה יש לחזור 10 פעמים.
כלומר, ניתן לתאר זאת בתת-המשימה הבאה שאותה יש לבצע 10 פעמים:

הצגת השורה הבאה בתור בטבלה

כאשר הצגה של שורה בטבלה אף היא מבוטאת על ידי תת-משימה שיש לבצעה 10 פעמים:

הצגת האיבר הבא בתור בשורה

בחירת משתנים

מאחר שיש לנו צורך בשתי הוראות לביצוע-חוזר (האחת בתוך האחרת) ואורכן ידוע מראש, נזדקק לשני משתני הבקרה מטיפוס שלם: **i** ו-**j**.

האלגוריתם

1. עבור כל i שלם מ-1 עד 10 בצע:
1.1. עבור כל j שלם מ-1 עד 10 בצע:
1.1.1. הצג כפלט את ערך הביטוי i*j
1.2. עבור לשורה הבאה

יישום האלגוריתם

```
/*  10*10 התוכנית מציגה כפלט את לוח הכפל של  */
using System;
public class MultTable
{
    public static void Main ()
    {
        const int TABLE_DIMENSION = 10;
1.      Console.WriteLine("Mult Table");
2.      for (int i = 1; i <= TABLE_DIMENSION; i++)
        {
2.1.        for (int j = 1; j <= TABLE_DIMENSION; j++)
            {
2.1.1.          Console.Write("{0,3} ", i * j);
            } // for j
2.2.        Console.WriteLine();// מעבר לשורת הפלט הבאה
        } // for i
    } // Main
} // MultTable
```

הוספת מספר כלשהו לסמן בהוראת פלט, כגון {0,3} משמעו הקצאת 3 מקומות בדיוק על המסך לצורך כתיבת הפלט

שאלה 7.64

בנו טבלת מעקב אחר מהלך ביצוע התוכנית MultTable.

סוף פתרון בעיה 12

שימו ♥ : הפלט מודפס באופן לא מושלם כיוון שמספרים שונים עלולים לתפוס מקום שונה על המסך. למשל, מספרים חד-ספרתיים תופס פחות מקום מאשר מספר דו-ספרתי. כדי ליצור טבלה מסודרת, שעמודותיה ממוקמות באופן אחיד, ניתן לשנות את הוראת הפלט, כך שלכל מספר יוקצו בדיוק 3 מקומות. לשם כך נוסיף לסמן שבהוראת הפלט שבשורה 2.1.1 את הערך 3 :

```
Console.Write("{0,3} ", i * j);
```

במבנה מקונן של לולאות נכללת **לולאה פנימית** בתוך גוף של **לולאה חיצונית**.
הלולאה הפנימית מתבצעת כולה בכל סיבוב של גוף הלולאה החיצונית.

בפתרון בעיה 12 הביצוע-החוזר של הלולאה החיצונית כלל 10 ביצועים-חוזרים של גוף הלולאה הפנימית. כיוון שהלולאה החיצונית מתבצעת 10 פעמים, הרי מספר הפעמים הכולל שיתבצע גוף הלולאה הפנימית הוא 100 (10*10=100).

שימו ♥ : קינון אינו מוגבל רק להוראות לביצוע-חוזר שמספר החזרות בהן ידוע מראש. ההוראה החיצונית יכולה להיות גם הוראה לביצוע-חוזר שתלויה בתנאי, וכך גם ההוראה הפנימית.

בכתיבת מבנה מקונן של לולאות נקפיד על **הזחה**. כיוון שהלולאה הפנימית היא חלק מגוף הלולאה החיצונית, נזיח את הלולאה הפנימית ביחד עם ההוראות של גוף הלולאה החיצונית. הדבר דומה להזחה שבמבנה המקונן של ביצוע-בתנאי.

שאלה 7.65

נניח שבאלגוריתם דומה לאלגוריתם המתואר בבעיה 12, משתנה הבקרה של הלולאה החיצונית i יתחיל מהערך 2, ויגדל בכל ביצוע של הלולאה ב-1, עד הגיעו לערך 6. מה יהיה הפלט של אלגוריתם זה? ומה יהיה מספר הפעמים הכולל שיתבצע גוף הלולאה הפנימית שבו?

שאלה 7.66

תארו עבור כל אחד מקטעי התוכניות הבאים את פלט קטע התוכנית ואת מספר הפעמים הכולל שמתבצע גוף הלולאה הפנימית:

א.

```
for (int i = 1; i <= 5; i++)
{
    for (int j = 1; j <= 5; j++)
        Console.Write("*");
    Console.WriteLine();
}
```

ב.

```
for (int i = 1; i <= 5; i++)
{
    for (int j = 1; j <= i; j++)
        Console.Write("*");
    Console.WriteLine();
}
```

ג.

```
for (int i = 1; i <= 5; i++)
{
    for (int j = 1; j <= 5; j++)
        Console.Write(j);
    Console.WriteLine();
}
```

ד.

```
for (int i = 1; i <= 5; i++)
{
    for (int j = 5; j >= i; j--)
        Console.Write("*");
    Console.WriteLine();
}
```

שאלה 7.67

עבור כל אחד מהסעיפים הבאים פתחו וישמו אלגוריתם אשר הפלט שלו הוא כפי שמופיע בתיאור הסעיף. בכל אחד מהאלגוריתמים מותר לכתוב **רק הוראה אחת** המדפיסה תו או ספרה **והוראה אחת** למעבר שורה בפלט (מכאן נובע כי עליכם להשתמש בלולאות מקוננות):

א.
```
*
**
***
****
*****
```
ב.
```
*****
****
***
**
*
```
ג.
```
*****
****
***
**
*
**
***
****
*****
```
ד.
```
1
12
123
1234
12345
```
ה.
```
1
22
333
4444
55555
```
ו.
```
12345
1234
123
12
1
```
ז.
```
55555
4444
333
22
1
```
ח.
```
11111
2222
333
44
5
```

שאלה 7.68

בכיתה 40 תלמידים, כל תלמיד לומד 20 מקצועות. לפניכם תוכנית המחשבת עבור כל תלמיד את ממוצע ציוניו:

```
/*
קלט: 20 הציונים עבור כל אחד מ-40 תלמידי הכיתה
פלט: הממוצע של כל תלמיד ותלמיד
*/
using System;
public class Grades
{
    public static void Main ()
    {
        const int STUDENT_NUM = 40;
        const int GRADE_NUM = 20;
        double grade;
        double sum;
        double average;
        for(int i = 1; i <= STUDENT_NUM; i++)
        {
            for(int j = 1; j <= GRADE_NUM; j++)
            {
              Console.WriteLine("Enter next student grade: ");
                grade = int.Parse(Console.ReadLine());
                sum = sum + grade;
            } // for j
        } // for i
    } // Main
} // Grades
```

המשפטים הבאים חסרים בתוכנית הנתונה, היכן צריך לשלב את המשפטים האלה כדי שהתוכנית תבצע את הנדרש?

א. `sum = 0;`

ב. `Console.WriteLine(sum / 20);`

שאלה 7.69

פתחו אלגוריתם שמקבל כקלט מספר חיובי שלם num, ולאחריו num סדרות באורך לא ידוע של מספרים חיוביים שלמים, כאשר כל סדרה מסתיימת ב-1-. האלגוריתם יציג כפלט את אורכה של הסדרה הארוכה ביותר. ישמו את האלגוריתם בשפת C#.

שאלה 7.70

במפעל לייצור נעליים יש עובדים רבים. פתחו אלגוריתם שיקבל כקלט את מספר העובדים במפעל, ואחר כך רשימה של כל המשכורות של השנה האחרונה (12 משכורות) **לכל אחד** מעובדי המפעל. האלגוריתם יחשב וידפיס את **המשכורת האחרונה** של העובד המסכן שסכום משכורותיו כל השנה היה הנמוך ביותר. ישמו את האלגוריתם בשפת C#.

בעזרת החומר הנלמד בפרק 7 ניתן לפתור גם בעיות המשתמשות בתבניות ***מציאת כל הערכים בסדרה המקיימים תנאי*** ו***מעבר על זוגות סמוכים בסדרה***. להעמקה בתבניות אלו פנו לסעיף התבניות המופיע בסוף הפרק.

סיכום

בפרק זה למדנו כיצד להורות על ביצוע-חוזר של תת-משימה. הדבר נעשה באמצעות **הוראה לביצוע-חוזר**, והיא הוראת בקרה הנקראת גם **לולאה**.

הכרנו שני מבנים של הוראה לביצוע-חוזר: **הוראה לביצוע-חוזר מספר פעמים ידוע מראש**, ו**הוראה לביצוע-חוזר-בתנאי**.

הוראה לביצוע-חוזר באורך ידוע מראש נכתבת בצורה:

בצע X פעמים:
הוראות לביצוע

או

עבור כל איבר בתחום ... בצע:
הוראות לביצוע

במבנה הראשון סדרת ההוראות לביצוע מתבצעת X פעמים.

במבנה השני איבריו של התחום המוגדר נסרקים, ועבור כל אחד מהם מבוצעת סדרת ההוראות לביצוע.

הוראה לביצוע-חוזר-בתנאי נכתבת בצורה:

כל עוד <תנאי> בצע:
הוראות לביצוע

התנאי המתואר נקרא **תנאי הכניסה** ללולאה. כל עוד התנאי מתקיים, סדרת ההוראות לביצוע מתבצעת שוב ושוב. כאשר התנאי לא מתקיים, מסתיימת ההוראה לביצוע-החוזר.

בכתיבת הוראה לביצוע-חוזר (מספר פעמים ידוע מראש או בתנאי) אנו מקפידים על **הזחה**: קבוצת ההוראות אשר יש לחזור על ביצוען כתובות כשהן מוזחות פנימה.

קבוצת ההוראות אשר יש לחזור על ביצוען נקראת **גוף הלולאה**.

כאשר מספר הפעמים לביצוע-חוזר אינו ידוע מראש (לפני תחילת ביצוע הלולאה), והוא תלוי בתנאי, נשתמש בהוראה לביצוע-חוזר-בתנאי. בכל מקרה אחר נשתמש בהוראה לביצוע-חוזר באורך ידוע מראש. בכך נסייע ביצירת תוכניות קריאות ובהירות.

בין ההוראות לביצוע-חוזר-בתנאי הבחנו בסוג מסוים: **הוראות לביצוע-חוזר התלויות בזקיף**.

זקיף הוא סימן מיוחד המציין את סוף רשימת איברי הקלט. הזקיף **אינו** נחשב כחלק מהקלט. לכן בהוראה לביצוע-חוזר-בתנאי, יש לוודא שאיבר הקלט התורן אינו הזקיף כדי שלא נעבד אותו כחלק מהקלט. זיהוי הזקיף צריך להביא לסיום הביצוע-החוזר.

הוראה לביצוע-חוזר משמשת, בין השאר לביצוע **פעולות צבירה ומנייה**. צבירה נעשית באמצעות צובר, ומנייה באמצעות מונה.

צובר הוא משתנה שתפקידו לצבור ערכים (למשל נתונים או תוצאות חישוב). צובר יכול לשמש לפעולות צבירה שונות (למשל, סכום מצטבר או מכפלה מצטברת).

מונה הינו משתנה אשר תפקידו למנות אירועים (למשל, מספר המופעים של נתונים). מאחר שהשימוש במונה הוא לצורך ספירה הרי הוא בדרך כלל מטיפוס שלם.

באלגוריתמים שיש בהם שימוש במונה או צובר יש להקפיד עבורם על **אתחול**. **באתחול** יושם במונה או בצובר ערך התחלתי המתאים להגדרת תפקידם.

הוראות לביצוע-חוזר משמשות גם לפתרון בעיות שנדרש למצוא בהן **איבר מקסימלי** או **איבר מינימלי** בתוך קבוצת איברים, או ערך נלווה (כמו **מיקומו) של האיבר המקסימלי** (או המינימלי) בקבוצת איברים סדורה.

כאשר אלגוריתם כולל לולאה ניתן לחשב את **מספר הפעמים שהלולאה תתבצע**. מספר זה תלוי בדרך כלל בקלט לאלגוריתם.

יתכן שעבור קלטים מסוימים תתבצע הלולאה אינסוף פעמים. לולאה כזאת נקראת **לולאה אינסופית**. אנו מחשיבים לולאה אינסופית כלולאה שגויה.

בפרק הבא נדגיש את החשיבות של פתרון בעיות באמצעות אלגוריתמים שבהם מספר הפעמים לביצוע-חוזר הוא קטן ככל האפשר.

משתנה בוליאני הוא משתנה שיכול לקבל אחד משני הערכים **true** ו-**false**. ניתן להשתמש במשתנים בוליאניים כחלק מביטויים בוליאניים.

הקשר הלוגי *לא* (not) משמש ליצירת ביטויים בוליאניים המורכבים מביטויים פשוטים יותר (בדומה לקשרים *וגם* (and) ו-*או* (or)). הוא מופעל על ביטוי בוליאני והוא הופך את ערכו הלוגי (מ-**true** ל-**false**, ולהיפך). עדיפותו של הקשר הלוגי *לא* גבוהה מעדיפותם של הקשרים *וגם* ו-*או*.

הוראה שנמצאת בתוך הוראה לביצוע-חוזר יכולה להיות בעצמה הוראה לביצוע-חוזר. בכך מתקבל **קינון של הוראות לביצוע-חוזר**.

סיכום מרכיבי שפת C# שנלמדו בפרק 7

הוראה לביצוע-חוזר מספר פעמים ידוע מראש מיושמת ב-C# במשפט **for**.

המבנה הכללי של משפט for הוא:

```
for (אתחול משתנה הבקרה; התנאי להמשך הביצוע ; שינוי משתנה הבקרה)
{
    הוראות לביצוע
}
```

כל עוד התנאי להמשך הביצוע (שתלוי בדרך כלל בערכו של משתנה הבקרה) מתקיים, ביצוע הלולאה ממשיך, כלומר מתבצעת סדרת ההוראות לביצוע. סדרת הוראות זו נקראת **גוף הלולאה**.

משתנה הבקרה הוא בדרך כלל מטיפוס שלם. אם אין בו שימוש מעבר לתחום הלולאה ניתן להצהיר עליו בתוך הלולאה, למשל כך:

```
for(int i = 1; i <= 10; i++)
```

הוראה לביצוע-חוזר-בתנאי מיושמת ב-C# במשפט **while**.

המבנה הכללי של משפט while הוא:

```
while (ביטוי בוליאני)
{
    הוראות לביצוע
}
```

ביצוע משפט while מתחיל בחישוב ערכו של הביטוי הבוליאני. אם ערכו **true** מתבצעות ההוראות שבגוף הלולאה. בתום ביצוע גוף הלולאה מחושב ערכו של הביטוי הבוליאני שוב. אם ערכו **true** מתבצעות שוב ההוראות שבגוף הלולאה, וכך הלאה כל עוד ערכו של הביטוי הבוליאני הוא **true**. כאשר ערכו **false** מסתיים ביצוע משפט ה-**while**.

ההצהרה על משתנה בוליאני נעשית בשפת C# באמצעות המילה השמורה **bool**, למשל כך:

```
bool flag;
```

הקשר לא נכתב ב-C# באמצעות הסימן !, למשל כך:

```
!(x == 5)
```

תבניות – פרק 7

פירוט מלא של התבניות ושל שאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

בסעיף זה מוצגות תבניות שונות, חלקן מתייחסות לסדרת קלט שאורכה ידוע מראש וחלקן מתייחסות לסדרת קלט שאורכה אינו ידוע מראש.

מנייה

שם התבנית: **מנייה**

נקודת מוצא: אורך סדרת נתוני הקלט limit, סדרת ערכי הקלט, תנאי מנייה condition

מטרה: מנייה של ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

אלגוריתם:

1. אתחל את count ב-0
2. **בצע** limit **פעמים**:
   2.1. קלוט ערך ב-element
   2.2. **אם** element מקיים את condition
      2.2.1. הגדל את count ב-1

שם התבנית: **מנייה**

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, תנאי מנייה conditionToCount

מטרה: מנייה של ערכי הקלט המקיימים את התנאי conditionToCount. משך ביצוע המנייה תלוי בתנאי conditionToEnd.

אלגוריתם:

1. אתחל את count ב-0
2. קלוט ערך ב-element
3. **כל עוד לא** מתקיים conditionToEnd **בצע**:
   3.1. **אם** element מקיים את conditionToCount
      3.1.1 הגדל את count ב-1
   3.2. קלוט ערך ב-element

צבירת סכום

שם התבנית: **צבירת סכום**

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה condition

מטרה: צבירת הסכום של הערך initial ושל ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

אלגוריתם:

1. אתחל את sum ב-initial
2. **בצע** limit **פעמים**:
   2.1. קלוט ערך ב-element
   2.2. **אם** element מקיים את condition
   2.2.1. הוסף ל-sum את ערכו של element

שם התבנית: **צבירת סכום**

נקודת מוצא: תנאי סיום conditionToEnd, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה conditionToSum

מטרה: צבירת סכום של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToSum. משך ביצוע הצבירה תלוי בתנאי conditionToEnd.

אלגוריתם:

1. אתחל את sum ב-initial
2. קלוט ערך ב-element
3. **כל עוד** לא מתקיים conditionToEnd **בצע**:
   3.1. **אם** element מקיים את conditionToSum
   3.1.1. הוסף ל-sum את ערכו של element
   3.2. קלוט ערך ב-element

צבירת מכפלה

שם התבנית: **צבירת מכפלה**

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה condition

מטרה: צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

אלגוריתם:

1. אתחל את mult ב-initial
2. **בצע** limit **פעמים**:
   2.1. קלוט ערך ב-element
   2.2. **אם** element מקיים את condition
   2.2.1. הכפל את mult ב-element והשם את המכפלה ב-mult

שם התבנית: **צבירת מכפלה**

נקודת מוצא: תנאי סיום conditionToEnd, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה conditionToMult

מטרה: צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToMult. משך ביצוע הצבירה תלוי בתנאי conditionToEnd.

אלגוריתם:

1. אתחל את mult ב-initial
2. קלוט ערך ב-element
3. כל עוד לא מתקיים conditionToEnd **בצע:**
   3.1. אם element מקיים את conditionToMult
      3.1.1. הכפל את mult ב-element והשם את המכפלה ב-mult
   3.2. קלוט ערך ב-element

ממוצע של סדרת מספרים

שם התבנית: **ממוצע של סדרת מספרים**

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, תנאי conditionToInclude

מטרה: חישוב ממוצע של ערכי הקלט המקיימים את התנאי conditionToInclude. ביצוע החישוב תלוי בתנאי conditionToEnd.

אלגוריתם:

1. אתחל את count ב-0
2. אתחל את sum ב-0
3. קלוט ערך ב-element
4. כל עוד לא מתקיים conditionToEnd **בצע:**
   4.1. אם element מקיים את conditionToInclude
      4.1.1. הגדל את count ב-1
      4.1.2. הוסף לערכו של sum את element
   4.2. קלוט ערך ב-element
5. השם ב-average את ערכו של הביטוי החשבוני sum/count

מציאת מקסימום או מינימום בסדרה

בשל הדמיון האלגוריתמי בין מציאת מקסימום למציאת מינימום וכדי לא לחזור בפירוט על התבניות, נפרט את התבניות באופן הבא: עבור התבנית ***מציאת מקסימום*** נתייחס לסדרה שאורכה ידוע מראש ואילו עבור התבנית ***מציאת מינימום*** נתייחס לסדרה שאורכה אינו ידוע מראש.

שם התבנית: **מציאת מקסימום בסדרה**
נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט
מטרה: מציאת הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit
אלגוריתם:

1. קלוט ערך ב-max
2. **בצע** limit-1 **פעמים**:
   2.1. קלוט ערך ב-element
   2.2 **אם** element > max
      2.2.1 השם ב-max את הערך של element

שם התבנית: **מציאת מינימום בסדרה**
נקודת מוצא: תנאי סיום conditionToEndMinSearch, ערכי הקלט
מטרה: מציאת הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי conditionToEndMinSearch
אלגוריתם:

1. קלוט ערך ב-min
2. קלוט ערך ב-element
3. **כל עוד** לא מתקיים conditionToEndMinSearch **בצע**:
   3.1. **אם** element < min
      3.1.1. השם ב-min את הערך של element
   3.2. קלוט ערך ב-element

מציאת ערך נלווה למקסימום או למינימום בסדרה

גם כאן בשל הדמיון האלגוריתמי בין התבניות, נפרט את התבניות באופן הבא: עבור התבנית **מציאת ערך נלווה למקסימום בסדרה** נתייחס לסדרה שאורכה ידוע מראש ואילו עבור התבנית **מציאת ערך נלווה למינימום בסדרה** נתייחס לסדרה שאורכה אינו ידוע מראש. בתבניות הבאות נמצא מיקום של איבר כערך נלווה.

שם התבנית: **מציאת ערך נלווה למקסימום בסדרה**

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט

מטרה: מציאת מיקום הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit

אלגוריתם:

1. קלוט ערך ב-max
2. אתחל את placeOfMax ב-1
3. בצע limit-1 פעמים:
   3.1. קלוט ערך ב-element
   3.2. אם element > max
      3.2.1. השם ב-max את הערך של element
      3.2.2. השם ב-placeOfMax את מיקומו של element בקלט

שם התבנית: **מציאת ערך נלווה למינימום בסדרה**

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט

מטרה: מציאת מיקום הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי conditionTtoEnd

אלגוריתם:

1. קלוט ערך ב-min
2. אתחל את placeOfMin ב-1
3. קלוט ערך ב-element
4. אתחל את currentPlace ב-2
5. כל עוד לא מתקיים conditionToEnd בצע:
   5.1. אם element < min
      5.1.1. השם ב-min את הערך של element
      5.1.2. השם ב-placeOfMin את הערך של currentPlace
   5.2. הגדל את currentPlace ב-1
   5.3. קלוט ערך ב-element

איסוף בקיזוז

שם התבנית: **איסוף בקיזוז באמצעות מנייה**

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי לקיזוז conditionToDecrease

מטרה: איסוף בקיזוז באמצעות מנייה של ערכי הקלט מתוך סדרה שאורכה limit. הקיזוז מתבצע על פי התנאי conditionToDecrease

אלגוריתם:

1. *אתחל את* count *ב-0*
2. ***בצע*** limit ***פעמים:***
   2.1. *קלוט ערך ב-*element
   2.2. *אם* element *לא מקיים את* conditionToDecrease
      2.2.1. *הגדל את* count *ב-1*
   2.3. *אחרת*
      2.3.1. *הקטן את* count *ב-1*

שם התבנית: **איסוף בקיזוז באמצעות צבירה**

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, ערך צבירה התחלתי initial, תנאי לקיזוז conditionToDecrease

מטרה: איסוף בקיזוז באמצעות צבירת סכום של ערכי הקלט ושל הערך initial. משך הביצוע תלוי בתנאי conditionToEnd, והקיזוז מתבצע על פי התנאי conditionToDecrease

אלגוריתם:

1. *אתחל את* sum *ב-*initial
2. *קלוט ערך ב-*element
3. ***כל עוד לא מתקיים*** conditionToEnd ***בצע:***
   3.1. *אם* element *לא מקיים את* conditionToDecrease
      3.1.1. *הוסף ל-*sum *את ערכו של* element
   3.2. *אחרת*
      3.2.1. *הפחת מ-*sum *את ערכו של* element
   3.3. *קלוט ערך ב-*element

פירוק מספר חיובי לספרותיו

שם התבנית: **פירוק מספר חיובי לספרותיו**

נקודת מוצא: מספר שלם חיובי num

מטרה: הצגה כפלט של ספרותיו של num

אלגוריתם:

1. *כל עוד* num *שונה מ-0 בצע*
2. *הצג כפלט את ספרת האחדות של* num
3. *הקטן את* num *פי 10*

בניית מספר

שם התבנית: **בניית מספר**
נקודת מוצא: אורך סדרת נתוני הקלט limit, ספרות הקלט
מטרה: בניית מספר מספרות הקלט
אלגוריתם:

1. אתחל את num ב-0
2. בצע limit פעמים:
   2.1. קלוט ספרה ב-digit
   2.2. השם ב-num את הערך של הביטוי החשבוני num * 10 + digit

האם כל הערכים בסדרה מקיימים תנאי?

שם התבנית: **האם כל הערכים בסדרה מקיימים תנאי?**
נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי condition
מטרה: קביעת הערך true אם כל הערכים בסדרה מקיימים את התנאי condition וקביעת הערך false אם קיים ערך אחד בסדרה שאינו מקיים את התנאי
אלגוריתם:

1. אתחל את all ב-true
2. אתחל את howmany ב-1
3. כל עוד howmany ≤ limit וגם ערכו של all הוא true בצע:
   3.1. קלוט ערך ב-element
   3.2. הגדל את ערכו של howMany ב-1
   3.3. אם element אינו מקיים את condition
      3.3.1. השם ב-all את הערך false

האם קיים ערך בסדרה המקיים תנאי?

שם התבנית: **האם קיים ערך בסדרה המקיים תנאי?**
נקודת מוצא: תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition
מטרה: קביעת הערך true אם קיים ערך בסדרה המקיים את התנאי condition וקביעת הערך false אם כל הערכים בסדרה אינם מקיימים את התנאי
אלגוריתם:

1. אתחל את found ל-false
2. קלוט ערך ב-element
3. כל עוד התנאי toEnd לא מתקיים וגם ערכו של found הוא false בצע:
   3.1. אם element מקיים את condition
      3.1.1. השם ב-found את הערך true
   3.2. אחרת
      3.2.1. קלוט ערך ב-element

מציאת כל הערכים בסדרה המקיימים תנאי

שם התבנית: **מציאת כל הערכים בסדרה המקיימים תנאי**
נקודת מוצא: תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition
מטרה: הצגה כפלט של כל ערכי הקלט המקיימים את התנאי condition
אלגוריתם:

*1. קלוט ערך ב-*element
2. כל עוד התנאי toEnd *לא מתקיים בצע:*
3.1 אם element *מקיים את* condition
3.1.1 הצג כפלט את ערכו של element
*3.2 קלוט ערך ב-*element

מעבר על זוגות סמוכים בסדרה

שם התבנית: **מעבר על זוגות סמוכים בסדרה**
נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט
מטרה: הצגה כפלט של סכומי כל זוגות הערכים הסמוכים בסדרת הקלט שאורכה limit
אלגוריתם:

*1. קלוט ערך ב-*beforeLast
2. בצע limit-1 *פעמים*
*2.1 קלוט ערך ב-*last
2.2 הצג כפלט את הערך של הביטוי החשבוני beforeLast + last
*2.3 השם ב-*beforeLast *את הערך של* last

פרק 8 – יעילות של אלגוריתמים

אלגוריתמים נבחנים על פי מספר קני מידה. קנה המידה החשוב ביותר הוא נכונות. כלומר השגת המטרה (מתן הפלט הנכון) עבור כל קלט חוקי. קנה מידה נוסף הוא יעילות.

יעילות של אלגוריתם (תוכנית) נמדדת על פי "משאבי המחשב" הדרושים לביצוע האלגוריתם. משאבים אלה הם **גודל המקום** (בזיכרון) ו**הזמן** הדרוש לביצוע האלגוריתם.

גודל המקום נמדד בעיקר על פי מספר המשתנים של האלגוריתם.

זמן-הביצוע נקבע על פי מספר פעולות היסוד שיתבצעו במהלך הרצת האלגוריתם.

פעולות היסוד הן: פעולת קלט, פעולת פלט ופעולות חישוב.

למעשה, בצורה פשטנית נאמר שכל הוראה באלגוריתם כוללת פעולת יסוד אחת, ומכאן **מדידת זמן-הביצוע של אלגוריתם** נעשית על פי מספר ההוראות שיתבצעו במהלך הרצת האלגוריתם במחשב.

שימו ♥: המדד הוא מספר ההוראות **שיתבצעו** ולא מספר ההוראות באלגוריתם! מספר ההוראות באלגוריתם אינו מעיד בהכרח על מספר ההוראות שיתבצעו במהלך הרצתו. בפרט, עבור אלגוריתם הכולל לולאה, ייתכן שמספר ההוראות שבו הוא מועט, אך מספר ההוראות שיתבצעו הוא רב, כי כל הוראה בלולאה יכולה להתבצע כמה פעמים. נפרט נקודה זו במהלך הפרק.

מדידת זמן-הביצוע **לא** נעשית על פי הזמן בפועל שאורכת ריצת תוכנית המיישמת את האלגוריתם. יש לכך כמה סיבות:

1. יש מחשבים מהירים יותר ומהירים פחות. מכיוון שפעולות היסוד נמשכות זמן שונה ממחשב למחשב, זמן ההרצה בפועל של אותה תוכנית יכול להיות שונה ממחשב למחשב.
2. גם אם נריץ את התוכנית באותו מחשב כמה פעמים ייתכן שבכל פעם ההרצה תימשך פרק זמן שונה, כתלות בעומס המחשב באותו הזמן. ניתן לדמות זאת לזמן המתנה במסעדה: זמן ההמתנה קצר יותר כאשר במסעדה פחות אורחים. בדומה, זמן ההמתנה לסיום ריצת תוכנית קצר יותר כאשר יש מעט תוכניות נוספות אשר רצות במקביל במחשב.
3. גם המהדר (הקומפיילר) שהשתמשנו בו כדי להדר את התוכנית משפיע על מהירותה של תוכנית. מהדרים שונים יוצרים תוכניות שונות (בשפת מכונה) ולכל אחת מהן זמן ריצה שונה במקצת, גם אם הן מורצות באותו המחשב ובאותה השעה.

בפרק זה נתמקד במדד זמן-הביצוע של אלגוריתם ונכיר אלגוריתמים שונים לפתרון אותה הבעיה. האלגוריתמים ייבדלו זה מזה בזמן-הביצוע, לפעמים בצורה משמעותית. למדד המקום נתייחס בפרק 10.

נמחיש את ההבדל בין זמני הביצוע של שני אלגוריתמים שונים באמצעות בעיה שאנו נתקלים בה מדי פעם בחיי יום-יום, והיא גם אחת מבעיות היסוד במדעי המחשב. הבעיה היא בעיית חיפוש ברשימה ממוינת:

קלט: רשימה של שמות ממוינים לפי סדר הא"ב (למשל מדריך טלפונים או דף קשר).

פלט: הודעה מתאימה אם שם מסוים מופיע ברשימה.

ניתן לחפש את השם המבוקש בשתי דרכים:

הדרך הראשונה, היא בחיפוש **סדרתי**, כלומר מעבר על פני הרשימה, שם אחרי שם, עד שנמצא את השם המבוקש או עד שיתברר שהשם לא מופיע ברשימה. ברשימות ארוכות מאוד, למשל כמו ספר טלפונים, חיפוש כזה עלול לקחת הרבה מאוד זמן.

אם תיזכרו כיצד אתם מחפשים בספר טלפונים תיווכחו שלחיפושים ברשימות ארוכות אנו משתמשים בדרך כלל בחיפושים לא סדרתיים, כמו החיפוש **הלא-סדרתי** הבא: נתבונן בשם המופיע באמצע הרשימה, ונשווה אותו לשם המבוקש, אם השמות זהים – סיימנו את תהליך החיפוש. אם לא, נבדוק לאן עלינו להמשיך את החיפוש: אחרי השם המבוקש או לפניו (כלומר בחֲציה השני של הרשימה או בחציה הראשון). אם השם המבוקש מופיע בסדר הא״ב אחרי השם האמצעי נמשיך את תהליך החיפוש רק בחצי השני (כיוון שהרשימה ממוינת, לא ייתכן שיהיה בחצי הראשון), ואם השם המבוקש מופיע בסדר הא״ב לפני השם האמצעי נמשיך את התהליך רק בחצי הראשון. תהליך החיפוש ימשיך בדיוק באותו אופן: בדיקת השם האמצעי והשוואתו לשם המבוקש, ובהתאם לתוצאת ההשוואה המשך חיפוש במחצית התחום המתאימה, וכך הלאה עד מציאת השם או עד שהסתיים התהליך (והשם לא נמצא). חיפוש זה נקרא חיפוש בינָרי, כיוון שהוא מבוסס על הקטנת תחום החיפוש לחצי מגודלו אחרי כל השוואה של השם המבוקש לשם נבחר (האמצעי בתחום החיפוש).

נדגים זאת:

לפנינו רשימה ממוינת של מספרים:

36, 35, 31, 27, 25, 24, 20, 19, 15, 13, 12, 9, 7, 6, 5, 3, 1

וברצוננו לחפש בתוכה את המספר 27.

בשיטת **החיפוש הסדרתי**, דרושות לנו 14 פעולות השוואה עד שנמצא את המספר המבוקש (כי 27 נמצא במקום ה-14 ברשימה).

נבדוק מה קורה בשיטת **החיפוש הבינרי**:

- נתבונן במספר האמצעי ברשימה, 15. כיוון שהמספר המבוקש **גדול** ממנו, נמשיך את החיפוש מימינו.
- כעת אנו מתמקדים ברשימת המספרים: 36, 35, 31, 27, 25, 24, 20, 19.
- נתבונן במספר האמצעי ברשימה זו, 25 (למעשה, אורך הרשימה זוגי ולכן שני מספרים נמצאים באמצע הרשימה; בחרנו שרירותית את הקטן מביניהם). כיוון שהמספר המבוקש **גדול** ממנו, נמשיך את החיפוש מימינו.
- כעת אנו מתמקדים ברשימת המספרים: 36, 35, 31, 27.
- נתבונן במספר האמצעי ברשימה זו, 31 (גם הפעם, זהו המספר הקטן מבין שני המספרים שבאמצע הרשימה). כיוון שהמספר המבוקש **קטן** ממנו, נמשיך את החיפוש משמאלו.
- כעת אנו מתמקדים ברשימת המספרים: 27.

  זו רשימה בת מספר אחד, והוא שווה בדיוק למספר המבוקש, ולכן סיימנו את תהליך החיפוש.

בסך הכול ביצענו ארבע השוואות (למספרים 15, 25, 31 ו-27).
בהשוואה לחיפוש הסדרתי ההבדל הוא משמעותי!

כאשר רשימת השֵמות ארוכה, יש חשיבות רבה לבחירת הדרך שתאפשר את ביצוע משימת החיפוש בזמן קצר עד כמה שניתן. זמן החיפוש נקבע בעיקר על פי מספר ההשוואות המתבצעות במהלך החיפוש (השוואות של השם המבוקש לשם נבחר). לכן נעדיף לבחור בדרך אשר בה יתבצעו פחות השוואות. עבור רשימה בת 1000 שמות, מספר ההשוואות בחיפוש הסדרתי עלול להיות

קרוב ל-1000, כיוון שייתכן שהשם המבוקש נמצא בקצה הרשימה. לעומת זאת, בחיפוש בינרי באותה רשימה מספר ההשוואות לא יעלה בכל מקרה על 10 השוואות. לכן עבור בעיית החיפוש הנתונה, חיפוש בינרי יעיל הרבה יותר. בפרקים הבאים נכיר חיפוש בינרי ביתר פירוט.

יש אנשים הסוברים שמחשבים הינם כה מהירים עד שאין משמעות למושג "זמן-ביצוע של אלגוריתם (או של תוכנית)" ולהשוואה בין זמני הביצוע של אלגוריתמים שונים. לדעה זו אין כל בסיס. קיימות בעיות אשר עבורן זמן הריצה של תוכנית יכול להיות דקות רבות, שעות ואף ימים (למשל תוכניות לחיזוי מזג אויר). עבור בעיית החיפוש שהצגנו, זמן הריצה של תוכנית המבצעת חיפוש סדרתי ברשימה בת 10,000,000 שמות עלול להיות מספר דקות. לעומת זאת, זמן הריצה של תוכנית המבצעת חיפוש בינרי באותה רשימה ובאותו המחשב לא יעלה על מספר שניות.

כזכור, זמן-הביצוע של אלגוריתם נמדד על פי מספר ההוראות שיתבצעו במהלך ביצוע האלגוריתם. מספר זה תלוי בדרך כלל בקלט של האלגוריתם.

למשל, בבעיית החיפוש הקלט לאלגוריתם הוא רשימת השמות הממוינת והשם שיש לחפש. מספר ההוראות שיתבצעו תלוי באורך רשימת השמות: ככל שהרשימה תהיה ארוכה יותר ייתכנו יותר השוואות, כלומר יתבצעו יותר פעולות השוואה.

בפתרון הבעיה הבאה נראה דוגמה לאלגוריתם אשר זמן-הביצוע שלו תלוי בערכו של הקלט.

בעיה 1

מטרת הבעיה ופתרונה: הצגת שלושה אלגוריתמים שונים לפתרון בעיה, הנבדלים זה מזה במידת יעילותם מבחינת זמן-ביצוע. באלגוריתמים היעילים יותר נעשה ניצול טוב של מאפייני קלט-פלט.

> פתחו אלגוריתם אשר הקלט שלו הוא מספר שלם גדול מ-1, והפלט שלו הוא המחלקים של המספר הנתון. המחלקים של מספר שלם חיובי הם כל המספרים השלמים החיוביים המחלקים את המספר ללא שארית. לכן למשל עבור הקלט 6 הפלט הדרוש הוא: 6 3 2 1.
>
> ישמו את האלגוריתם בשפת C#. תארו את זמן-ביצוע האלגוריתם על ידי הערכת מספר ההוראות שיתבצעו במהלך ביצוע האלגוריתם.

בדיקת דוגמאות קלט

שאלה 8.1

ציינו מהו הפלט המתאים עבור כל אחד מהקלטים הבאים:

א. 12

ב. 29

ג. 30

תחום המספרים השלמים שמתוכו יש להציג מחלקים הוא התחום שבין 1 ובין המספר הנתון כקלט. כלומר, יש להציג כל מספר שהוא גדול או שווה ל-1 וקטן או שווה למספר הנתון ומחלק אותו ללא שארית.

? כיצד אפשר לחשב את המספרים הדרושים לפלט?

הנה רעיון ראשון: נסרוק את תחום המספרים בין 1 לבין המספר הנתון, ונבדוק עבור כל מספר בתחום אם הוא מחלק את המספר הנתון ללא שארית.

ננסח זאת ניסוח ראשוני:

סריקת כל המספרים החיוביים בין 1 ועד למספר הנתון כקלט, ובדיקה עבור כל מספר אם הוא מחלק את המספר הנתון. אם כן, הצגתו כפלט.

? הרעיון המתואר מציג תת-משימה לביצוע-חוזר. מהי תת-משימה זו?

הרעיון המתואר כולל ביצוע-חוזר של התת-משימה הבאה, עבור כל מספר בתחום שבין 1 למספר הנתון:

אם המספר מחלק את המספר הנתון, הצגתו כפלט.

ננסח אלגוריתם לביטוי הרעיון המתואר.

בחירת משתנים

משתני האלגוריתם הם:

num – מטיפוס שלם, לשמירת נתון הקלט.

i – מטיפוס שלם, משתנה בקרה של ההוראה לביצוע-חוזר.

האלגוריתם

אלגוריתם 1:

1. קלוט מספר שלם חיובי ב-num
2. עבור כל מספר שלם חיובי i הקטן מ-num או שווה לו בצע:
   2.1. אם i מחלק את num ללא שארית
      2.1.1. הצג את ערכו של i

שימו ♥: מתאים להשתמש כאן בהוראה לביצוע-חוזר, שמספר הסיבובים בה מחושב מראש (ממומשת בשפת C# בלולאת **for**). בהוראה אנו מפרטים את תחום הערכים הנסרק באמצעות משתנה הבקרה של הלולאה (i).

באלגוריתם נעשה שימוש במשתנה הבקרה בגוף הלולאה. גוף הלולאה כולל בדיקה אם ערכו של משתנה הבקרה i הוא מחלק של num. כיוון שבמהלך ביצוע הלולאה ערכי משתנה הבקרה משתנים מ-1 עד num, הרי בדיקת החלוקה מתבצעת עבור כל מספר שלם בתחום 1 עד num.

יישום האלגוריתם

הנה התוכנית המיישמת את אלגוריתם 1:

```
/*
קלט: מספר שלם חיובי
פלט: כל המחלקים של המספר הנתון
*/
using System;
public class Divisors1
{
    public static void Main ()
    {
        int num;  // המספר הנתון
        int i;    // משתנה הבקרה
        Console.Write("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        Console.Write("The divisors of {0} are: ", num);
        for(i=1; i <= num ; i++)
```

```
            if (num % i == 0)
                    Console.Write(" {0}", i);
    } // Main
} // Divisors1
```

ניגש עתה לחישוב זמן-ביצוע האלגוריתם על ידי הערכת מספר ההוראות שיתבצעו במהלך הביצוע. כדי לבצע את ההערכה נתמקד במרכיב העיקרי באלגוריתם המשפיע על מספר ההוראות שיתבצעו.

❓ מהו המרכיב העיקרי באלגוריתם המעיד על מספר ההוראות שיתבצעו?

באלגוריתם אשר אינו כולל לולאה אין בעצם מרכיב בולט. מספר ההוראות שיתבצעו במהלך ביצוע האלגוריתם הוא לכל היותר מספר ההוראות באלגוריתם. לעומת זאת, באלגוריתם הכולל לולאה, הלולאה היא המרכיב העיקרי המעיד על מספר ההוראות שיתבצעו. זאת כיוון שייתכן שהלולאה תתבצע מספר רב של פעמים, ובכל ביצוע-חוזר יתבצעו שוב כל ההוראות שבגוף הלולאה.

מכיוון שייתכן שהלולאה תתבצע מספר רב של פעמים, הרי מספר ההוראות ש**יתבצעו** במהלך ריצת האלגוריתם עשוי להיות גדול בהרבה ממספר ההוראות ש**כתובות** באלגוריתם. למשל במהלך ביצוע אלגוריתם 1 עבור הקלט 1000, תתבצע ההוראה לביצוע-בתנאי שבגוף הלולאה 1000 פעמים. מספר זה גדול בהרבה ממספר ההוראות שבאלגוריתם.

בדרך כלל הלולאה תתבצע מספר שונה של פעמים עבור קלטים שונים. לכן כדי לתאר בצורה כללית את מספר הפעמים שלולאה תתבצע, יש לבטא מספר זה על פי הקלט.

❓ מהו מספר הפעמים שתתבצע הלולאה באלגוריתם 1 עבור קלט שערכו N ?

מספר הפעמים שהלולאה באלגוריתם 1 תתבצע עבור קלט שערכו N הוא N. הלולאה תתבצע פעם אחת עבור כל אחד מהערכים 1, 2, 3, ..., N למשתנה הבקרה i .

כיוון שלולאה היא המרכיב העיקרי המעיד על מספר הפעולות שיתבצעו באלגוריתם, נהוג להתייחס למספר זה כמדד לחישוב זמן-הביצוע. לכן נאמר שתוצאת חישוב זמן-הביצוע של אלגוריתם 1 היא שלולאת האלגוריתם תתבצע N פעמים עבור קלט שערכו N.

ננסה לראות אם ביכולתנו לפתח אלגוריתם יעיל יותר מאלגוריתם 1 מבחינת זמן-הביצוע.

❓ האם אפשר לפתח אלגוריתם שבו תהיה לולאה שתתבצע פחות מ-N פעמים עבור קלט שערכו N ?

כן. ניתן לפתח אלגוריתם ובו לולאה "יעילה" יותר. נראה זאת כעת.

אחד המאפיינים של חלוקה של מספרים שלמים היא שהמחלקים של מספר שלם חיובי num אינם גדולים מ-num/2, מלבד המספר num עצמו. ניתן לראות זאת בדוגמאות הקלט שנבדקו בשאלה 8.1. לכן בעצם אפשר "לחסוך" ולהריץ את משתנה הבקרה בלולאה שבאלגוריתם רק עד ל-num/2. נבטא רעיון זה באלגוריתם 2, והוא אלגוריתם יעיל יותר לפתרון הבעיה.

<u>**אלגוריתם 2:**</u>

1. *קלוט מספר שלם חיובי ב-num*
2. ***עבור כל** מספר שלם חיובי i הקטן מ-num/2 או שווה לו **בצע**:*
   2.1. ***אם** i מחלק את num ללא שארית*
      2.1.1. *הצג את ערכו של i*
3. *הצג את ערכו של num*

אמנם הוצאנו את הצגתו של num אל מחוץ ללולאה, אך עיקר העבודה נעשית בלולאה. באלגוריתם 2, תתבצע הלולאה רק N/2 פעמים עבור קלט שערכו N. זהו שיפור משמעותי, במיוחד עבור קלטים שערכם גדול (למשל 10,000).

יישום האלגוריתם

הנה התוכנית המיישמת את אלגוריתם 2:

```
/*
קלט: מספר שלם חיובי
פלט: כל המחלקים של המספר הנתון
*/
using System;
public class  Divisors2
{
    public static void Main ()
    {
        int num;  // המספר הנתון
        int i;          // משתנה הבקרה
        Console.Write("Enter a number: ");
        num = int.Parse(Console.ReadLine());
        Console.Write("The divisors of {0} are: ", num);
        for(i = 1; i <= num / 2 ; i++)
             if (num % i == 0)
                Console.Write("{0} ", i);
        Console.Write(num);
    } // Main
} // Divisors2
```

שיפרנו את הפתרון הראשון. האם נוכל להמשיך ולשפר גם את הפתרון השני?

? האם אפשר לפתח אלגוריתם שבו תהיה לולאה שתתבצע **פחות** מ-N/2 פעמים עבור קלט שערכו N?

כן. אפשר לפתח אלגוריתם ובו לולאה "יעילה" יותר.

לכל מספר שלם k שהוא מחלק של num, יש "בן-זוג" שלם num/k, שגם מחלק את num. (שהרי $num = k \cdot num/k$). אם באלגוריתם, עבור כל מחלק k שנמצא, נציג כפלט גם את num/k, לא נצטרך לעבור על כל המחלקים של num, אלא רק עד מחציתם (שהרי הצגנו כבר את "בני-הזוג").

? היכן נמצא קו המחצית של המחלקים? כלומר, מתי עלינו להפסיק את החיפוש כדי לא להציג מחלקים כפולים?

התשובה היא $\sqrt{num}$. כך נובע מהאבחנה הבאה: עבור כל זוג מחלקים k ו-num/k, לפחות אחד מהם אינו גדול מ- $\sqrt{num}$. מדוע? משום שאם שניהם גדולים מ- $\sqrt{num}$, אז מכפלתם גדולה מ-num! אם כך, אם נסרוק את כל המספרים מ-1 עד $\sqrt{num}$, ונציג עבור כל מחלק גם את "בן-זוגו" המחלק, למעשה נציג את כל המחלקים של num.

? מהו בן זוגו של $\sqrt{num}$?

זהו $\sqrt{num}$ בעצמו, משום שמכפלתו של $\sqrt{num}$ בעצמו שווה ל-num.

לכן משום שבן-זוגו של $\sqrt{num}$ הוא $\sqrt{num}$ עצמו, עלינו לשים לב לא להציג את $\sqrt{num}$ פעמיים כפלט.

לדוגמה: אם num=100 אז $\sqrt{num} = 10$. כל מחלקיו של 100 הקטנים מ-10 הם: 2, 4, ו-5. "בני זוגם" הם: 50, 25 ו-20, בהתאמה. אם נוסיף להם את $\sqrt{num} = 10$ נקבל את הרשימה המלאה של כל המחלקים של 100.

לכן בעצם ניתן להריץ את משתנה הבקרה בלולאה שבאלגוריתם עד ל- $\sqrt{num}$ בלבד. נבטא רעיון זה באלגוריתם 3, שהוא אלגוריתם יעיל יותר משני האלגוריתמים האחרים.

<u>אלגוריתם 3:</u>

1. קלוט מספר שלם חיובי ב-num
2. עבור כל מספר שלם חיובי i הקטן מ- $\sqrt{num}$ בצע:
   2.1. אם i מחלק את num ללא שארית
      2.1.1. הצג את ערכו של i
      2.1.2. הצג את ערכו של num/i
3. אם $\sqrt{num}$ מחלק את num ללא שארית
   3.1. הצג את ערכו של $\sqrt{num}$

כאמור, הוצאנו אל מחוץ ללולאה את הבדיקה של $\sqrt{num}$ כדי שלא נציג את ערכו פעמיים במקרה שהוא מחלק את num. לעומת זאת, אין צורך לטפל מחוץ בלולאה ב-num עצמו, בניגוד לאלגוריתם הקודם, משום שהוא בן הזוג של 1 ולכן יוצג כפלט בסיבוב הראשון בלולאה.

כעת שיפרנו את אלגוריתם 2 באופן משמעותי. עבור קלטים גדולים מאוד המספר $\sqrt{N}$ קטן משמעותית מהמספר N/2. למשל, אם N הוא 10,000 אז באלגוריתם 2 יהיו 5000 סיבובים בלולאה, בעוד שבאלגוריתם 3 יהיו 100 סיבובים בלבד!

ישמו בעצמכם את האלגוריתם בשפת C#.

סוף פתרון בעיה 1

נסכם את הנלמד מפתרון בעיה 1:

- באלגוריתם שאינו כולל לולאה מספר הפעולות שיתבצעו הוא לכל היותר מספר ההוראות באלגוריתם.
- באלגוריתם שכולל לולאה, מספר ההוראות שיתבצעו עשוי להיות גדול בהרבה ממספר הוראות האלגוריתם. מספר ההוראות שיתבצעו תלוי במספר הפעמים שהלולאה תתבצע. לכן מספר הפעמים שהלולאה תתבצע משמש כמדד לזמן-ביצוע האלגוריתם.
- מספר הפעמים שהלולאה תתבצע תלוי בדרך כלל בערכו של הקלט, ומבוטא באמצעות ערכו.

בפתרון בעיה 1 פיתחנו תחילה אלגוריתם אחד, ואחר כך פיתחנו אלגוריתם שני אשר היה יעיל יותר מבחינת מספר הפעמים של ביצוע הלולאה, כלומר מבחינת זמן-הביצוע. ולבסוף מצאנו אלגוריתם יעיל יותר גם ממנו. בפיתוח האלגוריתם השני השתמשנו בעובדה שאיברי הפלט, הלוא הם כל המחלקים של נתון הקלט N, אינם גדולים מ-N/2 מלבד N עצמו. בפיתוח האלגוריתם השלישי, היעיל מבין השלושה, השתמשנו בעובדה שאיברי הפלט, ניתנים לחלוקה לזוגות כך שמכפלת איברי כל זוג שווה ל-N ובכל זוג יש איבר אחד קטן מ- $\sqrt{N}$. העובדה האחרונה אפשרה לנו לכתוב לולאה אשר תתבצע $\sqrt{N}$ פעמים בלבד, במקום הלולאה המקורית שמתבצעת N פעמים.

- ♦ פיתוח אלגוריתם יעיל יותר נעשה תוך ניתוח וניצול טוב של מאפייני קלט-פלט. שימוש טוב במאפייני קלט-פלט חשוב לפיתוח לולאה אשר תתבצע מספר מועט של פעמים עד כמה שאפשר. כלומר השקעת מאמץ בניתוח מעמיק של הבעיה יכולה להתבטא אחר כך באלגוריתם שהוא משמעותית יעיל יותר.
- ♦ במהלך פתרון בעיה אלגוריתמית נשתדל לפתח אלגוריתם יעיל ככל האפשר מבחינת זמן-הביצוע. כלומר אלגוריתם שהלולאות שבו יתבצעו מספר פעמים מועט (לולאות "יעילות") עד כמה שניתן.

שאלה 8.2

ציינו עבור כל אחד מהקלטים הבאים את מספר הפעמים שתתבצע הלולאה בכל אחד משלושת האלגוריתמים שפיתחנו:

א. 1000

ב. 2000

שאלה 8.3

בקטע התוכנית הבא מחושבת המכפלה של שני נתוני קלט חיוביים שלמים בשימוש בפעולת חיבור בלבד:

```
x = int.Parse(Console.ReadLine());
y = int.Parse(Console.ReadLine());
sum = 0;
for(i = 1; i <= x ; i++)
    sum = sum + y;
Console.WriteLine( "The product is {0}", sum);
```

ידוע שאחד מנתוני הקלט גדול באופן משמעותי מהשני, אך לא ידוע אם זה הנתון הראשון או השני. השתמשו במאפיין זה של הקלט כדי לשנות את קטע התוכנית הנתון כך שיהיה יעיל ככל שניתן.

הדרכה: שימו לב שיש חשיבות לבחירת המשתנה אשר על פיו נקבע מספר הפעמים שתתבצע הלולאה.

מהו מספר הפעמים שתתבצע הלולאה של קטע התוכנית הנתון, ומהו מספר הפעמים שתתבצע הלולאה של קטע התוכנית החדש?

שאלה 8.4

פתחו אלגוריתם מבלי ליישמו שיהיה יעיל ככל האפשר, והקלט שלו הוא שני מספרים שלמים גדולים מ-1 שאינם מחלקים זה את זה, והפלט שלו הוא כל המספרים השלמים החיוביים המחלקים את שני מספרי הקלט. תארו לפי ערכו של הקלט את מספר הפעמים שתתבצע לולאת האלגוריתם.

שאלה 8.5 (מתקדמת)

נניח שבבעיה 1 הפלט הדרוש הוא כל המספרים השלמים החיובים הקטנים מ-N **שאינם** מחלקים את נתון הקלט N. מה יהיה מספר הפעמים שתתבצע הלולאה באלגוריתם לפתרון הבעיה החדשה?

שאלה 8.6

יש לפתח אלגוריתם אשר הקלט שלו הוא מספר שלם חיובי N, והפלט שלו הוא כל המספרים השלמים החיוביים אשר קטנים מ-N והשורש שלהם הוא מספר שלם.

למשל עבור הקלט 50 הפלט יהיה 49 36 25 16 9 4 1.

האלגוריתם הבא, הכולל משתנים מטיפוס שלם הוא פתרון אפשרי:

1. קלוט מספר שלם חיובי ב-num
2. **עבור כל** מספר שלם חיובי i שקטן מ-num **בצע**:
   2.1. **אם** השורש של i הוא מספר שלם
      2.1.1. הצג את ערכו של i

א. מהו מספר הפעמים שתתבצע הלולאה של האלגוריתם הנתון?

ב. ישמו בביטוי בוליאני בשפת C# את התנאי "השורש של i הוא מספר שלם"

ג. אפשר לכתוב אלגוריתם אשר זמן-הביצוע שלו יהיה קצר בהרבה מהאלגוריתם הנתון, וזאת ב"ייצור" מספרי הפלט, באמצעות העלאה בריבוע של כל המספרים השלמים אשר ריבועם קטן מן הקלט.

האלגוריתם החלקי הבא מבוסס על הרעיון המתואר:

1. קלוט מספר שלם חיובי ב-num
   1.1. **עבור כל** מספר שלם חיובי i הקטן מ-____________ **בצע**:
      1.1.1. הצג את ערכו של i^2

השלימו את האלגוריתם ותארו את מספר הפעמים שתתבצע הלולאה של אלגוריתם זה.

כל האלגוריתמים שניתחנו עד עתה בפרק כללו לולאה שמספר הביצועים שלה מחושב מראש ובה הורץ משתנה הבקרה מ-1 עד ערך כלשהו בקפיצות של 1. לכן קל היה לחשב את מספר הפעמים של ביצוע הלולאה.

בלולאות מורכבות יותר, בהן ערכו ההתחלתי של משתנה הבקרה אינו 1, והשינוי בערכו בין סיבוב לסיבוב הוא לאו דווקא 1, חישוב מספר הסיבובים עלול להיות מסובך יותר. בלולאת while שבה אין משתנה בקרה, חישוב זמן-הביצוע יכול להיות אף מורכב יותר. יש לספור את מספר הפעמים של ביצוע הלולאה במעקב אחר שינוי ערכי משתנים המתעדכנים בגוף הלולאה ומופיעים בתנאי הכניסה ללולאה.

שאלה 8.7

יש לפתח וליישם אלגוריתם אשר הקלט שלו הוא שני מספרים שלמים חיוביים, כך שהמספר השני גדול מהראשון. הפלט הדרוש הוא הכפולות של המספר הראשון אשר קטנות מהמספר השני או שוות לו. למשל עבור הקלט 1000 200 הפלט יהיה 1000 800 600 400 200.

משפטי התוכנית הבאים הם יישום של אלגוריתם לפתרון הבעיה:

```
x = int.Parse(Console.ReadLine());
y = int.Parse(Console.ReadLine());
i = x;
for (i = x; i <= y; i++)
    if (i % x == 0)
        Console.WriteLine(i);
```

א. כמה פעמים תתבצע הלולאה עבור הקלט 1000 200?

ב. תארו בצורה כללית את מספר הפעמים שתתבצע הלולאה על פי ערכי נתוני הקלט x ו-y.

ג. בפתרון המוצג `i` גדל בקפיצות של 1. ניתן לשפר את יעילות הפתרון הנתון בשינוי הקפיצות של `i` לקפיצות גדולות מ-1, קפיצות אשר מתאימות למרחק בין זוג מספרי פלט עוקבים (שימו לב שהמרחק בין כל זוג מספרי פלט עוקבים הוא אחיד). כתבו פתרון יעיל יותר המבוסס על הרעיון המתואר, ותארו את מספר הפעמים שתתבצע לולאת הפתרון החדש.

שאלה 8.8

נתונה לולאת ה-**for** הבאה:

```
for (i = 1 ; i <= 30000 ; i++)
    if (i % 500 == 0)
        Console.WriteLine(i);
```

א. מהו מספר הפעמים שתתבצע הלולאה?

ב. מהי מטרת הלולאה?

ג. כתבו לולאה יעילה הרבה יותר להשגת אותה המטרה. מהו מספר הפעמים שתתבצע הלולאה היעילה שכתבתם? פי כמה מספר זה קטן מתשובתכם בסעיף א?

סיכום

בפרקים הקודמים בחנו אלגוריתמים על פי קנה המידה **נכונות**. בפרק זה הכרנו קנה מידה חדש – **יעילות**.

יעילות של אלגוריתם נמדדת על פי "משאבי המחשב" הדרושים לביצוע האלגוריתם. משאבים אלה הם גודל המקום בזיכרון והזמן הדרוש לביצוע.

גודל המקום נמדד בעיקר על פי מספר המשתנים באלגוריתם.

זמן-הביצוע נקבע לפי מספר פעולות היסוד שיתבצעו במהלך ביצוע האלגוריתם.

פעולות היסוד הן: פעולות קלט, פעולות פלט ופעולות חישוב. בצורה פשטנית ניתן לומר, שכל הוראה באלגוריתם כוללת פעולת יסוד אחת, ומכאן – **מדידת זמן-הביצוע** של אלגוריתם נעשית על פי מספר ההוראות שיתבצעו במהלך ריצת האלגוריתם.

באלגוריתם אשר אינו כולל לולאה מספר ההוראות שיתבצעו במהלך הביצוע הוא לכל היותר מספר ההוראות באלגוריתם.

לעומת זאת באלגוריתם הכולל לולאה, מספר ההוראות שיתבצעו במהלך הביצוע אינו נקבע על פי מספר ההוראות באלגוריתם, אלא על פי מספר הפעמים שהלולאה תתבצע. מספר זה יכול להיות תלוי בקלט של האלגוריתם ואז הוא מבוטא באמצעות מאפייני הקלט.

כאשר נתונים שני אלגוריתמים שונים לפתרון בעיה, משווים את יעילותם מבחינת זמן-הביצוע לפי מספר הפעמים שהלולאות שבהם יתבצעו במהלך הרצת כל אלגוריתם.

בפיתוח אלגוריתם נשתדל לבחור לולאות שיתבצעו מספר פעמים מועט עד כמה שניתן, כלומר לולאות "יעילות" ככל האפשר. בחירת לולאה יעילה נעשית בניתוח ובניצול טוב של מאפייני הקשר בין הקלט לפלט.

יסודות מדעי המחשב 2

בשפת #C

תמר בניה וד״ר מיכל ארמוני – ראשי צוות הכתיבה

יעל בילצ׳יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי התבניות)

הילה קדמן (נספח)

ייעוץ: ד״ר דוד גינת

עריכה: לירון ברגר

תשס״ח 2007

אוניברסיטת תל-אביב החוג להוראת המדעים

מטה מל״מ המרכז הישראלי להוראת המדעים ע״ש עמוס דה-שליט

משרד החינוך האגף לתכנון ולפיתוח תכניות לימודים

יסודות מדעי המחשב 2 בשפת #C

תמר בניה וד"ר מיכל ארמוני– ראשי צוות הכתיבה

יעל בילצ'יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי תבניות)

הילה קדמן (נספח)

ייעוץ: ד"ר דוד גינת

עריכה: לירון ברגר

כל הזכויות שמורות © 2007

השראה הוצאה לאור, ת"ד 19022, חיפה 31190

טל': 04-8254752, פקס: 1534-8254752

E-Mail: books@hashraa.co.il

www.hashraa.co.il

השראה הוצאה לאור

מהדורה שנייה 2007

עיצוב העטיפה: טל גרין

אין לשכפל, להעתיק, לצלם, לתרגם, להקליט, לאחסן במאגר מידע כלשהו, לשדר או לקלוט בכל דרך או בכל אמצעי אלקטרוני, אופטי או מכני (לרבות צילום, הקלטה, אינטרנט, מחשב ודואר אלקטרוני), כל חלק שהוא מהחומר שבספר זה. שימוש מסחרי מכל סוג בחומר הכלול בספר זה אסור בהחלט, אלא ברשות מפורשת בכתב מהמו"ל ומהגורמים המפורטים להלן.

©
כל הזכויות שמורות
משרד החינוך

ISBN 965-90844-6-3 מסת"ב

פתח דבר

יחידת הלימוד "יסודות מדעי המחשב 2" היא יחידת המשך ליחידה "יסודות מדעי המחשב 1". היחידה מתבססת על הנלמד ב"יסודות מדעי המחשב 1" מחד, ומהווה שער ליחידה הרביעית "עיצוב תוכנה" מאידך. כמו "יסודות מדעי המחשב 1", גם יחידה זו משלבת שני ערוצים – ערוץ תיאורטי וערוץ יישומי. מטרתה של היחידה היא להעמיק בשני הערוצים, ובעיקר להרחיב את ההיכרות עם הפרדיגמה של תכנות מונחה עצמים ואת השימוש בתבניות אלגוריתמיות. חלק מהפרקים ביחידה זו מתבססים על "יסודות מדעי המחשב 1 ו- 2" שפותחו במכון ויצמן למדע בסוף שנות ה-90.

ביחידה זו מושם דגש על מגוון של שאלות ברמות קושי שונות תוך התמקדות בפיתוח פתרון, בניתוח פתרון, ובתיקון פתרון שגוי. בחלק מהשאלות התלמידים נדרשים להגדיר מחלקה מתאימה לפתרון בעיה, ליצור עצמים ממחלקה זו ולהפעיל את פעולותיהם. המטרה היא מצד אחד לכוון את התלמידים לחשיבה מונחית עצמים, ויחד עם זאת לשמור על ההיבט האלגוריתמי המהווה את הבסיס בכל סוג של תכנות (פרוצדורלי, פונקציונלי, מונחה עצמים וכו').

ביחידה זו חמישה פרקים: הפרק הראשון מציג את המחלקה מחרוזת והוא כולל מבוא ראשוני לעצמים. הפרק השני מציג מערכים ומתמקד בתבניות אלגוריתמיות נפוצות. פירוט מלא של התבניות מופיע באתר הספר www.tau.ac.il/~csedu/yesodot.html. הפרק השלישי מרחיב בנושא של חשיבה מונחית עצמים. בפרק זה ילמדו התלמידים להגדיר מחלקה ולהשתמש בה. התלמידים יכירו מושגים בסיסיים בתכנות מונחה עצמים כגון תכונות ופעולות של עצם, מאפייני גישה, פעולה-בונה ועוד. הפרק הרביעי מרחיב בכיוון של חשיבה אלגוריתמית, הוא מציג מבנה נתונים של מערך דו-ממדי ותבניות לחיפוש, למיון ולמיזוג ומציג שילוב של תבניות.

הפרק האחרון מוקדש לפתרון בעיות והוא משלב את כל החומר שנלמד בשתי היחידות ומיישם אותו בפתרון בעיות אלגוריתמיות. בנוסף, הפרק מציג מגוון רחב של בעיות מתוך פרק ג של בחינות הבגרות בשנים האחרונות. לכל בעיה מצורפות הנחיות מתאימות, על מנת להקנות לתלמיד את הניסיון הנדרש בניתוח ובפתרון בעיות אלגוריתמיות בסביבה מונחית עצמים. תודתנו נתונה למשרד החינוך על השימוש בשאלות מתוך בחינות הבגרות בשנים האחרונות.

תודות. ספר זה פותח בתמיכת מפמ"ר מדעי המחשב במשרד החינוך ד"ר אבי כהן וחברי שתי ועדות המקצוע האחרונות להוראת מדעי המחשב – הועדה בראשות פרופ' עמיהוד אמיר והועדה (הנוכחית) בראשות פרופ' יהודית גל-עזר. תודתנו נתונה להם על תמיכתם ועל הערותיהם. בנוסף, לאורך הספר משולבת התייחסות מפורשת לתבניות בפיתוח ובניתוח של אלגוריתמים. ההתייחסות מבוססת על הספר "תבניות במדעי המחשב" שפיתחו חברי הקבוצה להוראת מדעי-המחשב בחוג להוראת המדעים באוניברסיטת תל-אביב בשנת 2001.

תוכן עניינים

פרק 9 – המחלקה מחרוזת (string) **1**

9.1 היכרות ראשונית עם המחלקה string 1

9.2 תכונות של עצם ותכונת האורך של מחרוזת 3

9.3 ביצוע פעולות על מחרוזות 5

שרשור מחרוזות 9

ומה בפנים? פעולות המסתכלות אל תוך הקנקן 10

9.4 הוראת השמה במחרוזות 12

סיכום 20

רשימת פעולות על מחרוזות 21

פרק 10 – מערכים **25**

10.1 מערך ואיברי מערך 25

10.2 חריגה מגבולות מערך 41

10.3 קשרים בין מערכים 43

עיבוד מערכים במקביל ובקצב התקדמות זהה 43

עיבוד מערכים במקביל בקצב התקדמות לא זהה 44

10.4 מערך מונים 46

10.5 מערך צוברים 50

10.6 יעילות מקום 50

שאלות נוספות 56

סיכום 57

סיכום מרכיבי שפת #C שנלמדו בפרק 10 58

תבניות – פרק 10 59

מערך מונים, מערך צוברים, חישוב שכיח, הזזה מעגלית בסדרה, הזזה של תת-סדרה,
היפוך סדר הערכים בסדרה

פרק 11 – מחלקות ועצמים: הרחבה והעמקה **63**

11.1 מחלקה - הגדרה ושימוש 63

11.2 פעולות גישה 75

11.3 תכונות מורכבות 85

סיכום 110

סיכום מרכיבי שפת #C שנלמדו בפרק 11 111

פרק 12 – תבניות אלגוריתמיות **115**

12.1 מערך דו-ממדי 115

12.2 חיפוש בינרי 131

12.3 מיונים 136

מיון בחירה ... 136
מיון הכנסה ... 139
מיון בועות ... 146
12.4 מיזוג ... 147
סיכום ... 150
סיכום מרכיבי שפת C# שנלמדו בפרק 12 ... 151
שאלות נוספות ... 151
פרק 13 – פתרון בעיות ... 155
שאלות נוספות ... 181
סיכום ... 188
אינדקס ... 189

תוכן יסודות 1

פרק 1 – מבוא

פרק 2 – פתרון בעיות אלגוריתמיות.

פרק 3 – מודל חישוב בסיסי.

פרק 4 – הרחבה בפיתוח אלגוריתמים.

פרק 5 – ביצוע מותנה.

פרק 6 – נכונות אלגוריתמים.

פרק 7 – ביצוע-חוזר.

פרק 8 – יעילות של אלגוריתמים.

פרק 9 – המחלקה מחרוזת (string)

בתוכניות שכתבנו עד כה השתמשנו בטיפוסים שונים המוגדרים בשפת C#: שלם, ממשי, תווי ובוליאני. יכולנו להגדיר משתנים מטיפוסים אלו ולבצע עליהם פעולות שונות (קלט, פלט, חישובים וכו'). עם זאת מרבית התוכניות שכתבנו התייחסו גם ל**מחרוזות**, סדרות של תווים. עד עתה השתמשנו במחרוזות כאשר רצינו להדפיס הודעות למשתמש ותחמנו אותן בגרשיים, למשל בהוראה:

```
Console.Write("Enter two numbers:");
```

ההודעה "`Enter two numbers:`" היא מחרוזת.

הנה דוגמאות נוספות למחרוזות בשפת C#: `"Hello, How Are You?"`, `"453"`, `"J"`, `""`
(האחרונה היא מחרוזת ריקה, שאינה מכילה אף תו).

בנוסף להדפסה נרצה לעתים להפעיל על מחרוזות פעולות נוספות. בפרק זה נראה כיצד ניתן לקלוט מחרוזות מהמשתמש, ולעבד אותן. אם עד עתה קלטנו מספרים או תווים בודדים בלבד, עכשיו נוכל לבקש מהמשתמש להקיש את שמו, את כתובתו וכדומה ולבצע פעולות שונות על המחרוזות הנקלטות.

9.1 היכרות ראשונית עם המחלקה string

בשפת C# מוגדרת **מחלקה** בשם **`string`** שבאמצעותה אפשר ליצור מחרוזות. משתנה מטיפוס מחרוזת אינו משתנה פשוט כמו **int** או כמו **char** אלא **מופע (unstance)** של המחלקה **string**.

הצהרה על משתנה מטיפוס מחרוזת מתבצעת כך:

```
string s1;
```

אמנם הצהרה על עצמים דומה להצהרה על משתנים מטיפוסים רגילים. אבל בשונה ממשתנים מטיפוסים רגילים, עלינו ליצור עצמים. יצירת עצם כוללת הקצאת שטח בזיכרון עבור העצם ואת אתחולו. עבור משתנים רגילים, הקצאת מקום בזיכרון נעשית אוטומטית עם ההצהרה עליהם ואילו עבור עצמים הקצאת הזיכרון והאתחול מתבצעים באמצעות הפעולה **new**. מכיוון שהשימוש במחרוזות נפוץ כל כך, שפת C# מאפשרת ליצור ולאתחל מחרוזת בצורה ישירה ללא שימוש בפעולה **new**, למשל כך:

```
s1 = "";
```

פעולה זו יוצרת עצם בשם `s1` שמפנה לשטח בזיכרון המכיל מחרוזת מחרוזת ריקה `""`. אם נרצה לאתחל את העצם בערך כלשהו, נוכל לעשות זאת בציון הערך הרצוי בזמן היצירה, למשל כך:

```
s1 = "abc";
```

כעת העצם `s1` מפנה לשטח בזיכרון המכיל את המחרוזת `"abc"`.

בדומה למשתנים פשוטים, אפשר לאחד את ההצהרה ואת האתחול כך:

```
string s1 = "hello";
string s2 = "good morning";
```

בהמשך הפרק נכיר פעולות שונות שאפשר לבצע על עצמים מטיפוס **string**.

בשפת C# מחרוזת היא עצם של המחלקה **string**.

הצהרה על מחרוזות מתבצעת באופן דומה להצהרה על משתנים רגילים, למשל:

```
string s1;
```

כדי שניתן יהיה להשתמש בעצם מהמחלקה מחרוזת, יש ליצור את העצם. **יצירת העצם** כוללת **הקצאת שטח בזיכרון ואתחול העצם**.

יצירת העצם נעשית באמצעות השמה פשוטה של סדרת תווים בגרשיים:

```
string s2 = "good morning";
```

בעיה 1

מטרת הבעיה ופתרונה: הצגת קלט של מחרוזת.

> פתחו וישמו אלגוריתם שיקבל כקלט את שמכם ויציג Your name is: ומיד אחר כך את השם שנקלט.

בחירת משתנים

❓ באיזה טיפוס נשתמש כדי לשמור את השם שנקלט?

כיוון שעלינו לקלוט מילה נשתמש במחלקה מחרוזת. אם כך רשימת המשתנים תכלול את העצם name מהמחלקה מחרוזת.

האלגוריתם:

1. *קלוט מחרוזת ב*-name
2. *הצג כפלט* "Your name is: "
3. *הצג כפלט את* name

יישום האלגוריתם

עד כה ראינו כיצד לקלוט ערכים מטיפוסים פשוטים – שלמים, ממשיים ותווים. קליטת מחרוזת נעשית בצורה דומה. את הוראה 1 נוכל לממש במשפטים:

```
name = Console.ReadLine();
```

בניגוד להוראות קלט אחרות שהשתמשנו בהן, הפעם לא נשתמש בפעולה נוספת (למשל ב-`int.Parse`) כדי לפרש את הקלט ולתרגמו לערך מטיפוס מסוים, אלא נקלוט את שורת המחרוזת שיקליד המשתמש ונשמור אותה כמו שהיא.

הפעולה `ReadLine` קולטת את כל התווים עד סוף השורה ומחזירה עצם מסוג מחרוזת. למשל אם המשתמש הקליד:

```
my name is Moshe
```

העצם `name` יכיל את המחרוזת: `"my name is Moshe"`

התוכנית המלאה

```
/*
קלט: שם, הנקלט כמחרוזת
פלט: השם, מלווה בהודעה מקדימה
*/
using System;
```

```
public class YourName
{
    public static void Main ()
    {
        string name;    // הצהרה על עצם מהמחלקה מחרוזת
        // הוראת קלט + הקצאת מקום בזיכרון
        Console.Write("Enter your name: ");
        name = Console.ReadLine();
        // פלט
        Console.WriteLine("your name is: {0}", name);
    } // Main
} // class YourName
```

שימו ♥: כמו פעולות קלט של ערכים מטיפוסים פשוטים, גם פעולת קלט למחרוזת היא למעשה פעולה המחזירה את הערך הנקלט. הפעולה קולטת מחרוזת מהקלט, ויוצרת עצם חדש מהמחלקה מחרוזת. פעולת הקלט מקצה עבורו מקום בזיכרון ושומרת בו את מחרוזת הקלט. הפעולה `Console.readLine()`, מחזירה הפניה למחרוזת החדשה שהוקצתה.

בעת ביצוע ההוראה:

```
name = Console.ReadLine();
```

אנו מבצעים השמה של המחרוזת הנקלטת במשתנה name. בעקבות ההשמה מַפְנֶה `name` אל אותו שטח זיכרון שהוקצה על ידי פעולת הקלט.

סוף פתרון בעיה 1

ניתן לקלוט מחרוזת באמצעות הפעולה `Console.ReadLine`. פעולת הקלט מקצה שטח בזיכרון עבור המחרוזת החדשה.

9.2 תכונות של עצם ותכונת האורך של מחרוזת

לכל מחרוזת יש אורך. למשל אורכה של המחרוזת "abc" הוא 3, ואורכה של המחרוזת הריקה "" הוא 0. ניתן לומר כי אורך הוא אחד המאפיינים למחרוזת מסוימת, ולכן הוא מוגדר כ**תכונה** של המחרוזת. וכך עבור עצמים מהמחלקה מחרוזת מוגדרת התכונה `Length` השומרת לכל מחרוזת את אורכה.

אם `s1` הוא עצם מהמחלקה מחרוזת, אז ערכו של הביטוי `s1.Length` הוא אורכה של המחרוזת `s1`.

שימו ♥: את התכונה `Length` ניתן לקרוא בלבד. לא ניתן לשנות את ערכה בהשמה. ניסיון לשנות את ערך התכונה יגרום לשגיאת הידור.

בעיה 2

מטרת הבעיה ופתרונה: הדגמת השימוש בתכונה של מחרוזת. הכרת התכונה `Length` ואופן הגישה לתו בתוך מחרוזת.

פתחו וישמו אלגוריתם אשר יקבל כקלט מחרוזת. אם אורך המחרוזת הוא זוגי יוצג כפלט התו הראשון במחרוזת ואם אורך המחרוזת הוא אי-זוגי יוצג התו האחרון במחרוזת. למשל, עבור הקלט shalom יוצג התו: 's'. ועבור הקלט dog יוצג התו: 'g'.

שימו ♥: בשפת C# מיקום התווים במחרוזת מתחיל ב-0. כלומר, התו הראשון במחרוזת נמצא במקום 0, התו השני במחרוזת הוא במקום 1, השלישי במקום 2 וכן הלאה.

? אם התו הראשון נמצא במקום 0, באיזה מקום נמצא התו האחרון?

נתבונן למשל במחרוזת "abcd" שאורכה 4. התו האחרון בה הוא 'd' והוא נמצא במקום 3.

מיקום התווים במחרוזת מתחיל ב-0. בכל מחרוזת, אם נסמן את אורך המחרוזת ב-len אז התו האחרון במחרוזת נמצא במקום len-1.

בחירת משתנים

נשתמש במשתנים הבאים:

str – עצם מהמחלקה מחרוזת

letter – האות המבוקשת, מטיפוס **char**

len – לשמירת אורך המחרוזת, מטיפוס שלם

האלגוריתם

1. *קלוט מחרוזת ב-*str
2. *השם את אורכה של המחרוזת* str *ב-*len
3. *אם* len *זוגי*
   3.1 *השם ב-*letter *את התו הנמצא במקום 0 ב-*str
4. *אחרת*
   4.1 *השם ב-*letter *את התו הנמצא במקום* len-1 *ב-*str
5. *הצג כפלט את ערכו של* letter

יישום האלגוריתם

? כדי לבדוק את אורכה של המחרוזת נשתמש בתכונה `Length`, כיצד נדע מהו התו שנמצא במקום מסוים במחרוזת?

פנייה לתו מסוים במחרוזת נעשית בסוגריים מרובעים `[]`. למשל, כדי לפנות אל התו שנמצא במקום 0 במחרוזת `str` נכתוב `str[0]`. התו שנמצא במקום האחרון במחרוזת `str` הוא `str[str.Length-1]`.

שימו ♥: את התווים שבמחרוזת ניתן לקרוא בלבד. לא ניתן לשנות את ערכם בהשמה. ניסיון לשנות את ערך התו יגרום לשגיאת הידור.

שימו ♥: פנייה לתו שאינו בתחום המחרוזת יגרום לחריגה במהלך ריצת התכנית.

התוכנית המלאה

```
/*
קלט: מחרוזת
פלט: התו הראשון או האחרון במחרוזת, בהתאם לאורכה (זוגי או אי-זוגי)
*/
using System;
public class IsEven
{
    public static void Main()
```

```
    {
        string str;
        int len;
        char letter;
        Console.Write("Enter a string: ");
        str = Console.ReadLine();
        len = str.Length;
        if (len % 2 == 0)
            letter = str[0];
        else
            letter = str[len - 1];
        Console.WriteLine("The letter is: {0}", letter);
    } // Main
} // class IsEven
```

סוף פתרון בעיה 2

נסכם את המושג המרכזי שהוצג בפתרון בעיה 2:

> במחלקה מוגדרות **תכונות** שהן משתנים המוגדרים עבור עצמים מהמחלקה.
>
> כדי לגשת ל**ערך התכונה** כותבים כך:
>
> `שם העצם.שם התכונה`
>
> **התכונה** `Length` מוגדרת לעצמים מהמחלקה מחרוזת. התכונה שומרת את אורך המחרוזת. ערכה של התכונה אינו ניתן לשינוי.

שאלה 9.1

פתחו וישמו אלגוריתם שמקבל כקלט רשימה של 100 מחרוזות. האלגוריתם מחשב בכמה מחרוזות מתוך ה-100 התו הראשון שווה לתו האחרון, ומציג את הערך שחישב.

שאלה 9.2

פתחו אלגוריתם שיקבל כקלט מחרוזת ותו. פלט האלגוריתם יהיה מספר ההופעות של התו הנתון בתוך המחרוזת הנתונה. ישמו את האלגוריתם בשפת C#.

הדרכה: מאחר שניתן לברר את אורך המחרוזת הנתונה ניתן להשתמש בהוראה לביצוע-חוזר שמספר הסיבובים בה ידוע מראש (לולאת **for**).

9.3 ביצוע פעולות על מחרוזות

כדי לבצע פעולות על עצמים יש לרשום את שם העצם, אחריו את סימן הנקודה ולאחר מכן את שם הפעולה. למשל, כדי להגריל מספר הפעלנו את הפעולה Next על עצם מסוג Random כך:

```
Random rnd = new Random();
int num;
num = rnd.Next(100);
```

במקרה זה הפעולה Next מחזירה מספר אקראי בין 0 ל-99 הנשמר במשתנה num.

באופן דומה אפשר להפעיל פעולות שונות על עצמים מסוג מחרוזת. אחת הפעולות המוגדרות עבור מחרוזות היא הפעולה `CompareTo` המשווה מחרוזת למחרוזת נוספת. אם `s1` הוא עצם

מהמחלקה מחרוזת, ו-s2 הוא עצם נוסף מהמחלקה מחרוזת, אז הביטוי `s1.CompareTo(s2)` מפעיל את הפעולה `CompareTo` המשווה את `s1` למחרוזת נוספת (`s2`) שהתקבלה כפרמטר.

הפעולה `CompareTo` מחזירה את תוצאת ההשוואה כערך שלם באופן הבא:

- ♦ אם המחרוזות שוות יוחזר הערך 0.
- ♦ אם המחרוזת `s1` שמופעלת עליה הפעולה, **קודמת** למחרוזת `s2` שהתקבלה כפרמטר בסדר מילוני, יוחזר מספר שלם **שלילי**.
- ♦ אם המחרוזת `s1` שמופעלת עליה הפעולה, מופיעה **אחרי** המחרוזת `s2` שהתקבלה כפרמטר בסדר מילוני, יוחזר מספר שלם **חיובי**.

הסוגריים שאחרי שם הפעולה נועדו להעברת ערכים (פרמטרים) שהפעולה נזקקת להם. במקרה של הפעולה `CompareTo` מועבר פרמטר מסוג מחרוזת. בהמשך הפרק נכיר פעולות נוספות של המחלקה מחרוזת.

בעיה 3

מטרת הבעיה ופתרונה: הדגמת השימוש בפעולה להשוואת מחרוזות במחלקה **string**, ושילוב מחרוזות בתבנית מנייה.

> בסיום בחינות הקבלה למקהלת הזמר העירונית יוצא הבוחן הראשי וקורא את רשימת המועמדים שהתקבלו על פי סדר הא"ב. מועמד אשר שומע את שמו יודע כי התקבל ללהקה. מועמד ששמו עדיין לא נקרא, אשר שומע שם אחר שמופיע אחריו בסדר הא"ב, יודע כי לא התקבל ללהקה.
> פתחו אלגוריתם אשר הקלט שלו הוא שם מועמד, ולאחר מכן רשימת שמות שקורא הבוחן על פי סדר הא"ב. פלט האלגוריתם הוא אם המועמד התקבל, ומספר השמות שהיה עליו לשמוע לפני שהגיע למסקנה אם התקבל או לא. ישמו את האלגוריתם בשפת #C.

פירוק הבעיה לתת-משימות

למעשה האלגוריתם צריך להשוות מחרוזות ובנוסף לבצע מנייה. אלגוריתם זה דומה לאלגוריתמים שפיתחנו בפרקים קודמים. אם כך, נפרק את הבעיה לתת-משימות הבאות:

1. קליטת שם המועמד
2. קליטת שמות המועמדים שהתקבלו והשוואתם לשם המועמד
3. מניית מספר השמות שנקלטו
4. הצגה אם המועמד התקבל והצגת מספר השמות שהיה עליו לשמוע

תת-משימה 3 תבוצע בדומה למשימות מנייה אחרות, ההבדל הוא כמובן **בטיפוס** הערכים הנמנים – מחרוזת במקרה זה – ובאופן ביצוע הפעולות עליהם. בשלב יישום האלגוריתם נראה כיצד לעשות זאת.

בחירת משתנים

name – שם המועמד, עצם מהמחלקה מחרוזת
str – השם התורן מהקלט, עצם מהמחלקה מחרוזת
counter – למניית מספר המחרוזות, מטיפוס שלם

האלגוריתם

1. אתחל את counter ב-1
2. קלוט מחרוזת ב-name
3. קלוט מחרוזת ב-str
4. כל עוד name > str (בסדר מילוני) בצע:
   - 4.1. הגדל את ערכו של counter ב-1
   - 4.2. קלוט מחרוזת ב-str
5. אם name = str
   - 5.1. הצג כפלט "המועמד <name> התקבל"
6. אחרת
   - 6.1. הצג כפלט "המועמד <name> לא התקבל"
7. הצג את ערכו של counter

יישום האלגוריתם

? כיצד נבדוק אם שם המועמד גדול על-פי סדר מילוני מהמחרוזת הנקלטת?

כאמור, הפעולה `CompareTo` בודקת את היחס המילוני שבין שתי מחרוזות. אם שם המועמד `name` גדול מהמחרוזת הנקלטת `str`, הפעולה `name.CompareTo(str)` תחזיר ערך שלם חיובי, ואילו אם שם המועמד קטן מהמחרוזת הנקלטת, פעולה זו תחזיר ערך שלם שלילי. אם המחרוזות שוות הפעולה תחזיר 0.

? כיצד נבדוק אם המחרוזת האחרונה שנקלטה שווה לשם המועמד?

אפשר להשתמש שוב בפעולה `CompareTo` ולבדוק אם היא מחזירה את הערך 0. לחילופין, אפשר לבדוק שוויון בין שתי מחרוזות באמצעות הפעולה `Equals` הבודקת שוויון של המחרוזת שעליה היא מופעלת למחרוזת המתקבלת כפרמטר. למשל, הביטוי `name.Equals(str)` יחזיר **true** אם המחרוזת `str` שווה למחרוזת `name`, אחרת יוחזר **false**.

בנוסף לכך, בשפת C# אפשר לבדוק שוויון בין שתי מחרוזות כפי שבודקים שוויון בין שני משתנים מאותו הסוג: על ידי פעולת השוואה == או =! (לא שווה). אם כך, את הביטוי הבוליאני שבהוראה 5 אפשר ליישם כך:

```
name == str
```

שימו ♥: את המונה נאתחל ב-1 כיוון שמחוץ ללולאה אנו קולטים את השם הראשון שנקרא.

התוכנית המלאה

```
/*
קלט: שם מועמד ורשימה ממוינת של שמות
פלט: האם המועמד התקבל ומספר השמות שהיה עליו לשמוע
*/
using System;
public class NameFinder
{
    public static void Main ()
    {
        // הגדרת משתנים ואתחולם
        string name;
        string str;
        int counter = 1;
```

```
        // קלט שם מבוקש
        Console.Write("Enter the candidate name: ");
        name = Console.ReadLine();
        // לולאת זקיף
        Console.Write("Enter first winner name: ");
        str = Console.ReadLine();
        // כל עוד לא עברנו את שם המועמד
        while (name.CompareTo(str) > 0)
        {
            counter++;
            Console.Write("Enter next winner name: ");
            str = Console.ReadLine();
        } // while
        if (name == str)
             Console.WriteLine("{0} is accepted", name);
        else
             Console.WriteLine("{0} is not accepted", name);
        Console.WriteLine("{0} Names", counter);
    } // Main
} // class NameFinder
```

שימו ♥: לולאת ה-**while** ממשיכה כל עוד מחרוזות הקלט קטנות משם המועמד. קליטת מחרוזת שווה לשם המועמד או גדולה ממנה גורמת ליציאה מהלולאה. לכן, אחרי הלולאה עלינו לבדוק את סיבת היציאה מהלולאה: האם המחרוזת האחרונה שנקלטה שווה לשם המועמד או גדולה ממנה.

סוף פתרון בעיה 3

בסוף הפרק תוכלו למצוא טבלה המפרטת את הפעולות השכיחות המוגדרות ב-C# עבור עצמים מהמחלקה **`string`**.

העמודה הימנית בטבלת הפעולות מתארת את שם הפעולה ואת רשימת הפרמטרים שהיא מצפה לקבל (בתוך סוגריים). כפי שניתן לראות, יש פעולות בטבלה כגון `ToLower` אשר אינן מצפות לקבל ערך כלשהו, ולכן הסוגריים ריקים. לעומתן, הפעולה `Equals` מצפה לקבל פרמטר אחד שהוא עצם מהמחלקה מחרוזת, והפעולה `IndexOf` מצפה לקבל עצם מהמחלקה מחרוזת או לקבל תו.

העמודה השלישית מתארת את טיפוס הערך המוחזר כתוצאה מהפעלת הפעולה. בדומה לפעולות המתמטיות שלמדנו בפרק 4, גם כאן יש לשים לב לטיפוס הערך המוחזר ולוודא התאמה בינו ובין הפעולות שנבצע עליו. כלומר כאשר נשים בתוך משתנה ערך שמוחזר מפעולה, עלינו לוודא התאמה בין טיפוס הערך המוחזר לטיפוס המשתנה שההשמה תתבצע בו.

מחלקות מגדירות אוסף של **פעולות** שאפשר להפעיל על עצמים מהמחלקה.

הפעלת פעולה נכתבת כך (משמאל לימין):

`שם העצם.שם הפעולה(פרמטרים)`

יש לבדוק היטב בתיאור הפעולה אילו ערכים היא מצפה לקבל ומה טיפוס הערך שהיא מחזירה.

שאלה 9.3

פתחו אלגוריתם המקבל כקלט רשימת מחרוזות המסתיימת במחרוזת "***". פלט האלגוריתם יהיה אורך המחרוזת הארוכה ביותר ברשימה. ישמו את האלגוריתם בשפת #C.

שאלה 9.4

פתחו אלגוריתם שיקבל כקלט שתי מחרוזות ויציג אותן לפי סדר הא"ב. ישמו את האלגוריתם בשפת #C.

שאלה 9.5

פתחו אלגוריתם שהקלט שלו הוא שלוש מחרוזות, והפלט שלו הוא שלוש המחרוזות לפי סדר מילוני. ישמו את האלגוריתם בשפת #C.

למשל עבור הקלט: good today apple הפלט המתאים הוא: today good apple.

שרשור מחרוזות

פעולה נוספת ושימושית מאוד על מחרוזות היא פעולת **השִׁרְשׁוּר**. פעולה זו מקבלת שתי מחרוזות ויוצרת מחרוזת חדשה, המורכבת מהמחרוזת הראשונה ואחריה מוצמדת המחרוזת השנייה. הפעולה אינה משנה את המחרוזות המקוריות.

בשפת #C מתבצע שרשור באמצעות סימן הפעולה +. כלומר, אם `s1` ו-`s2` הן מחרוזות, אז כדי לשרשר אותן זו לזו נכתוב את הביטוי `s1+s2`. למשל, אם ב-`s1` נמצאת המחרוזת `"he"` וב-`s2` נמצאת המחרוזת `"llo"`, אז `s1+s2` היא המחרוזת החדשה `"hello"`.

שימו ♥: פעולת השרשור אינה פעולה של המחלקה מחרוזת ולכן אופן הפעלתה שונה מזה של פעולות כמו `Equals`, `ToUpper` או כמו פעולות אחרות השייכות למחלקה. היא אינה מופעלת על עצם מהמחלקה, ולכן אין שימוש בסימן הנקודה. אופן הפעלת פעולת השרשור על מחרוזות דומה לאופן הפעלת פעולות מתמטיות על טיפוסים רגילים בשפה.

ניתן לשרשר למחרוזות ערכים מטיפוסים שונים. למשל הביטוי:

```
"The number is " + number
```

הוא ביטוי חוקי. אם, למשל, ערכו של המשתנה `number` הוא 3, אז ערך הביטוי הוא המחרוזת הנוצרת משרשור של המחרוזת `"3"` למחרוזת `"The number is: "`. המחרוזת החדשה הנוצרת עקב השרשור אם כך היא: `"The number is: 3"`.

שימו ♥: מאחר שבמשתנה `number` יש ערך מטיפוס שלם יש להמיר אותו למחרוזת. ואכן טרם השרשור התבצעה פעולת המרה של הערך המספרי למחרוזת המתאימה לו באופן אוטומטי. כך קורה תמיד כאשר נשרשר ערך מטיפוס שאינו מחרוזת למחרוזת.

הפעולה + משמשת **לשרשור מחרוזות** באופן הבא: `מחרוזת1 + מחרוזת2`

פעולת השרשור יוצרת מחרוזת חדשה ומקצה עבורה מקום. היא אינה משפיעה על המחרוזות המקוריות.

כאשר משרשרים למחרוזות ערך שאינו מחרוזת, הוא מומר תחילה למחרוזת ואז מתבצעת פעולת השרשור.

שאלה 9.6

פתחו אלגוריתם שיקבל כקלט 100 מחרוזות. עבור כל זוג מחרוזות תוצג כפלט מחרוזת שהיא שרשור של שתי המחרוזות עם הסימן '@' ביניהן (כלומר בסך הכול תוצגנה 50 מחרוזות). ישמו את האלגוריתם בשפת C#.

ומה בפנים? פעולות המסתכלות אל תוך הקנקן

המחלקה מחרוזת מגדירה פעולות שונות המאפשרות להסתכל פנימה לתוך המחרוזת ולהתייחס לתווים המרכיבים אותה. פעולות כאלו מסייעות מאוד בפתרון בעיות שונות, כפי שמדגימה הבעיה הבאה:

בעיה 4

מטרת הבעיה ופתרונה: הדגמת השימוש בפעולות המחלצות מידע מתוך מחרוזת.

פתחו אלגוריתם שהקלט שלו הוא מחרוזת המהווה משפט באנגלית, והפלט הוא האות **האחרונה** של המילה **הראשונה** במשפט ומספר המילים במשפט.
למשל, עבור הקלט: !Welcome to Israel and have a nice day הפלט המתאים הוא 8 e.
ישמו את האלגוריתם בשפת C#.
אפשר להניח שהמשפט מכיל לפחות מילה אחת.

ניתוח הבעיה בעזרת דוגמאות

נתבונן במשפט שניתן כדוגמה: !Welcome to Israel and have a nice day.

? כיצד אפשר לדעת כמה מילים יש במשפט הנתון?

כיוון שלפני כל מילה פרט למילה הראשונה יש רווח נוכל למנות את מספר הרווחים במחרוזת ולהוסיף 1. למשל, במשפט !Welcome to Israel and have a nice day יש 7 רווחים, ולכן 8 מילים. (זאת בהנחה שבין מילה למילה יש רווח אחד בלבד!)

? מהי האות האחרונה במילה הראשונה?

במקרה שהמשפט מכיל מילה אחת בלבד, האות האחרונה במילה הראשונה היא האות האחרונה במחרוזת. בכל שאר המקרים, האות האחרונה במילה הראשונה היא האות שנמצאת מיד לפני הרווח הראשון.

פירוק הבעיה לתת-משימות

לשם פתרון הבעיה נפתור את התת-משימות הבאות:

1. קליטת המחרוזת
2. מניית מספר הרווחים במחרוזת שנקלטה
3. חישוב מספר המילים במחרוזת
4. מציאת התו האחרון של המילה הראשונה במחרוזת
5. הצגה של מספר המילים במחרוזת ושל התו שנמצא בתת-משימה 4

בחירת משתנים

sentence – מחרוזת לשמירת המשפט שנקלט

numOfSpaces –שלם, מונה את מספר הרווחים במשפט

numOfWords – שלם, שומר את מספר המילים במשפט

placeOfLastLetter – שלם, שומר את מיקומו של התו האחרון במילה הראשונה

יישום האלגוריתם

כדי למנות את מספר הרווחים במחרוזת, נעבור על כל תו במחרוזת ונבדוק אם הוא שווה לרווח. לצורך בדיקת התו במיקום ה-i נשתמש בסימון [i] המחזיר את התו במקום המבוקש:

```
if (sentence[i]==' ')
```

שימו ♥: הפנייה לתו במחרוזת מחזירה ערך מטיפוס **char** ולא מטיפוס מחרוזת, ולכן יש להשוות את הערך המוחזר לתו רווח ' ' (ולא למחרוזת המכילה את התו רווח " ").

כדי למצוא את התו האחרון של המילה הראשונה נבדוק את מספר המילים במחרוזת. אם המחרוזת מורכבת ממילה אחת בלבד נחלץ את התו במקום האחרון, אחרת נחלץ את התו שנמצא לפני הרווח הראשון.

כדי למצוא את מיקומו של הרווח הראשון אפשר להשתמש בפעולה `IndexOf` המקבלת תו ומחזירה מספר שלם המייצג את מקומו במחרוזת:

```
placeOfLastLetter = sentence.IndexOf(' ') - 1;
```

טבלת הפעולות בסוף הפרק מתארת גרסה נוספת של פעולה זו המקבלת מחרוזת ומחזירה את מיקומה בתוך המחרוזת שעליה מופעלת הפעולה.

שימו ♥: יש תמיד לדאוג להתאמה בין טיפוסי הערכים שאנו משתמשים בהם ובין אלו המתוארים בכותרת הפעולה.

התוכנית המלאה

```
/*
קלט: משפט באנגלית
פלט: האות האחרונה במילה הראשונה ומספר המילים במשפט
*/
using System;
public class HowManyWords
{
    public static void Main ()
    {
        string sentence;
        int placeOfLastLetter;
        int numOfSpaces = 0;
        int numOfWords;
        Console.WriteLine("Enter a sentence, with exactly one " +
                                        " space between words: ");
        sentence = Console.ReadLine();
        // מניית הרווחים במשפט
        for (int i = 0 ; i < sentence.Length ; i++)
            if (sentence[i] == ' ')
                numOfSpaces++;
```

```
        numOfWords = numOfSpaces + 1;
        // מציאת מקומו של הרווח הראשון
        if (numOfWords == 1)
           placeOfLastLetter = sentence.Length - 1;
        else
           placeOfLastLetter = sentence.IndexOf(' ') - 1;
        Console.WriteLine("The last letter of the first word is: {0}"
                              ,sentence[placeOfLastLetter] );
        // כמה מילים במשפט
        Console.WriteLine("There are {0} words", numOfWords);
    }//Main
}// class HowManyWords
```

סוף פתרון בעיה 4

שאלה 9.7

פתחו וישמו אלגוריתם שהקלט שלו הוא תו ומחרוזת והפלט הוא הודעה אם התו נמצא במחרוזת או לא.

שאלה 9.8 (מתוך בגרות 1995)

פתחו וישמו אלגוריתם הקולט מחרוזת. האלגוריתם מציג כפלט כל תו במחרוזת פעמיים, פרט לתו '*' (כוכבית). גם אם התו כלול במחרוזת הוא לא מוצג כלל. הפלט מוצג בשורה אחת.
למשל עבור הקלט: ?AB*3B*? הפלט יהיה ??AABB33BB.

שאלה 9.9

פתחו וישמו אלגוריתם הקולט מחרוזת. האלגוריתם מציג כפלט את המחרוזת ללא אותיות זהות צמודות. למשל עבור הקלט: apple הפלט יהיה: aple, עבור הקלט: Yellow balloon הפלט יהיה: Yelow balon ועבור הקלט: abbba הפלט יהיה: aba.

שאלה 9.10

פתחו וישמו אלגוריתם שיקבל כקלט מילה באנגלית ויציג אותה פעמיים, פעם באותיות גדולות ופעם באותיות קטנות. למשל עבור הקלט Memory הפלט יהיה: MEMORY ומיד אחר-כך memory. חפשו פעולות מתאימות בטבלת הפעולות המופיעה בסוף הפרק.

9.4 הוראת השמה במחרוזות

עד כה ראינו כיצד לבצע פעולות קלט ופלט של מחרוזות, פעולת שרשור ופעולות שונות שמחזירות מידע על המחרוזות. בעיבוד של משתנים רגילים, אחת מהפעולות השימושיות ביותר היא הוראת השמה. בסעיף זה נדון בהשמה עבור מחרוזות, כלומר בהשמה של מחרוזת במשתנה שטיפוסו מחרוזת.

למעשה, כפי שכבר אמרנו, גם בעת קליטת מחרוזת אנו מבצעים השמה של המחרוזת הנקלטת, בעצם מסוג מחרוזת. למשל, בהוראה:

```
str = Console.ReadLine();
```

הפעולה `Console.ReadLine`, מחזירה הפניה לשטח הזיכרון החדש שהוקצה עבור מחרוזת הקלט. בעקבות ההשמה, המשתנה `str` מפנה אל אותו שטח זיכרון.

בסעיף זה נראה דוגמאות נוספות להשמה של מחרוזות.

בעיה 5

מטרת הבעיה הבאה: הצגת השמת מחרוזת למחרוזת והדגמה נוספת של השימוש בפעולות של המחלקה **string**.

נאמר שכתובת דואר אלקטרוני היא חוקית אם היא מקיימת את התנאים הבאים:

מתחילה באות אנגלית, מורכבת מרצף לא ריק של תווים ואחריו מגיע התו '@', אחריו שוב רצף לא ריק של תווים, אחריו התו '.' ולאחריו עוד רצף לא ריק של תווים. כתובת ישראלית מסתיימת במחרוזת ".il"

בקביעת חוקיות של כתובת דואר אין משמעות להבדל בין אותיות גדולות לקטנות.

למשל, המחרוזת d@ccv.hhh היא כתובת דואר אלקטרוני חוקית, וכך גם t@ייי.IL, שהיא כתובת ישראלית. לעומתן, המחרוזות הבאות אינן כתובות חוקיות: יח@ח.cc (לא מתחילה באות אנגלית), tr@.il (רצף התווים שבין התו '@' לבין התו '.' הוא ריק), rt@jjj (לא מכילה את התו '.') sda.asd@sad (התו '.' מופיע לפני התו '@').

פתחו אלגוריתם המקבל כקלט מחרוזת. ניתן להניח כי במחרוזת המתקבלת התו '.' לא מופיע יותר מפעם אחת וכך גם התו '@'. האלגוריתם מציג כפלט את המחרוזת שנקלטה, בצירוף הודעה המבהירה אם המחרוזת מהווה כתובת דואר אלקטרוני חוקית, ואם כן, אם זוהי כתובת ישראלית.

ישמו את האלגוריתם בשפת התכנות C#.

ניתוח הבעיה בעזרת דוגמאות

הכתובת tr@xxx.IL היא חוקית וישראלית. גם הכתובת tr@xxx.il היא חוקית וישראלית. אין משמעות לשימוש באותיות קטנות או גדולות – ההבדל ביניהן אינו משפיע על קביעת חוקיות המחרוזת וגם לא על סיווגה ככתובת ישראלית. אבל הפלט אינו לגמרי זהה בשני המקרים: אמנם בשני המקרים נציג הודעה כי המחרוזת חוקית וישראלית, אך במקרה הראשון נציג את המחרוזת tr@xxx.IL ובמקרה השני את המחרוזת tr@xxx.il.

כיוון שבשביל לבדוק את חוקיות המחרוזת וכדי לקבוע אם היא ישראלית או לא אין משמעות להבדל בין אותיות גדולות לקטנות, נוכל לפשט את הבדיקה אם ראשית נמיר את המחרוזת למחרוזת זהה המורכבת מאותיות גדולות או קטנות בלבד, ורק אז נבדוק. נחליט כי המחרוזת האחידה תהיה כולה אותיות קטנות. את מחרוזת הקלט המקורית נשמור כמו שהיא, כדי שנוכל להציגה כפלט.

קל לבדוק אם התו הראשון הוא אות אנגלית. ברגע שבדיקה זו הצליחה, כבר ברור שהמילה אינה מתחילה בתו '@', כלומר שהרצף שלפני התו '@' אינו ריק.

פירוק הבעיה לתת-משימות

1. קליטת מחרוזת
2. יצירת מחרוזת זהה לזו שנקלטה ובה כל האותיות הן אותיות קטנות בלבד
3. בדיקה שהתו הראשון הוא אות אנגלית
4. בדיקה שהתווים '@' ו-'.' מופיעים במחרוזת
5. בדיקה שהתו '.' מופיע אחרי התו '@', אך לא מיד אחריו ולא בסוף המחרוזת

6. עבור מחרוזת חוקית: בדיקה אם היא ישראלית
7. הצגה של המחרוזת ושל הודעה מלווה מתאימה

בחירת משתנים

בשאלה מתוארים כמה תנאים שמחרוזת חוקית צריכה לעמוד בהם. מספיק שאחד מהם לא מתקיים כדי לקבוע שהמחרוזת אינה חוקית. נוכל להיעזר במשתנה בוליאני: כל עוד לא מצאנו בעיה בכתובת הדואר האלקטרוני ערכו של המשתנה יהיה **true**. כאשר נמצא שגיאה באחד התנאים נציב בו **false**. ההודעה לפלט תוצג לפי ערכו של המשתנה.

בנוסף נזדקק לשתי מחרוזות: האחת לשמירת מחרוזת הקלט המקורית, והשנייה לשמירת המחרוזת האחידה.

כדי לבדוק את חוקיות המחרוזת נשתמש בשני משתנים שלמים שישמרו את מיקום התווים '.' ו-'@'.

לבסוף, נקדיש משתנה גם לאורך של המחרוזת שנקלוט. כך נוכל לחשב את האורך פעם אחת, ולהשתמש בערך המשתנה בכל הפעמים שנזדקק לאורך המחרוזת.

isLegal – משתנה בוליאני ("דגל"), שיעיד אם הכתובת היא חוקית או לא
str – מחרוזת לשמירת הכתובת הנקלטת
lowerStr – מחרוזת זהה למחרוזת הקלט ובה כל האותיות הן אותיות קטנות
atPlace – מספר שלם, לשמירת מיקומו של התו '@'
dotPlace – מספר שלם, לשמירת מיקומו של התו '.'
len – אורך מחרוזת הקלט

האלגוריתם

1. *אתחל את ערך* isLegal *ב-***true**
2. *קלוט כתובת ב-*str
3. *צור מחרוזת חדשה, זהה ל-*str *ובה כל האותיות הגדולות מוחלפות בקטנות, והשם אותה ב-*lowerStr
4. *אם האות הראשונה היא לא אות אנגלית*
   4.1. *שנה את ערך* isLegal *ל-***false**
5. *אם התו '.' או התו '@' אינם מופיעים במחרוזת*
   5.1. *שנה את ערך* isLegal *ל-***false**
6. *אחרת*
   6.1. *אם התו '.' מופיע לפני התו '@' או מיד אחריו או בסוף המחרוזת*
      6.1.1. *שנה את ערך* isLegal *ל-***false**
7. *אם ערכו של* isLegal *שווה ל-***true**
   7.1. *הצג כפלט הודעה כי הכתובת חוקית*
   7.2. *אם שלושת התווים האחרונים הם המחרוזת* ".il"
      7.2.1. *הצג הודעה כי הכתובת ישראלית*
   7.3. *אחרת*
      7.3.1. *הצג הודעה כי הכתובת אינה ישראלית*
8. *אחרת*
   8.1. *הצג כפלט הודעה כי הכתובת אינה חוקית*

יישום האלגוריתם

עלינו לבדוק אם התווים '.' ו-'@' מופיעים במחרוזת str, ואם כן – אם מיקומם תקין. לשם כך נשתמש בפעולה str.IndexOf המקבלת תו, ומחזירה את מיקומו במחרוזת str.

כדי ליצור ממחרוזת הקלט (השמורה ב-str) מחרוזת חדשה הזהה לה ובה כל האותיות הן קטנות, נשתמש בפעולה str.ToLower. כמו כל פעולה הפועלת על מחרוזות, גם הפעולה ToLower אינה משנה את המחרוזת שעליה היא מופעלת, אלא יוצרת מחרוזת חדשה. כמו כל פעולה שיוצרת מחרוזת חדשה, הפעולה ToLower מקצה מקום עבור המחרוזת החדשה.

אנו מעוניינים שהמשתנה lowerStr יפנה למחרוזת החדשה. לכן נשתמש במשפט השמה, ונשים את הערך המוחזר מהפעולה str.ToLower במשתנה lowerStr, כך:

```
lowerStr = str.ToLower();
```

בעקבות הוראה זאת העצם lowerStr מפנה אל שטח בזיכרון, ששמורה בו מחרוזת חדשה, הזהה למחרוזת הקלט ובה כל האותיות הן קטנות. במחרוזת המקורית השמורה ב-str לא חל כל שינוי.

לא היינו צריכים להקצות במפורש שטח זיכרון לעצם lowerStr, מפני שהפעולה ToLower הקצתה בעצמה שטח עבור המחרוזת. לאחר פעולת ההשמה העצם lowerStr מפנה לשטח זה.

התוכנית המלאה

```
/*
התוכנית מקבלת כקלט מחרוזת, מציגה אותה כפלט, ומציגה גם הודעה המבהירה
אם זו מחרוזת דואר אלקטרוני חוקית, ואם כן, אם היא כתובת ישראלית */
using System;
public class EmailAddress
{
    public static void Main ()
    {
        string str;            // מחרוזת הקלט
        string lowerStr;       // מחרוזת כמחרוזת הקלט, פרט לכך שאותיות
                               // ובה כל האותיות קטנות
        int atPlace;           // שומר מיקום התו '@'
        int dotPlace;          // שומר מיקום התו '.'
        bool isLegal = true;   // דגל חוקיות הכתובת
        int len;               // אורך המחרוזת שנקלטה
        // קלט מחרוזת ושמירת אורכה
        Console.Write("Enter a valid e-mail address: ");
        str = Console.ReadLine();
        len = str.Length;
        // יצירת המחרוזת החדשה ושמירתה
        lowerStr = str.ToLower();
        // בדיקה שהתו הראשון הוא אות לועזית
        if (!(lowerStr[0] >= 'a' && lowerStr[0] <= 'z'))
             isLegal = false;
        // מציאת מקום התווים '.' ו-'@'
        atPlace = lowerStr.IndexOf('@');
        dotPlace = lowerStr.IndexOf('.');
        if ((dotPlace == -1)||(atPlace == -1)) // אחד משני התווים חסר
             isLegal = false;
        else
```

```
            if ((dotPlace < atPlace) || (dotPlace == atPlace + 1)
                                        || (dotPlace == len - 1))
                // הסדר בין התווים שגוי או שהם מופיעים ברצף
                // או שהנקודה מופיעה בסוף
                isLegal = false;
        // הפלט
        if (isLegal)
        {
            Console.WriteLine("{0} is a legal Email address", str);
            // בדיקת שלושת התווים האחרונים במחרוזת, האם שווים ל-il.
            if (lowerStr.IndexOf(".il") == len - 3 )
                Console.WriteLine("Email address is Israeli");
            else
                Console.WriteLine("Email address is not Israeli");
        }
        else
            Console.WriteLine("{0} is not a valid Email address",
                                                                str);
    } // Main
} // EmailAddress
```

סוף פתרון בעיה 5

השמת מחרוזות נכתבת כהוראת השמה רגילה:

`s1 = s2;`

בעקבות ביצוע ההשמה, מפנה s1 אל אותו שטח זיכרון שאליו מפנה s2. לכן אין צורך לבצע קודם הקצאת זיכרון עבור s1.

שאלה 9.11

פתחו אלגוריתם המקבל כקלט רשימת מחרוזות המסתיימת במחרוזת "stop". פלט האלגוריתם יהיה המחרוזת הארוכה ביותר ברשימה. ישמו את האלגוריתם בשפת #C.

שאלה 9.12

פתחו אלגוריתם המקבל מחרוזת כקלט ומציג אותה בסדר הפוך. למשל עבור הקלט university הפלט יהיה: ytisrevinu. אין צורך ליצור מחרוזת הפוכה אלא רק להציג את התווים שלה בסדר הפוך. ישמו את האלגוריתם בשפת #C.

בעיה 6

מטרת הבעיה ופתרונה: הדגמת בניית מחרוזת בשלבים.

פתחו אלגוריתם אשר הקלט שלו הוא מחרוזת. הפלט יהיה מחרוזת חדשה, הזהה למחרוזת המקורית, פרט לכך שמופיעה בה האות c בכל מקום שהופיעה האות b או B במחרוזת המקורית. למשל עבור המחרוזת "abcd" הפלט יהיה המחרוזת "accd".

שימו ♥: פרט לשינוי המתואר, המחרוזת החדשה צריכה להיות זהה למקורית. לא ניתן להפוך בה אותיות גדולות לקטנות או להפך.

פירוק הבעיה לתת-משימות

1. קליטת מחרוזת
2. בניית המחרוזת החדשה
3. הדפסת המחרוזת החדשה

את תת-משימה 2 נוכל לפרק באופן הבא:

2.1. יצירת עצם מסוג מחרוזת ואתחולו במחרוזת ריקה
2.2. בנייה הדרגתית של המחרוזת החדשה, תו אחר תו

את תת-משימה 2.2 נוכל ליישם בלולאת **for** ומספר הפעמים לביצועה נקבע לפי אורך המחרוזת.

בחירת משתנים

str – המחרוזת שמתקבלת כקלט
newStr – המחרוזת החדשה

יישום האלגוריתם

אתחול העצם `newStr` במחרוזת ריקה (תת-משימה 2.1) אפשר לבצע יחד עם הצהרת המחרוזת:

```
string newStr = "";
```

את תת-משימה 2.2 נוכל ליישם בלולאת **for**, שמספר הפעמים לביצועה נקבע לפי אורך המחרוזת. בכל פעם נבדוק את התו הבא במחרוזת בעזרת הסימנים `[]`. ונבנה את המחרוזת החדשה באמצעות פעולת השרשור, למשל כך:

```
newStr = newStr + 'c';
```

שימו ♥: פעולת השרשור היא פעולה שמחזירה מחרוזת. כמו כל פעולה שמחזירה מחרוזת, פעולת השרשור אינה משנה את המחרוזות שהיא פועלת עליהן, אלא יוצרת מחרוזת חדשה (כמובן אחרי שהקצתה מקום עבורה), המתקבלת משרשור המחרוזות המקוריות.

נניח למשל שב-`newStr` נמצאת המחרוזת `"ad"`. מה קורה בעת ביצוע הוראת ההשמה שלעיל?

פעולת השרשור מקבלת שתי מחרוזות (במקרה זה את `newStr` ואת המחרוזת `"c"`, אחרי המרת התו `'c'` למחרוזת) ויוצרת תוך הקצאת מקום מתאים בזיכרון מחרוזת חדשה שמכילה את שרשור שתי המחרוזות, כלומר את המחרוזת `"adc"`. היא מחזירה את המחרוזת החדשה.

בפעולת ההשמה המחרוזת חדשה מושמת ב-`newStr`. כלומר במקום ש-`newStr` יַפנה אל המחרוזת המקורית (`"ad"`), הוא מפנה כעת אל השטח החדש בזיכרון. מה קרה למחרוזת המקורית? ניתן לומר שהיא הלכה לאיבוד, מכיוון שאין הפניות אליה. בכך פעולת השמה של מחרוזות אינה שונה מפעולת השמה רגילה של משתנים: גם כאשר אנו מבצעים השמה כגון `x = y` למשתנים `x` ו-`y` שהם משתנים רגילים, אז ערכו המקורי של `x` אובד והערך החדש של `y` מחליף אותו.

כאשר נשרשר למחרוזת תו נוסף, אותו תהליך יחזור על עצמו: הקצאת שטח למחרוזת חדשה שתהיה השרשור של התו החדש למחרוזת המקורית, והשמה הגורמת ל-`newStr` להפנות אל השטח בזיכרון של המחרוזת החדשה.

אם כך בבנייה הדרגתית של מחרוזת כפי שתואר לעיל, לא יהיה מדויק לומר ש**אותה** מחרוזת גדלה כל פעם בתו נוסף. למעשה, נוצרת סדרה של מחרוזות: בכל פעם נוצרת מחרוזת חדשה, ארוכה בתו אחד, והיא תופסת את מקומה של המחרוזת הקודמת.

התוכנית המלאה

```
/*
קלט: מחרוזת
פלט: מחרוזת זהה למקורית, ובמקום כל 'B' או 'b' מופיע 'c'
*/
using System;
public class ReplaceCForB
{
    public static void Main ()
    {
        string str;           // מחרוזת הקלט
        string newStr = "";   // המחרוזת החדשה כעצם
                              //המאותחל במחרוזת ריקה
        // קלט
        Console.Write("Enter a string: ");
        str = Console.ReadLine();
        // בניית המחרוזת החדשה
        for(int i = 0; i < str.Length; i++)  // סריקת מחרוזת הקלט
                                             // תו אחר תו
        {
            if ((str[i] == 'b') || (str[i] == 'B'))
                newStr = newStr + 'c'; // החלפת 'B' או 'b' ב-'c'
            else
                newStr = newStr + str[i]; // העתקת התו המקורי
        } // for
        // פלט
        Console.WriteLine("The new string is: {0}", newStr);
    } // Main
} // class ReplaceCForB
```

סוף פתרון בעיה 6

כאשר אנו נדרשים **לבנות מחרוזת חדשה בשלבים, תו אחרי תו** ניתן לעשות זאת באופן הבא: ליצור עצם שיהיה מחרוזת חדשה ולאתחל אותו במחרוזת ריקה (""); באמצעות הוראה לביצוע-חוזר, לבצע בכל פעם שרשור של התו החדש למחרוזת הקיימת, והחלפת המחרוזת הקיימת במחרוזת החדשה שהתקבלה מפעולת השרשור.

שאלה 9.13

פתחו וישמו אלגוריתם שהקלט שלו הוא מחרוזת המהווה משפט משובש: במקום כל רווח מופיע הסימן $. האלגוריתם מייצר מחרוזת **חדשה** ובה המשפט התקני, ומציג אותו כפלט. ישמו את האלגוריתם בשפת C#.

שאלה 9.14

אורי ונעמי המציאו שפה מוצפנת. המשפטים הניתנים להצפנה כוללים מילים ורווחים בלבד, בהינתן שכל מילה נכתבת רק באותיות אנגליות קטנות.

ההצפנה מתבצעת באופן הבא: כל רווח מוחלף בסימן '@', ולאחר כל אות מופיעה האות האנגלית הגדולה המתאימה לה.

למשל, המשפט good morning מוצפן כך gGoOoOdD@mMoOrRnNiInNgG.

פתחו אלגוריתם המקבל משפט כקלט ומציג את המשפט המוצפן המתאים לו כפלט. ישמו את האלגוריתם בשפת C#.

הדרכה: צרו מחרוזת חדשה המורכבת מאותיות גדולות, והשתמשו בה ובמחרוזת הקלט לצורך בניית המחרוזת המוצפנת.

שאלה 9.15

פתחו אלגוריתם שמקבל כקלט מחרוזת, מייצר מחרוזת חדשה ובה סדר התווים הפוך למחרוזת המקורית, ומציג את המחרוזת החדשה כפלט. בנוסף האלגוריתם בודק אם שתי המחרוזות (המקורית וההפוכה) שוות זו לזו. במילים אחרות האלגוריתם בודק אם המחרוזת המקורית היא פלינדרום. האלגוריתם מציג כפלט הודעה מתאימה לתוצאת הבדיקה.

למשל: עבור הקלט: dog יהיה הפלט: god – not a palindrome.

עבור הקלט: aba יהיה הפלט: aba – a palindrome.

ישמו את האלגוריתם בשפת C#.

שאלה 9.16

תלמידי הכיתה התעניינו לדעת למי יש סבתא בשם הארוך ביותר. פתחו אלגוריתם אשר יקבל כקלט את מספר התלמידים בכיתה, ולאחר מכן רשימה של שמות הסבתות של תלמידי הכיתה כמחרוזות. אורך הרשימה כמספר תלמידי הכיתה. פלט האלגוריתם יהיה השם הארוך ביותר. ישמו את האלגוריתם בשפת C#.

שאלה 9.17

כתובת אתר אינטרנט של חברה מסחרית בינלאומית בנויה בדרך כלל מ-3 חלקים המופרדים בנקודות:

www.**שם החברה**.com

פתחו אלגוריתם המקבל כקלט כתובת של אתר של חברה מסחרית בינלאומית, במבנה שתואר לעיל, ומציג כפלט את שם החברה בלבד. ישמו את האלגוריתם בשפת C#.

הדרכה: השתמשו בפעולה `Substring` שבטבלת הפעולות הנמצאת בסוף הפרק.

שאלה 9.18

כתובת אתר אינטרנט של חברה מסחרית שאינה בינלאומית בנויה בדרך כלל מ-3 חלקים המופרדים בנקודות:

www.**שם החברה**.**סיומת המדינה**

פתחו אלגוריתם המקבל כקלט רשימת כתובות של אתרי חברות מסחריות כאלה (כל אחת מהן במבנה שתואר לעיל). הרשימה תסתיים במחרוזת "end". האלגוריתם יציג כפלט עבור כל חברה את שמה ואת סיומת המדינה שלה. ישמו את האלגוריתם בשפת C#.

שאלה 9.19

יעל ועומרי המציאו משחק. כל אחד בתורו רושם משפט המתחיל במילה האחרונה של המשפט מהתור הקודם. המשפט הראשון במשחק יתחיל במילה "start". למשל:

start the game

game is one of the best ways to kill time

time is money

פתחו אלגוריתם המסייע למשחקים: האלגוריתם מקבל כקלט את מספר התורות המבוקש. בתחילת כל תור הוא מציג את המילה שצריך להתחיל בה המשפט הבא, ולאחר מכן הוא מקבל כקלט את המשפט החדש. המשחק מסתיים כאשר מספר התורות הסתיים, או כאשר המשפט שבחר אחד השחקנים מתחיל במילה שגויה. ישמו את האלגוריתם בשפת C#.

סיכום

בפרק זה הכרנו את המחלקה **string** ולמדנו כיצד לעבד מחרוזות.

בשפת C# מחרוזת אינה טיפוס פשוט, כמו **int** או **char**. הטיפוס החדש מוגדר בשפה כ**מחלקה בשם string**, ומחרוזות הן **עצמים** של מחלקה זו.

למעשה כבר הכרנו עצמים כאשר השתמשנו במחלקה Random ליצירת מספרים אקראיים. עצמים הם משתנים של הטיפוס החדש ואפשר להפעיל עליהם פעולות המוגדרות במחלקה. אופן השימוש בעצמים שונה מהשימוש במשתנים מהטיפוסים הפשוטים המוכרים לנו.

הצהרה על עצם ממחלקה מסוימת דומה להצהרה על משתנה מטיפוס סטנדרטי (כגון **int** או **char**). למשל:

```
string str;
```

עבור משתנים רגילים, הקצאת מקום בזיכרון נעשית אוטומטית עם ההצהרה עליהם ואילו עבור עצמים הקצאת הזיכרון והאתחול מתבצעים באמצעות הפעולה **new**. מכיוון שהשימוש במחרוזות נפוץ כל כך, שפת C# מאפשרת ליצור ולאתחל מחרוזת בצורה ישירה ללא שימוש בפעולה **new**.

למשל, ההוראה הבאה מקצה מקום עבור `str` ומאתחלת אותו כך שישמור את המחרוזת `"ab"` :

```
str = "ab";
```

מיקום התווים במחרוזת מתחיל ב-0. בכל מחרוזת, אם נסמן את אורך המחרוזת ב-len, אז התו האחרון במחרוזת נמצא במיקום len-1, והתו הראשון נמצא במיקום ה-0.

פנייה אל תו בתוך מחרוזת נעשית בסוגריים מרובעים. למשל ערכו של הביטוי `str[k]` הוא התו הנמצא במקום `k` במחרוזת `str`. ניתן לקרוא את ערכו של תו במחרוזת, אך לא לשנות אותו.

לעצם יכולות להיות תכונות (נרחיב בנושא בפרק 11). למשל עבור עצם של המחלקה **string**, כלומר עבור מחרוזת, מוגדרת **התכונה Length** המייצגת את אורכה של המחרוזת. כדי לקרוא את ערכה של התכונה משתמשים **בסימון הנקודה**: למשל הביטוי `str.Length` מחזיר את מספר התווים במחרוזת `str`.

ערכה של התכונה `Length` אינו ניתן לשינוי.

במחלקה **string** מוגדרות **פעולות שאפשר לבצע על מחרוזות**, כלומר על עצמים של המחלקה.

לכל פעולה מוגדרים סוגי הערכים (הפרמטרים) שהיא מצפה לקבל, וכן מוגדר טיפוס הערך המוחזר ממנה. חשוב לוודא התאמה של טיפוסי הפרמטרים המועברים לפעולה לאלה שהיא מצפה לקבל, ושל טיפוס הערך המוחזר ממנה לביטוי שהוא משולב בו. למשל, אם אנו מבצעים השמה של הערך המוחזר מפעולה במשתנה, יש לוודא התאמה של טיפוס המשתנה לטיפוס הערך המוחזר. יש פעולות שלא מצפות לקבל פרמטרים ופעולות שלא מחזירות ערך.

כדי לציין ביצוע פעולה של עצם מהמחלקה משתמשים **בסימון הנקודה**: ראשית נכתוב את שם העצם, אחריו נקודה, לאחר מכן את שם הפעולה לביצוע ומיד אחר כך נפרט בסוגריים את הערכים המועברים לפעולה. למשל:

```
if (str.Equals(otherStr))
      place = str.IndexOf('.');
```

עבור עצמים של המחלקה **string** מוגדרות פעולות רבות, המשמשות אותנו ביישום אלגוריתמים הקשורים למחרוזות. למחלקה זו פעולות רבות נוספות פרט לאלו שהוצגו בפרק. תוכלו למצוא הרחבה על כל אחת מהפעולות בקישור הזה:

http://msdn2.microsoft.com/en-us/library/system.string_members.aspx

ביחידה זו נשתמש רק בפעולות שהוצגו בפרק זה.

הפעולות שמחזירות מחרוזת, אינן משנות את המחרוזת שהופעלו עליה. הן יוצרות מחרוזת חדשה ומקצות שטח עבורה.

לביצוע **קלט של מחרוזת** השתמשנו בפעולה `Console.ReadLine`. גם פעולה זו מקצה שטח זיכרון עבור המחרוזת שנקלטה.

בעקבות ביצוע **השמה של מחרוזת** אל מחרוזת, `s1 = s2`, מפנה `s1` אל אותו שטח זיכרון שאליו מפנה `s2`. לכן אין צורך לבצע קודם הקצאת זיכרון עבור `s1`.

בנוסף לפעולות המוגדרות במחלקה **string**, למדנו בפרק זה על **פעולת השרשור**. פעולה זו מקבלת שתי מחרוזות ויוצרת מחרוזת חדשה והיא השרשור של שתי המחרוזות. זוהי פעולה שימושית מאוד לצורך הדפסה. פעולת השרשור אינה פעולה של המחלקה **string** ולכן בכתיבתה אין שימוש בסימון הנקודה. סימן פעולת השרשור הוא הסימן `+`.

בפרקים הבאים נלמד להגדיר מחלקות בעצמנו, ולא רק להשתמש במחלקות קיימות.

רשימת פעולות על מחרוזות

הפעולה	תיאור הפעולה	טיפוס הערך המוחזר	דוגמאות – הפעולה	דוגמאות – הערך המוחזר
`IndexOf(string s)`	פעולה המקבלת מחרוזת או תו, ומחפשת בתוך המחרוזת שעליה מופעלת הפעולה את המיקום הראשון שבו מופיעה המחרוזת או התו שהתקבלו. הפעולה תחזיר את המיקום. היא תחזיר את הערך 1-, אם החיפוש נכשל.	שלם	`s1.IndexOf("pl")` כאשר ב-`s1` נמצאת המחרוזת `"people"`	3
`IndexOf(char c)`			`s1.IndexOf('o')` כאשר ב-`s1` נמצאת המחרוזת `"people"`	2

Equals(**string** s)	פעולה המקבלת מחרוזת, ומחזירה **true** אם המחרוזת שעליה הופעלה הפעולה והמחרוזת שהתקבלה שוות זו לזו בדיוק. אחרת, היא מחזירה **false**. פעולה זו זהה לפעולת ההשוואה ==	בוליאני	s1.Equals(s2) כאשר ב-s1 נמצאת המחרוזת "love" וב-s2 נמצאת המחרוזת "Love"	**false**
			s1.Equals(s2) כאשר ב-s1 נמצאת המחרוזת "love" וב-s2 נמצאת המחרוזת "love"	**true**
CompareTo(**string** s)	פעולה המקבלת מחרוזת, ומשווה אותה למחרוזת שעליה הופעלה הפעולה. אם הן שוות זו לזו מוחזר הערך אפס. אם המחרוזת שעליה מופעלת הפעולה **קודמת** למחרוזת שהתקבלה בסדר **מילוני**, יוחזר מספר שלם שלילי. אם המחרוזת שעליה מופעלת הפעולה, מופיעה **אחרי** המחרוזת שהתקבלה בסדר מילוני, יוחזר מספר שלם חיובי.	שלם	s1.CompareTo(s2) כאשר ב-s1 נמצאת המחרוזת "aa" וב-s2 נמצאת המחרוזת "ab"	מספר שלילי כלשהו
ToLower()	פעולה שיוצרת מחרוזת זהה למחרוזת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות קטנות. הפעולה מחזירה את המחרוזת החדשה.	מחרוזת	s1.ToLower() כאשר ב-s1 נמצאת המחרוזת "Peace"	המחרוזת החדשה "peace"
ToUpper()	פעולה שיוצרת מחרוזת זהה למחרוזת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות גדולות. הפעולה מחזירה את המחרוזת החדשה.	מחרוזת	s1.ToUpper() כאשר ב-s1 נמצאת המחרוזת "Peace"	המחרוזת החדשה "PEACE"
Substring(**int** k)	פעולה שיוצרת מחרוזת זהה לתת-מחרוזת המתחילה מהמקום ה-k של המחרוזת שעליה הפעולה מופעלת ועד סופה. הפעולה מחזירה את המחרוזת החדשה.	מחרוזת	s1.Substring(4) כאשר ב-s1 נמצאת המחרוזת "GoodBye"	המחרוזת החדשה "Bye"

Substring(**int** k, **int** s)	פעולה שיוצרת מחרוזת זהה לתת-מחרוזת המתחילה מהמקום ה-k של המחרוזת עליה היא מופעלת ואורכה הוא s. הפעולה מחזירה את המחרוזת החדשה. שימו לב שעל ערכו של k+s להיות קטן או שווה לאורך המחרוזת.	מחרוזת	s1.Substring(4,6) כאשר ב-s1 נמצאת המחרוזת "GoodBye Israel"	המחרוזת החדשה "Bye Is"

פרק 10 – מערכים

האלגוריתמים שפיתחנו לפתרון בעיות שונות בפרקים הקודמים היו שונים ומגוונים. הם היו שונים זה מזה בפרט בכמות המידע שנקלט בהם, כלומר בגודל הקלט. אבל בכל האלגוריתמים שהצגנו עד כה, גם כאשר כמות המידע הנקלט הייתה גדולה, הרי כמות המידע שהיה צריך **לשמור** או **לזכור** במהלך ביצוע האלגוריתם הייתה קטנה. למשל כאשר רצינו לחשב ממוצע של 100 מספרי קלט, שמרנו אך ורק את סכומם המצטבר ואת נתון הקלט התורן, ולא את כל 100 נתוני הקלט. בכל האלגוריתמים שפיתחנו עד כה השתמשנו במספר משתנים מצומצם, ולכל משתנה הוגדר תפקיד ייחודי משלו.

בפרק זה יוצגו בעיות אשר לצורך פתרונן יש לשמור מספר גדול של נתונים שיש קשר ביניהם: הם בעלי משמעות דומה וניתנים לתיאור כאוסף סדור של איברים מאותו טיפוס. אוסף סדור כזה של איברים נקרא "מערך".

10.1 מערך ואיברי מערך

בעיה 1

מטרת הבעיה ופתרונה: הצגת שימוש במערך לפתרון בעיה אלגוריתמית.

פתחו אלגוריתם המקבל כקלט את זמני ההקפה של כל אחד מארבעים משתתפי מרוץ הקרטינג. האלגוריתם מציג כפלט את מספר המשתתפים שביצעו הקפה בזמן נמוך מהזמן הממוצע של כלל המשתתפים. ישמו את האלגוריתם בשפת C#.

ניתוח הבעיה בעזרת דוגמאות

שאלה 10.1

מהו הפלט כאשר נתונים 40 זמני הקפה, עשרים מהם שווים ל-4.50, עשרה שווים ל-4.20, ועשרה שווים ל-4.60?

פירוק הבעיה לתת-משימות

בפרק 7 כבר ראינו כיצד לחשב ממוצע של רשימת ערכי קלט:

1. קליטת נתוני הקלט וצבירתם
2. חלוקת הסכום המצטבר במספר ערכי הקלט

? במהלך צבירת הנתונים לצורך חישוב הממוצע נוסף לצובר ערכו של כל נתון שנקלט, אך הנתון איננו נשמר. לאחר קליטת כל נתוני הקלט ניתן לחשב את הממוצע, אך ערכי הקלט עצמם אינם שמורים. אבל כדי למנות את מספר הנתונים שמתחת לממוצע יש **לשמור** את ערכי הקלט עד לאחר חישוב הממוצע, משום שרק לאחר חישוב הממוצע, אפשר להשוות כל אחד מערכי הקלט לממוצע. כיצד נרחיב את התת-משימות המתוארות כך שיתייחסו גם למציאת ערכי הקלט הקטנים מהממוצע?

1. קליטת נתוני הקלט, **שמירתם** וצבירתם
2. חלוקת הסכום המצטבר במספר ערכי הקלט
3. השוואת כל ערך קלט לממוצע, ומנייתו אם הוא קטן מהממוצע

בחירת משתנים

יש 40 נתוני קלט, ועלינו לשמור כל אחד מהם בנפרד. האם נצהיר על 40 משתנים, כל אחד בנפרד?

מכיוון שלכל אחד מנתוני הקלט אפיון דומה, אפשר לקשרם יחד. ניתן להתייחס אל נתוני הקלט כאל סדרת ערכים בת 40 איברים דומים, ולפיכך לשמור אותם בסדרת משתנים. הערך שנקלט ראשון יישמר במשתנה הראשון בסדרה, הערך שנקלט שני יישמר במשתנה השני בסדרה וכך הלאה. הקישור בין המשתנים יתבצע על ידי מתן שם לסדרה ופנייה לכל אחד מהמשתנים לפי מיקומו הסידורי בסדרה. סדרה כזאת של משתנים הקשורים זה לזה נקראת **מערך**.

מערך (array) הוא אוסף סדור של איברים מאותו טיפוס. ניתן להתייחס לכל אחד מאיברי המערך כמו למשתנה לכל דבר. כלומר ניתן לשמור בו ערכים ולקרוא את הערכים השמורים בו. מיקומו הסידורי של איבר במערך מצוין ב**מציין** (index).

אם כך בבעיה זו אנו זקוקים למערך בן ארבעים איברים, איבר אחד עבור כל אחד מארבעים זמני ההקפה הנתונים. הטיפוס של כל איבר יהיה ממשי. למערך לשמירת זמני ההקפה ניתן את השם scores.

לכן זוהי רשימת המשתנים שנזדקק לה:

scores – מערך של 40 איברים ממשיים, לשמירת כל זמני ההקפה
sumOfScores – מטיפוס ממשי, ישמור את סכום זמני ההקפות הנתונים בקלט
averageScore – ממוצע זמני ההקפה, מטיפוס ממשי
belowAverageCounter – למניית מספר הזמנים שערכם הוא מתחת לממוצע, מטיפוס שלם

בשפת C# ניתן להגדיר מערך שאיבריו הם מכל טיפוס נתונים שהוא, למשל, מערך שאיבריו הם מטיפוס שלם, או מערך שאיבריו הם מטיפוס תו. לצורך פתרון בעיה זו אנו נדרשים להגדיר מערך מטיפוס ממשי, כלומר שאיבריו הם מטיפוס ממשי. ההצהרה על מערך של מספרים ממשיים נעשית באופן הבא:

```
double[] scores;
```

שימו ♥: ההצהרה דומה להצהרה על משתנה מטיפוס ממשי, ורק תוספת הסוגריים המרובעים (`[]`) מבהירה כי אין הכוונה כאן למשתנה יחיד מטיפוס ממשי אלא לעצם שהוא מערך שאיבריו הם מטיפוס ממשי.

אך ההצהרה אינה מספיקה. עלינו להקצות מקום בזיכרון עבור אוסף האיברים שבמערך באמצעות ההוראה **new**. כמובן בעת ההקצאה עלינו לציין את מספר האיברים במערך, וכך נקבע את גודל שטח הזיכרון שיש להקצות.

אם כך, הגדרת עצם שהוא מערך מטיפוס ממשי, והקצאת מקום בזיכרון עבורו (בציון מספר האיברים שהוא אמור להכיל) תיעשה כך:

```
double[] scores = new double[40];
```

כזכור, הפעולה **new** מחזירה הפניה לשטח הזיכרון שהוקצה. לכן למעשה `scores` מפנה כעת אל השטח שהוקצה עבורו בזיכרון.

נשתמש בקבוע על מנת להגדיר את מספר האיברים, וכך תיראה ההגדרה:

```
const int NUM_OF_RUNNERS = 40;
double[] scores = new double[NUM_OF_RUNNERS];
```

יש להדגיש כי בשפת C#, כאשר מוקצה שטח זיכרון עבור מערך של איברים מטיפוס פשוט, מתבצע גם אתחול אוטומטי של כל איבריו. איברי מערך מספריים (שלמים או ממשיים) מאותחלים ב-0, איברי מערך בוליאני מאותחלים ב-**false**, ואיברי מערך תווי מאותחלים בתו מיוחד שנקרא תו ריק.

בשפת C# האיבר הראשון במערך הוא במיקום 0 מתחילת המערך ולכן פנייה לאיבר הראשון תיעשה באמצעות המציין 0, כך: scores[0]. האיבר השני נמצא במיקום 1 מתחילת המערך ולכן פנייה אליו תיעשה באמצעות המציין 1, כך: scores[1]. האיבר האחרון ברשימה, איבר מספר 40, נמצא במיקום 39 מתחילת המערך ונפנה אליו בכתיבת scores[39]. באופן כללי, ציון איברי מערך בשפת C# מתחיל תמיד ב-0, ופנייה לאיבר הנמצא במיקום i מתחילת מערך בשם anArray נכתבת כך: anArray[i]. איבר זה הוא האיבר ה-i+1 ברשימת האיברים.

נמחיש את המערך scores בעזרת האיור הבא:

scores[0]	scores[1]		scores[19]		scores[39]
		…		…	

האלגוריתם

1. *אתחל את* sumOfScores *ב-0*
2. *אתחל את* belowAverageCounter *ב-0*
3. ***עבור כל*** i ***שלם מ-0 עד מספר הערכים הנקלטים פחות 1 בצע:***
   3.1. *קלוט את זמן ההקפה הבא והשם באיבר ה-*i *במערך* scores
   3.2. *הוסף לערך השמור ב-*sumOfScores *את זמן ההקפה הנקלט*
4. *חלק את הערך השמור ב-*sumOfScores *במספר ערכי הקלט והשם ב-* averageScore
5. ***עבור כל*** i ***שלם מ-0 עד מספר הערכים הנקלטים פחות 1 בצע:***
   5.1. ***אם*** *ערכו של האיבר ה-*i *במערך* scores *גדול מ-*averageScore
      5.1.1. *העלה את ערכו של* belowAverageCounter *ב-1*
6. *הצג כפלט את ערכו של* belowAverageCounter

יישום האלגוריתם

לכל עצם שהוא מערך מוגדרת בשפה תכונה בשם `Length` השומרת את גודל המערך, כלומר את מספר האיברים שהוא מכיל. לתכונה זו נוכל לגשת באמצעות סימון הנקודה, בדומה לאופן שבו הפעלנו פעולות על עצם. למשל `scores.Length` היא תכונת האורך של המערך `scores`, וערכה שווה ל-40.

תכונת האורך של מערך היא קבועה ואינה ניתנת לשינוי. כלומר נוכל לשלב אותה בתוך ביטויים שונים בתוכנית, למשל `x = scores.Length + 1`, אך לא נוכל להציבה בצד שמאל של הוראת השמה. כך למשל, הוראה כמו `scores.Length = 3` היא שגויה. אם כך `scores.Length` הוא בעצם קבוע לכל דבר (בדומה לקבוע `NUM_OF_RUNNERS`), אך הוא מקושר לעצם `scores`, וניתן לגשת אליו רק דרך העצם `scores` כפי שמביע סימון הנקודה.

שימו ♥: גם כדי להשתמש בפעולה על עצם וגם כדי לגשת לתכונה שלו אנו משתמשים בסימון הנקודה, אך הפנייה לפעולה תלווה תמיד בסוגריים (אולי ריקים, אם הפעולה אינה מצפה לקבל פרמטרים), ואילו פנייה לתכונה, בדומה לפנייה למשתנה רגיל, היא ללא סוגריים.

באלגוריתם לפתרון בעיה 1 כללנו הוראה לביצוע-חוזר מספר פעמים ידוע מראש, למעבר על כל איברי המערך בזה אחר זה. נוכל להשתמש בתכונת האורך של המערך הנסרק כדי לשלוט במספר הסיבובים בלולאה. כלומר נוכל לקבוע מראש את מספר הסיבובים בלולאה ל-`scores.Length`.

מאחר שהתכונה `Length` שומרת את מספר האיברים במערך, ניתן גם לומר שאיברי המערך נמצאים בו מהמקום 0 ועד המקום `scores.Length-1`. אם כך, נאתחל את משתנה הבקרה של הלולאה ב-0, והלולאה תסתיים כאשר ערכו יגיע ל-`scores.Length-1` (אפשר כמובן גם לקבוע את ערך הסיום של משתנה הבקרה ל-`NUM_OF_RUNNERS-1`).

התוכנית המלאה

```
/*
קלט: זמני ההקפה של ארבעים משתתפי מרוץ קרטינג
פלט: מספר המשתתפים שביצעו הקפה בזמן נמוך מהממוצע
הקבוצתי
*/
using System;
public class BelowAverage
{
    public static void Main()
    {
        // הגדרת קבוע
        const int NUM_OF_RUNNERS = 40;
        // הגדרת משתנים
        double[] scores = new double[NUM_OF_RUNNERS];
                                       // מערך זמני ההקפה
        double sumOfScores = 0;       // צובר זמני ההקפה
        double averageScore;          // ממוצע זמני ההקפה
        int belowAverageCounter = 0;  //  מונה
        // קלט וצבירה
1.      for (int i = 0; i < scores.Length; i++)
        {
1.1.        Console.Write("Enter score: ");
1.2.        scores[i] = int.Parse(Console.ReadLine());
1.3.        sumOfScores = sumOfScores + scores[i];
        } // for
        // חישוב ממוצע
2.      averageScore = sumOfScores/scores.Length;
      // מניית הנמוכים מהממוצע
3.      for (int i = 0; i < scores.Length; i++)
3.1.        if (scores[i] < averageScore)
3.1.1.          belowAverageCounter++;
        // פלט
4.      Console.WriteLine("{0} participants are below average",
                                                 belowAverageCounter);
    } // Main
} //class BelowAverage
```

מעקב

נעקוב באופן חלקי אחר מהלך ביצוע התוכנית עבור הקלט 4.20 ... 4.10 4.50
נניח שסכום זמני ההקפות הוא 168.8.

משפט לביצוע	i	Scores [0]	…	Scores [39]	sumOf Scores	scores[i] < average Score	average Score	below Average Counter	פלט
1	0	?		?	0		?	0	
1.1	0	?		?	0		?	0	Enter score:
1.2	0	4.50		?	0		?	0	
1.3	0	4.50		?	4.50		?	0	
1	1	4.50		?	4.50		?	0	
1.1	1	4.50		?	4.50		?	0	Enter score:
1.2	1	4.50		?	8.60		?	0	
1.3	1	4.50		?	8.60		?	0	
. . .	. . .	. . .	. . .	. . .	. . .		. . .	. . .	
1	39	4.50		?	164.6		?	0	
1.1	39	4.50		?	164.6		?	0	Enter score:
1.2	39	4.50		4.20	164.6		?	0	
1.3	39	4.50		4.20	168.8		?	0	
1	40	4.50		4.20	168.8		?	0	
2	?	4.50		4.20	168.8		4.22	0	
3	0	4.50		4.20	168.8		4.22	0	
3.1	0	4.50		4.20	168.8	**false**	4.22	0	
3	1	4.50		4.20	168.8		4.22	0	
3.1	1	4.50		4.20	168.8	**true**	4.22	0	
3.1.1	1	4.50		4.20	168.8		4.22	1	
. . .	. . .	. . .	. . .	. . .	. . .		. . .	. . .	
3	39	4.50		4.20	168.8		4.22	1	
3.1	39	4.50		4.20	168.8	**true**	4.22	1	
3.1.1	39	4.50		4.20	168.8		4.22	18	
3	40	4.50		4.20	168.8		4.22	18	
4	?	4.50		4.20	168.8		4.22	18	18 participants…

שימו ♥ : המעקב אחר הערכים השמורים באיבר של המערך זהה למעקב אחר הערכים השמורים במשתנה.

סוף פתרון בעיה 1

בפתרון הבעיה הכרנו שימוש במערך עבור שמירת נתונים שניתן להתייחס אליהם כאל אוסף סדור של איברים מאותו טיפוס. ניתן להתייחס אל איבר במערך כאל משתנה. כלומר, ניתן לשמור בו

ערכים ולקרוא את הערכים השמורים בו. מיקומו הסידורי של איבר במערך מצוין **במציין** (index).

באלגוריתם לפתרון בעיה זו הכרנו דרך נוחה לסרוק מערך ולבצע עיבוד על כל איבריו באמצעות הוראה לביצוע-חוזר, שמספר הסיבובים בה ידוע מראש ושווה למספר האיברים במערך (במילים אחרות, לאורכו של המערך).

נסכם את המושגים הבסיסיים שהכרנו בפתרון בעיה 1, הנוגעים למערכים ולעבודה עמם בשפת C#:

מערך בשפת C# הוא עצם המוגדר בשפה.

משום שזהו עצם, אחרי ההצהרה על מערך צריך לבצע עבורו **הקצאת מקום בזיכרון**, באמצעות ההוראה **new**. בעקבות ההקצאה שם המערך מפנה אל שטח הזיכרון שהוקצה עבורו.

בשפת C#, מיד אחרי הקצאת שטח זיכרון עבור מערך של איברים מטיפוס פשוט, מתבצע גם **אתחול אוטומטי** של כל איבריו. איברי מערך מספריים (שלמים או ממשיים) יאותחלו ב-0, איברי מערך בוליאני יאותחלו ב-**false**, ואיברי מערך תווי יאותחלו בתו הריק.

ציון איברי מערך מתחיל תמיד ב-0. **פנייה לאיבר** הנמצא במיקום i מתחילת מערך בשם anArray נכתבת כך: anArray[i]. איבר זה הוא האיבר ה-i+1 ברשימת האיברים.

לכל עצם שהוא מערך יש **תכונה השומרת את אורכו** (Length). ניתן להשתמש בה כדי לדעת את מספר האיברים במערך אך לא ניתן לשנותה. נוח להשתמש בתכונה זו כדי לקבוע את מספר הפעמים לביצוע בלולאות שסורקות את כל איברי המערך.

פנייה לתכונת האורך נעשית באמצעות סימון הנקודה, למשל כך: `anArray.Length`.

בתוכנית שבפתרון בעיה 1 הצהרנו על המערך `scores` והקצינו לו מקום בזיכרון באופן הבא:

```
double[] scores = new double[NUM_OF_RUNNERS];
```

ננסח זאת באופן כללי:

הצהרה על מערך מטיפוס כלשהו נכתבת בשפת C# כך:

```
טיפוס[] שם המערך;
```

שם הטיפוס, לאחריו סוגריים מרובעים ואז שמו של המערך.

הקצאת שטח למערך מתבצעת באמצעות ההוראה **new**, אחריה מופיע טיפוס איברי המערך, ואחריו מופיע בסוגריים מרובעים מספר איברי המערך:

```
new טיפוס[מספר הערכים];
```

כזכור, ניתן לצרף את ההצהרה וההקצאה בהוראה אחת:

```
טיפוס[] שם המערך = new טיפוס[מספר הערכים];
```

וניתן גם לבצען בנפרד:

```
טיפוס[] שם המערך;
שם המערך = new טיפוס[מספר הערכים];
```

כדי להמחיש את המושגים החדשים נתבונן בדוגמה הבאה:

בשורה הבאה יש הצהרה על מערך מטיפוס שלם, והקצאת שטח המספיק ל-5 איברים.

```
int[] arr = new int[5];
```

אם כך המערך arr מורכב בעצם מחמישה תאים בזיכרון:

	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
arr					

האיבר הראשון במערך arr הוא arr[0], האיבר השני הוא arr[1] וכן הלאה. האיבר האחרון הוא arr[4].

ניתן להתייחס לכל איבר במערך arr כאל משתנה מטיפוס שלם. למשל:

- השמה:

```
arr[4] = num;
```

- קלט:

```
arr[i] = int.Parse(Console.ReadLine());
```

- פלט:

```
Console.WriteLine(arr[i]);
```

נראה כיצד ייראה המערך arr לאחר ביצוע ההוראות הבאות:

```
arr[0] = 9;
arr[1] = 7;
arr[2] = 2 * 4;
arr[3] = 4 + arr[1];
arr[4] = arr[2] + arr[3];
```

	arr[0]	arr[1]	arr[2]	arr[3]	arr[4]
arr	9	7	8	11	19

שאלה 10.2

נניח כעת שבמרוץ הקרטינג משתתפים רק שלושה מתחרים, כלומר ערכו של הקבוע NUM_OF_RUNNERS הוא 3. הקלט לתוכנית (כלומר זמני ההקפה שלהם) הוא: 4.40 4.80 4.20.
בנו טבלת מעקב לתוכנית BelowAverage ועקבו אחר מהלך ביצוע האלגוריתם לפי הנתונים החדשים. מהו הפלט המתקבל?

שאלה 10.3

הוסיפו הוראה או הוראות לתוכנית BelowAverage כך שהפלט יהיה רשימת זמני ההקפות הנמוכים מהממוצע.

שאלה 10.4

נתון קטע התוכנית הבא:

```
int a1,a2;
int[] arr = new int[4];
arr[0] = 2;
a1 = int.Parse(Console.ReadLine());
arr[2] = int.Parse(Console.ReadLine());
a2 = int.Parse(Console.ReadLine());
arr[3] = 2 * arr[2];
arr[1] = a2 + arr[2];
a1 = a[1] + a[2] + a1;
Console.WriteLine("{0} {1}", a1, a2);
for (i = 0; i < arr.Length; i++)
    Console.WriteLine(arr[i]);
```

בנו טבלת מעקב אחר מהלך ביצוע קטע התוכנית עבור הקלט: 3 2 1.
מה יהיה פלט קטע התוכנית עבור הקלט הנתון?

שאלה 10.5

נתונה הצהרת המערך הבאה:

```
int[] temp = new int[10];
```

א. כתבו לולאה להשמת הערך 0 בכל אחד מאיברי המערך.
ב. כתבו לולאה לקליטת עשרה נתוני קלט בעשרת איברי המערך.
ג. כתבו לולאה להכפלה ב-2 של ערכו של כל איבר במערך.
ד. כתבו לולאה להצגה של חצי מערכו של כל איבר במערך.

שאלה 10.6

פתחו אלגוריתם אשר הקלט שלו הוא רשימה של עשרה ציונים, והפלט שלו הוא רשימה הכוללת לכל ציון את מרחקו מהציון הממוצע. מרחקו של ציון מהציון הממוצע הוא |ציון ממוצע – ציון|, כלומר הערך המוחלט של ההפרש של הציון והציון הממוצע. ישמו את האלגוריתם בשפת C#.

שימו ♥: חשוב להבחין בין ערכו של **איבר** ובין ערכו של **המציין של איבר**, כפי שמראה הדוגמה הבאה.

נניח שנתון המערך arr הבא:

arr

arr[0]	arr[1]	arr[2]
-1	2	1

ערכו של האיבר הראשון הוא 1-, ערכו של האיבר השני הוא 2 וערכו של האיבר השלישי הוא 1.
נתון משתנה i, מטיפוס שלם, ונתון קטע התוכנית הבא:

```
1. i = 0;
2. Console.WriteLine("{0} {1}", i, arr[i]);
3. i = i + 1;
4. Console.WriteLine("{0} {1}", i, arr[i]);
5. arr[i] = arr[i] + 1;
6. Console.WriteLine("{0} {1}", i, arr[i]);
```

הנה טבלת המעקב אחר מהלך ביצוע ההוראות שבקטע התוכנית:

מספר שורה	המשפט הבא לביצוע	i	arr[0]	arr[1]	arr[2]	פלט
1	i = 0	0	-1	2	1	
2	Console.Write("{0} {1}",i, arr[i])	0	-1	2	1	0 -1
3	i = i + 1	1	-1	2	1	
4	Console.Write("{0} {1}",i, arr[i])	1	-1	2	1	1 2
5	arr[i] = arr[i] + 1	1	-1	3	1	
6	Console.Write("{0} {1}",i,arr[i])	1	-1	3	1	1 3

בקטע התוכנית הזה תפקידו של המשתנה i הוא להיות מציין של איברי המערך arr, כלומר משתנה שבאמצעותו פונים אל איברי המערך. כפי שניתן לראות מטבלת המעקב ערכו של i אינו שווה בהכרח לערכו של האיבר שנמצא במקום ה-i, כלומר ל-arr[i].

בפרט, שינוי בערכו של האיבר שנמצא במקום ה-i אינו משפיע על ערכו של i, כפי שמדגימות הוראת ההשמה שבשורה 5 והוראת הפלט שבשורה 6.

שאלה 10.7

כתבו קטע תוכנית המצהיר על מערך מטיפוס שלם בגודל 10, ומציב בכל תא ערך מספרי ששווה לריבוע מקומו הסידורי. למשל, בתא 0 יהיה הערך 0 ובתא 5 יהיה הערך 25.

שאלה 10.8

במערך t שלהלן שמורים ערכים שלמים:

```
int[] t = new int[20];
```

א. מה מטרת משפט ה-**for** הבא:

```
for (int i = 0; i < t.Length; i++)
    if (t[i] > i)
        Console.WriteLine(i);
```

ב. כתבו משפט **for** אשר יציג כפלט את מיקומם במערך (כלומר, את מצייניהם) של כל האיברים במערך שערכם כפול ממיקומם הסידורי במערך.

שאלה 10.9

בתוכנית הבאה נשתמש במערך של תווים:

```
/*
קלט: עשר אותיות לועזיות
פלט: ______________________________
*/
using System;
public class Letters
{
    public static void Main()
    {
        char[] letters = new char[10];
        for (int i = 0; i < letters.Length; i++)
        {
            Console.Write("Enter a character: ");
            letters[i] = char.Parse(Console.ReadLine());
        } // for
        for (int i = 0; i < letters.Length; i++)
            if (letters[i]  == letters[letters.Length -1] )
                Console.WriteLine(i);
    } // Main
} // Letters
```

א. מהו פלט התוכנית עבור הקלט: ABBASABABA?

ב. מהי מטרת התוכנית? מלאו את תיאור הפלט בהערה שבראש התוכנית.

ג. האם נחוץ שימוש במערך לצורך השגת המטרה שתיארתם בסעיף ב?

בדוגמאות שראינו עד כה ההצהרה על מערך לוותה בהקצאה מיידית של מקום עבורו, בעזרת הפעולה **new**. עם זאת, הזכרנו שהקצאת המקום בזיכרון יכולה להתבצע גם מאוחר יותר. לעתים, אכן נרצה לדחות את ההקצאה, למשל, כאשר לא ידוע מראש גודל המערך והוא תלוי בקלט לתוכנית, כפי שמדגימה הבעיה הבאה.

בעיה 2

מטרת הבעיה ופתרונה: הצגת מערך שאורכו אינו ידוע מראש.

> לכיתה המדעית יתקבלו רק התלמידים אשר ציוניהם במבחן הקבלה גבוה מהציון הממוצע של הניגשים למבחן.
> פתחו אלגוריתם אשר קולט את מספר הניגשים למבחן, ואחר כך קולט את רשימת הציונים של הנבחנים. פלט האלגוריתם יהיה מספר התלמידים המתקבלים לכיתה. ישמו את האלגוריתם בשפת C#.
> ניתן להניח כי לפחות תלמיד אחד ניגש למבחן.

פירוק הבעיה לתת-משימות

התהליך המבוקש כמעט זהה לזה שבפתרון בעיה 1: עלינו לקלוט נתונים ולשמור אותם, לחשב את הממוצע, ולמנות את מספר הנתונים במערך ה**גדולים** מהממוצע. (שלא כמו בבעיה 1, שם היה עלינו למנות את מספר הנתונים ה**קטנים** מהממוצע).

אבל ההבדל העיקרי בין בעיה זו לבעיה 1 הוא שכעת לא ידוע מראש מספר הנתונים, ולכן צריך קודם כול לקרוא ערך זה מהקלט.

בהתאם לכך נקבל את הפירוק הבא לתת-משימות:

1. קליטת מספר הניגשים למבחן
2. קליטת הציונים, שמירתם וצבירתם
3. חלוקת הסכום המצטבר במספר הנבחנים
4. השוואת כל ציון לממוצע, ומנייתו אם הוא גדול מהממוצע

בחירת משתנים

נשתמש במשתנים הבאים:

numOfStudents – מטיפוס שלם, לשמירת מספר התלמידים הנבחנים
grades – מערך באורך numOfStudents מטיפוס ממשי, לשמירת הציונים
sumOfGrades – מטיפוס ממשי, ישמור את סכום הציונים
averageGrade – ממוצע הציונים, מטיפוס ממשי
aboveAverageGrade – למניית מספר הציונים מעל לממוצע, מטיפוס שלם

האלגוריתם

האלגוריתם כמעט זהה לאלגוריתם שהוצג בפתרון בעיה 1:

שאלה 10.10
כתבו את האלגוריתם לפתרון בעיה 2. היעזרו באלגוריתם שניתן לפתרון בעיה 1, ושנו אותו כך שיכלול הוראת קלט של מספר התלמידים הנבחנים, יתייחס למשתנים שנבחרו וייתן את הפלט המבוקש.

יישום האלגוריתם

הנה קטע התוכנית המטפל בהצהרה על המערך, בקליטת מספר התלמידים, ובהקצאת מקום בזיכרון עבור המערך:

```
int numOfstudents;
int[] grades;
Console.Write("Enter number of students: ");
numOfstudents = int.Parse(Console.ReadLine());
grades = new int[numOfstudents];
```

מיד אחרי שמוקצה למערך מקום מתאים בזיכרון, התכונה Length של המערך שומרת את אורכו.

שאלה 10.11

השלימו את התוכנית לפתרון הבעיה.

סוף פתרון בעיה 2

שאלה 10.12

כתבו קטע תוכנית המקבלת כקלט מספר שלם חיובי וזוגי, ומקצה מערך מטיפוס שלם בגודל הערך שנקלט. לאחר מכן התוכנית תשים באיברי המערך הנמצאים במחציתו הראשונה את המספר 0, ובאיברי המערך הנמצאים במחציתו השנייה את המספר 1.

בדוגמאות שראינו עד כה איברי מערך נסרקו באופן רציף, זה אחר זה. לעתים, נרצה לסרוק איברי מערך באופן לא רציף. למשל, נניח ש-arr הוא מערך של עשרה איברים ויש להציג כפלט את ערכיהם של כל האיברים שבמקומות הזוגיים במערך. גם עבור סריקה כזאת נוכל להשתמש בהוראה לביצוע-חוזר באורך ידוע מראש, אך נקדם את משתנה הבקרה של הלולאה בדילוגים של 2:

```
for (int i = 0; i < arr.Length; i = i + 2)
    Console.WriteLine(arr[i]);
```

שאלה 10.13

נתון המערך arr המכיל מאה איברים מטיפוס שלם.

א. תארו את מטרת הלולאה הבאה:

```
for (int i = 0; i < arr.Length; i++)
    if (arr[i] % 5 == 0)
        Console.WriteLine(arr[i]);
```

ב. תארו את מטרת הלולאה הבאה:

```
for (int i = 0; i < arr.Length; i++)
    if (i % 5 == 0)
        Console.WriteLine(arr[i]);
```

ג. כתבו לולאה יעילה יותר (שמספר הסיבובים בה קטן יותר) להשגת המטרה של סעיף ב.

בבואנו לכתוב הוראה לביצוע-חוזר המבצעת סריקה ועיבוד של איברי מערך, נתאים את סוג ההוראה להגדרת הסריקה שצריכה להתבצע.

אם ידועים מראש הן המציין שממנו צריכה הסריקה להתחיל והן המציין שהיא אמורה להסתיים בו, והסריקה עצמה היא בדילוגים ידועים, נשתמש בהוראה לביצוע-חוזר שמספר הסיבובים בה ידוע מראש (המיושמת בלולאת **for**).

אם סיום הסריקה תלוי בקיום תנאי, נשתמש בביצוע-חוזר-בתנאי (המיושמת בלולאת **while**).

שאלה 10.14

במערך s יש ערכים מטיפוס תו.

א. תארו את מטרת קטע התוכנית הבא:

```
int i = 0;
int len = s.Length;
while (s[i] < s[len - 1])
    i = i + 1;
Console.WriteLine(i);
```

ב. תארו את מטרת קטע התוכנית הבא:

```
int c = 0;
int len = s.Length;
for (int i = 0; i < len; i++)
    if ( (i % 10 == 0) && (s[i] < s[len - 1]) )
        c = c + 1;
Console.WriteLine(c);
```

ג. לולאת ה-**for** שבקטע התוכנית בסעיף ב מתבצעת מספר פעמים השווה לאורך המערך s.
כתבו לולאה שמבצעת אותה המשימה אך מספר הסיבובים בה יהיה הרבה יותר קטן.

בעיה 3

מטרת הבעיה ופתרונה: שימוש במערך כדי לתאר מבנה כגון לוח משחק, ושימוש בביטויים חשבוניים כמצייני מיקום.

נתאר משחק על לוח ובו שורה של שלושים משבצות ובחלקן מצויים מוקשים. המשבצות ממוספרות מ-1 עד 30.
שחקן מתקדם לאורך הלוח באמצעות הטלת קובייה שעל פאותיה הספרות 1 עד 6. אם המספר המתקבל מהטלת הקובייה מוביל למשבצת שיש בה מוקש, השחקן נשאר במקומו. אם המספר המתקבל מהטלת הקובייה מוביל למשבצת פנויה השחקן יכול להתקדם.
פתחו אלגוריתם אשר הקלט שלו הוא תיאור הלוח המתקבל באמצעות שלושים תווים שערכיהם 'T' (עבור משבצת פנויה) או 'F' (עבור מוקש), ואחריו מספר המציין את מספר המשבצת שהשחקן מוצב עליה. פלט האלגוריתם הוא הטלות הקובייה אשר עבורן יוכל השחקן להתקדם. הניחו שהשחקן מוצב על משבצת שמספרה בין 1 ל-24 ואין בה מוקש. ישמו את האלגוריתם בשפת התכנות C#.

ניתוח הבעיה בעזרת דוגמאות

נתבונן בלוח הבא, כאשר הסימון ☺ מציין את מיקום השחקן והסימון ☼ מציין מוקש:

	☺		☼		☼				☼		......	☼		

בלוח זה, השחקן יכול לקדם את הכלי עבור כל אחת מן ההטלות: 1, 3, 5 או 6, אך לא יוכל לקדם אותו עבור ההטלות 2 או 4.

פירוק הבעיה לתת-משימות

הנה חלוקה ראשונית לתת-משימות עבור פתרון הבעיה:

1. קליטה ושמירה של מצב הלוח ושל מקום השחקן על הלוח
2. חישוב הטלות הקובייה המותרות להתקדמות והצגתן כפלט

בחירת משתנים

את לוח המשחק נתאר באמצעות מערך של ערכים בוליאניים בגודל 30, שמצייני איבריו נעים בין 0 ל-29. כל איבר במערך ישמור את מצב המשבצת המתאימה בלוח. הערך **true** יתאר משבצת פנויה, ואילו הערך **false** יתאר מוקש. בנוסף נזדקק למשתנה שיציין את מיקום השחקן על הלוח.

אם כך אלה יהיו המשתנים שנשתמש בהם:

board – מערך של 30 איברים מטיפוס בוליאני
pawn – מטיפוס שלם, לשמירת מיקום השחקן על הלוח

למשל עבור הדוגמה שלעיל המערך board ייראה כך:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	…	[27]	[28]	[29]
true	**true**	**true**	**false**	**true**	**false**	**true**	**true**	**true**	**false**	**true**	…	**false**	**true**	**true**

והמשתנה pawn יכיל את הערך 1 (כיוון שהשחקן נמצא על המשבצת השנייה).

ההתייחסות לאיברי המערך היא בעזרת מציין, לדוגמה הערך בתא board[pawn] מבטא את מצב המשבצת שכלי השחקן מוצב עליה.

שימו ♥: אמנם לוח המשחק ממוספר מ-1 עד 30, אבל המערך המתאר אותו ממוספר מ-0 עד 29. לכן ערכי המשתנה pawn יהיו בתחום 0 עד 29.

האלגוריתם

כיצד נשתמש במערך board כדי לחשב את ההטלות שעבורן יוכל השחקן להתקדם?

כדי לחשב את ההטלות שעבורן יוכל השחקן להתקדם יש לבדוק את מצבן של שש המשבצות העוקבות למשבצת שהשחקן מוצב עליה. כיוון שמשבצת זו מתוארת באמצעות האיבר board[pawn] במערך, שש המשבצות העוקבות מיוצגות על ידי ששת האיברים הבאים לפי הסדר: board[pawn+1], board[pawn+2], ..., board[pawn+6].

עבור כל איבר מששת האיברים האלה יש לבדוק אם הוא מייצג משבצת פנויה (כלומר אם ערכו הוא **true**). אם כן, יש להציג כפלט את הטלת הקובייה אשר מביאה אליו את השחקן.

למשל, אם ערכו של board[pawn+3] מבטא משבצת פנויה, יוצג 3 כפלט, כיוון שהטלת 3 בקובייה מביאה את השחקן למשבצת פנויה.

1. **עבור כל** i שלם מ-0 עד גודל המערך פחות 1 **בצע:**
   1.1. קלוט את מצב המשבצת ה-i של הלוח
2. קלוט את מיקום השחקן
3. **עבור כל** i שלם מ-1 עד 6 **בצע:**
   3.1. **אם** המשבצת במרחק i ממיקום השחקן היא ריקה
      3.1.1. הצג את ערכו של i

התוכנית המלאה

```
/*
קלט: תיאור לוח משחק בן 30 משבצות
ומיקום משבצת שעליה מוצב שחקן (בין 1 ל-24)
פלט: הטלות קובייה המביאות את השחקן למשבצת פנויה
*/
using System;
public class MineBoardGame
{
    public static void Main()
    {
        // הגדרת קבוע
        const int BOARD_SIZE = 30;
        // הגדרת משתנים
        bool[] board = new bool[BOARD_SIZE];
        char cellStatus;
        int pawn;
        // קלט הלוח
        Console.WriteLine("Enter the board details: Type F for a " +
                          "cell with a mine, and T for a free cell: ");
        for (int i = 0; i < board.Length; i++)
        {
            Console.Write("Enter status of cell number {0}: ", i+1);
            cellStatus = char.Parse(Console.ReadLine());
            board[i] = (cellStatus == 'T');
        }
        // קלט מיקום השחקן והתאמתו לאופן שמירת הלוח
        Console.Write("Enter the pawn position: ");
        pawn = int.Parse(Console.ReadLine());
        pawn = pawn - 1;
        // פלט
        Console.Write("Throws that lead to empty positions: ");
        for (int i = 1; i <= 6; i++)
            if (board[pawn + i])
                Console.Write("{0} ",i);
    }// Main
} //class MineBoardGame
```

שימו ♥: בלולאה הראשונה השתמשנו במציין `i` כדי לעדכן את מצב המשבצת התורנית בלוח. כדי להודיע למשתמש את מספר המשבצת שאת מצבה עליו להקליד השתמשנו בביטוי `i + 1`. הביטוי מתאים את מצייני המערך (הנעים בין 0 ל-29) לאופן הספירה של הלוח (מ-1 עד 30). וכך בסיבוב הראשון של הלולאה, כאשר ערכו של `i` הוא 0, מוצגת ההודעה:

`Enter status of cell number 1:`

בלולאה השנייה השתמשנו בביטוי `i + pawn` כמציין. הביטוי `i + pawn` מבטא מיקום במערך שמרחקו מהמיקום `pawn` הוא `i`.

? לאחר קליטת מיקום השחקן, מדוע מושם במשתנה `pawn` הערך הנקלט פחות 1?

המשתמש בתוכנית יקליד את מיקום השחקן על הלוח כמספר שלם בין 1 ל-30. לעומת זאת, ערכי המשתנה `pawn` צריכים להתאים לאופן מספור המערך מ-0 עד 29.

סוף פתרון בעיה 3

שאלה 10.15

בנו טבלת מעקב אחר מהלך ביצוע התוכנית `MineBoardGame` לפתרון בעיה 3 עבור הקלט:

T T F F T F T F T . . . T 2

כלומר יש מוקש במשבצת השלישית והרביעית, ומהמשבצת השישית ואילך יש מוקש בכל משבצת זוגית. השחקן נמצא במשבצת השנייה.

מהו הפלט המתקבל?

שאלה 10.16

הרחיבו את האלגוריתם לפתרון בעיה 3 כך שיציג כפלט לא רק את ההטלות שמאפשרות לשחקן להתקדם, אלא גם את מספרי המשבצות שהטלות אלו מובילות אליהן. למשל עבור הקלט שבשאלה הקודמת יכלול הפלט החדש גם את הרשימה: 7 5.

שאלה 10.17

נניח שכתבנו את משפטי ההצהרה הבאים:

```
int[] arr = new int[100];
int position;
int counter = 0;
```

באיברי המערך `arr` הושמו ערכים ובמשתנה `position` נקלט ערך חיובי שלם בתחום 1 עד 79.

א. תארו את מטרת קטע התוכנית הבא:

```
if (arr[position] == arr[position - 1] )
    Console.WriteLine("previous one is equal");
if (arr[position] == arr[position + 1] )
    Console.WriteLine("next one is equal");
```

ב. תארו את מטרת קטע התוכנית הבא:

```
for (int i = 0; i < 10; i++)
    if (arr[position] == arr[position + i])
        counter = counter + 1;
Console.WriteLine(counter);
```

ג. כתבו לולאה שמטרתה להציג את מצייניהם של איברי המערך, מבין 20 האיברים העוקבים ל-`arr[position]`, אשר ערכם גדול מערכו של `arr[position]`.

ד. כתבו לולאה שמטרתה להציג את מצייניהם של איברי המערך, מבין 20 האיברים העוקבים ל-`arr[position]`, אשר ערכם גדול **בדיוק ב-1** מערכו של `arr[position]`.

שאלה 10.18

פתחו וישמו אלגוריתם אשר הקלט שלו הוא רצף של 20 תווים, והפלט שלו הוא מספר הזוגות של תווים עוקבים ששווים לזוג התווים האחרון.

למשל, עבור הקלט abcabacadcababddaaab הפלט הוא 4 כיוון שהזוג האחרון הוא ab, ויש עוד ארבעה זוגות השווים ל-ab.

שימו ♥: עבור קלט של 20 תווים זהים הפלט צריך להיות 18 (כיוון שכל הזוגות, מהראשון ועד הזוג לפני האחרון, שווים לזוג האחרון).

שאלה 10.19

במשחק נתון לוח הכולל שורה של 25 משבצות אשר בכל אחת מהן רשום מספר שלם בין 0 ל-5. על הלוח מוצב שחקן בין המשבצת העשירית והחמש-עשרה.

השחקן מטיל קובייה כדי לנסות ולהתקדם על הלוח. אם המספר המתקבל בהטלת הקובייה מוביל למשבצת אשר המספר הרשום בה קטן או שווה למספר המתקבל בקובייה, השחקן מקדם את הכלי למשבצת זו. אחרת השחקן נסוג אחורה מספר משבצות השווה למספר המתקבל בקובייה.

פתחו אלגוריתם אשר הקלט שלו הוא תיאור הלוח (25 מספרים שערכיהם נעים בין 0 ל-5), ואחריו מיקום השחקן על הלוח וערכה של הטלת הקובייה. הפלט שלו הוא מיקומו החדש של השחקן על הלוח. ישמו את האלגוריתם בשפת #C.

שימו ♥: יש להתאים בין מיקום השחקן כפי שנקלוט (מספר בין 1 ל-25) לבין מציין המערך (מספר בין 0 ל-24).

שאלה 10.20

מה יהיו ערכי איברי המערך בסיום קטע התוכנית הבא?

```
int[] arr = new int[10];
for (int i = 0; i < arr.Length / 2; i++)
    arr[i * 2] = i;
```

שאלה 10.21

המערך arr הוא מערך באורך 10 המכיל מספרים שלמים. תארו מה יהיו ערכי איברי המערך בסיום קטע התוכנית הבא.

```
for (int i = 0; i < arr.Length; i++)
    arr[i] = arr[arr.Length - i];
```

שאלה 10.22

א. המערך arr הוא מערך באורך 10 המכיל מספרים שלמים. תארו מה יהיו ערכי איברי המערך בסיום קטע התוכנית הבא.

```
for (int i = 0; i < arr.Length - 1; i++)
    arr[i + 1] = arr[i];
```

ב. תקנו את קטע התוכנית וכתבו אותו מחדש כך שיבצע הזזה של ערכי התאים במערך בצורה מעגלית, כלומר, התא השני יכיל את ערכו המקורי של התא הראשון, התא השלישי יכיל את ערכו המקורי של התא השני וכן הלאה, והתא הראשון – יכיל את ערכו המקורי של התא האחרון.

שאלה 10.23

א. חברי קבוצת "סיורי הגליל" הולכים תמיד בטור, זה אחר זה. לאחר עשרים דקות הליכה, הראשון בטור נעצר וממתין עד שהטור כולו עובר אותו, ואז מצטרף לטור שוב כאחרון. פתחו אלגוריתם המקבל כקלט את מספר חברי הקבוצה, ולאחר מכן את רשימת השמות של חברי הקבוצה, לפי הסדר שבו הם מתחילים את הטיול. פלט האלגוריתם יהיה סדר חברי הקבוצה לאחר שעה ורבע של הליכה. ישמו את האלגוריתם בשפת התכנות #C.

ב. חברי קבוצת "סיורי הגליל למתחילים" הולכים תמיד בטור; המדריך צועד בראש, ובסוף נמצאים החובש, המלווה והמאסף. בקדמת הטור צועדות הבנות ואחריהן צועדים הבנים. רק הבנים מחליפים את מקומותיהם כל עשרים דקות כפי שתואר עבור קבוצת "סיורי הגליל". פתחו אלגוריתם המקבל כקלט את מספר הבנים ואת מספר הבנות בקבוצה, ואת הרשימה של שמות חברי הקבוצה לפי הסדר שבו הם מתחילים את הטיול. פלט האלגוריתם יהיה סדר חברי הקבוצה לאחר שעה ורבע של הליכה. ישמו את האלגוריתם בשפת התכנות #C.

שאלה 10.24

פתחו אלגוריתם המקבל כקלט סדרת מספרים בסדר עולה. האלגוריתם מסדר ושומר את הסדרה הנתונה בסדר יורד. ישמו את האלגוריתם בשפת התכנות #C.

10.2 חריגה מגבולות מערך

? בהגדרת בעיה 3 נכללה הנחה כי השחקן מוצב על משבצת שמספרה בין 1 ל-24. מדוע הנחה זו חשובה?

נניח כי השחקן עומד על משבצת שערכה גבוה מהמספר 24, למשל 25. הטלת הקובייה נותנת מספר כלשהו בין 1 ל-6. אם הטלת הקובייה תיתן את המספר 6, השחקן ינסה לנוע 6 צעדים קדימה החל מהמשבצת ה-25. כך יגיע השחקן למשבצת מספר 31. משבצת זו אינה קיימת כמובן ואינה חוקית בתנאי המשחק!

ניסיון להתייחס במהלך הרצת תוכנית בשפת #C לתא שאיננו קיים במערך (למשל `board[30]`) יגרום להפסקת פעולתה של התוכנית ולהצגת הודעת שגיאה על חריגה מגבולות המערך. חריגה מגבולות המערך היא שגיאת ריצה, המתגלה בזמן ריצת התוכנית ולא בזמן ההידור.

שאלה 10.25

נוסיף לפתרון בעיה 3 משתנה בשם `limit` מטיפוס שלם, אשר ישמש כערך סופי למשתנה הבקרה `i` של הלולאה השנייה (לולאת הצגת הפלט).

נשתמש ב-`limit` כדי להרחיב את משפטי התוכנית `MineBoardGame`, כך שהתוכנית תאפשר לקלוט ערך בין 1 ל-29 למיקום השחקן (במקום 24 כמו בבעיה המקורית). יחד עם זאת לא תהיה חריגה מגבולות המערך בלולאה השנייה.

לפני הלולאה השנייה נוסיף הוראה לביצוע-בתנאי:

```
if (________________)
    ________________
else
    ________________
```

וכותרת הלולאה השנייה תהיה:

```
for (int i = 1; i <= limit; i++)
```

השלימו את ההוראה לביצוע-בתנאי, כך שהתוכנית תבצע את הדרוש.

שאלה 10.26

נתון המערך `arr` שהצהרתו היא:

```
int[] arr = new int[8];
```

וערכי איבריו הם:

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]
10	20	40	30	4	5	7	6

א. מה יהיה פלט הלולאה הבאה?

```
for (int i = 0; i < arr.Length - 1; i++)
    if ((i == arr[i]) || (i == arr[i + 1]))
        Console.WriteLine(i);
```

ב. מה יהיו ערכי המערך אחרי ביצוע הלולאה הבאה?

```
for (int i = 0; i < arr.Length; i++)
{
    if (arr[i] == (10 * i))
        arr[i] = arr[i] + 1;
    else
        arr[i] = arr[i] - 1;
}
```

ג. עבור אילו מארבעת הלולאות הבאות תהיה חריגה מגבולות המערך? תקנו לפי הצורך.

.1

```
for (int i = 0; i < arr.Length; i++)
    arr[i] = arr[i + 1];
```

.2

```
for (int i = 0; i < arr.Length - 1; i++)
    arr[i] = arr[i - 1];
```

.3

```
for (int i = 0; i < arr.Length - 1; i++)
    arr[i] = arr[i] * 2;
```

.4

```
for (int i = 0; i < arr.Length - 1; i++)
    arr[i] = arr[i * 2];
```

שאלה 10.27

השאלה מתייחסת למערך arr בן 8 האיברים שהוגדר בשאלה הקודמת.

כתבו לולאה אשר תשווה את ערכיהם של כל זוג איברים שכנים במערך (arr[0] ל-arr[1], arr[1] ל-arr[2], ... , arr[6] ל-arr[7]), ותחשב את מספר זוגות האיברים השכנים שערכיהם שווים זה לזה.

שימו ♥: לא לחרוג מגבולות המערך.

שאלה 10.28

על תמר הוטל לכתוב תוכנית המגדירה מערך בן 50 תאים, ומציבה בהם את הערכים:

0.5 1 1.5 2 2.5 3 3.5...

תמר כתבה:

```
double[] arr = new double[50];
for (int i = 0; i < arr.Length * 2; i++)
    arr[i] = i / 2;
```

האם קטע התוכנית נכון? אם איננו נכון – תקנו את השגיאות כך שתושג המטרה המבוקשת.

שאלה 10.29 (מבגרות 2002)

לפניכם קטע תוכנית:

```
for (int i = 0; i < n - 2; i++)
    if (a[i] + 2 == a[i + 2])
        Console.WriteLine("{0} {1}", i, a[i]);
```

a הוא המערך הבא בגודל 10:

a:										
	3	18	5	20	2	4	5	6	1	9

א. עקבו בעזרת טבלת מעקב אחר קטע התוכנית עבור n=10 ועבור המערך a הנתון, ורשמו מה יהיה הפלט.

ב. הסבירו מדוע קבע כותב קטע התוכנית שהתנאי להמשך ביצוע הלולאה יהיה i < n-2 ולא i<n.

10.3 קשרים בין מערכים

בדוגמאות שראינו עד כה השתמשנו במערך יחיד. לעתים פתרון בעיה מצריך יותר ממערך אחד. בסעיף זה נראה פתרונות אלגוריתמיים המשתמשים ביותר ממערך אחד. בחלקו הראשון של הסעיף נציג עיבוד של כמה מערכים במקביל בקצב התקדמות זהה, ובחלקו השני נעסוק גם בעיבוד של כמה מערכים שאינו נעשה באותו קצב התקדמות.

עיבוד מערכים במקביל ובקצב התקדמות זהה

השאלה הבאה עוסקת בניתוח קטע תוכנית המעבד במקביל שני מערכים, ופתרון השאלה שאחריה מצריך שימוש בשלושה מערכים.

שאלה 10.30

m1 ו-m2 הם שני מערכים שהוצהרו באופן הבא:

```
const int SIZE = 10;
int[] m1 = new int[SIZE];
int[] m2 = new int[SIZE];
```

נתונים שני קטעי התוכניות הבאים אשר אמורים להציג כפלט אם שני המערכים מכילים בדיוק את אותם איברים באותו סדר.

א.

```
bool compare = false;
for (int i = 0; i < SIZE; i++)
if (m1[i] == m2[i])
        compare = true;
if (compare)
    Console.WriteLine("The arrays
                  are equal");
else
    Console.WriteLine("The arrays
                  are not equal");
```

ב.

```
bool compare = false;
for (int i = 0; i < SIZE; i++)
    compare = (m1[i] == m2[i]);
if (compare)
   Console.WriteLine("The arrays
                  are equal");
else
    Console.WriteLine("The arrays
              are not equal");
```

שני קטעי התוכניות שגויים. ענו על הסעיפים הבאים ביחס **לכל אחד** מהקטעים:

א. הביאו דוגמה לערכים של איברים בשני המערכים שקטע התוכנית משיג את מטרתו עבורם.

ב. הביאו דוגמה לערכים של איברים בשני המערכים שקטע התוכנית אינו משיג את מטרתו עבורם, והסבירו מדוע זה קורה.

ג. תקנו את קטע התוכנית כך שישיג את מטרתו.

שאלה 10.31

כיתה י"5 ניגשה למבחן. לאחר המבחן הוחלט כי הכיתה כולה תיגש למבחן נוסף, והציון הגבוה מבין שני ציוני המבחנים לכל תלמיד יהיה הציון הקובע.

פתחו אלגוריתם הקולט את מספר התלמידים בכיתה, ולאחר מכן קולט את ציוני המבחן הראשון למערך בשם grade1 ואת ציוני המבחן השני למערך בשם grade2. האלגוריתם ימלא מערך בשם finalGrade כך שיכיל את הציון הקובע לכל אחד מתלמידי הכיתה, ויציג כפלט עבור כל אחד מהתלמידים את ציון המבחן הראשון, את ציון המבחן השני ואת הציון הקובע בליווי הודעות מתאימות. ישמו את האלגוריתם בשפת C#.

עיבוד מערכים במקביל בקצב התקדמות לא זהה

בעיה 4

מטרת הבעיה ופתרונה: הצגת עיבוד של כמה מערכים במקביל בקצב התקדמות לא זהה.

> פתחו אלגוריתם המקבל כקלט מספר חיובי שלם num, ולאחר מכן קורא מהקלט num מספרים שלמים לתוך מערך numbers. האלגוריתם מעתיק מהמערך numbers את המספרים החיוביים למערך positive ואת המספרים השליליים למערך negative, ושומר על סדר המספרים. ישמו את האלגוריתם בשפת C#.

ניתוח הבעיה בעזרת דוגמאות

נניח כי הקלט הוא 2- 4 3 1- 1 5.

המערך numbers ישמור בתוכו 5 מספרים שלמים. בעקבות ההעתקה הדרושה יכיל המערך positive שלושה איברים: 1, 3 ו-4. המערך negative יכיל שני איברים: 1- ו-2-.

אנו רואים כי לא רק שלכל מערך מועתק מספר שונה של איברים, אלא שקצב ההתקדמות על המערכים הללו שונה: ייתכן כי נסרוק כמה איברים ברצף במערך numbers, נשים אותם באחד המערכים, אך במקביל לא נתקדם כלל בהשמה במערך האחר. למשל בדוגמת הקלט שלעיל, בזמן שבמערך numbers נסרק האיבר השלישי והרביעי בזה אחר זה, אין כלל התקדמות במילוי המערך negative.

פירוק הבעיה לתת-משימות

1. קליטת מספר הערכים לעיבוד
2. קליטת רשימת הערכים ושמירתה במערך numbers
3. ביצוע במקביל של סריקת המערך numbers ומילוי המערכים positive ו-negative

בחירת משתנים

כדי לסרוק את המערכים בקצב התקדמות לא זהה, עלינו לשמור שלושה מציינים שונים, אחד עבור כל מערך: המציין של המערך numbers יסרוק את איברי המערך בזה אחר זה. המציין של המערך positive והמציין של המערך negative יצביעו בכל רגע על המקום הפנוי הבא במערך המתאים להם, כדי שהעתקת האיבר הבא תתבצע למקום הנכון.

המציין של המערך numbers יתקדם ב-1 בכל פעם שנסיים עיבוד של איבר תורן במערך ונתקדם לעבר האיבר הבא.

כל אחד מהמציינים האחרים, של המערך positive ושל המערך negative, יתקדם ב-1 רק אחרי ביצוע העתקה למערך שאליו הוא מתייחס.

נשתמש במשתנים הבאים:

size – מספר הערכים לעיבוד
numbers – מערך מטיפוס שלם, בגודל size
positive – מערך מטיפוס שלם, בגודל size
negative – מערך מטיפוס שלם, בגודל size

indexNum – מטיפוס שלם, יציין את מיקום האיבר התורן במערך numbers
indexPos – מטיפוס שלם, יציין את המקום הפנוי הבא במערך positive
indexNeg – מטיפוס שלם, יציין את המקום הפנוי הבא במערך negative

האלגוריתם

1. קלוט ערך חיובי שלם ב-size
2. **עבור כל** i שלם מ-0 עד size-1 **בצע:**
   2.1. קלוט נתון ב-numbers[i]
3. אתחל את indexPos ב-0
4. אתחל את indexNeg ב-0
5. **עבור כל** indexNum שלם מ-0 עד size-1 **בצע:**
   5.1. אם numbers[indexNum] > 0
      5.1.1. העתק את ערכו של numbers[indexNum] ל-positive[indexPos]
      5.1.2. הגדל את ערכו של indexPos ב-1
   5.2. אחרת
      5.2.1. העתק את ערכו של numbers[indexNum] ל-negative[indexNeg]
      5.2.2. הגדל את ערכו של indexNeg ב-1

שאלה 10.32
ישמו את האלגוריתם לפתרון בעיה 4 בשפת #C.

סוף פתרון בעיה 4

שאלה 10.33
פתחו אלגוריתם אשר מקבל כקלט מספר חיובי שלם, ולאחר מכן קולט רשימת מספרים שאורכה כערך הנתון הראשון. האלגוריתם ישמור את המספרים הנקלטים בשני מערכים שונים, מערך אחד עבור מספרים זוגיים, ומערך אחר עבור מספרים אי-זוגיים. האלגוריתם יציג כפלט את כל הערכים שברשימה: ראשית את כל המספרים הזוגיים, לפי סדר קליטתם ואחר-כך את כל המספרים האי-זוגיים לפי סדר קליטתם. ישמו את האלגוריתם בשפת #C.

שאלה 10.34
נניח ש-arr1 ו-arr2 הם שני מערכים מטיפוס שלם ואורכם אינו בהכרח שווה. כתבו קטע תוכנית אשר מעתיק למערך שלישי arr3, רק את האיברים אשר נמצאים גם ב-arr1 וגם ב-arr2.
למשל, אם ב-arr1 וב-arr2 נמצאים הערכים הבאים:

arr1	36	8	9	73	11	3	4
arr2	4	77	8	15	12		

אז ב-arr3 יהיו הערכים הבאים:

arr3	4	8	0	0	0	0	0

הדרכה: יש להשתמש בקינון של לולאות.

10.4 מערך מונים

בפרק 7 למדנו על משתנה מסוג מונה. למדנו כי מונה הוא משתנה אשר תפקידו למנות מספר של אירועים שמתרחשים (כמו למשל הופעות של נתונים או מספר חישובים שביצענו). כיוון שהשימוש במונה הוא לצורך ספירה, מונה הוא משתנה מטיפוס שלם. לעתים יש למנות במקביל אירועים שונים. לשם כך ניתן להשתמש במערך של מונים, כפי שמדגימה הבעיה הבאה. העבודה עם מערך מונים היא תבנית שימושית לפתרון בעיות אלגוריתמיות.

בעיה 5

מטרת הבעיה ופתרונה: הצגת מערך מונים ושימוש בו

בתוכנית "כוכב נולד" יש N מועמדים ולכל מועמד מספר סידורי בין 1 ל-N. הצופים בבית נתבקשו לבחור את אחד המועמדים ולהצביע עבורו. בכל תוכנית המועמד שמקבל את מספר הקולות הקטן ביותר עוזב את התוכנית. פתחו אלגוריתם אשר מקבל כקלט את מספר המועמדים (מספר חיובי שלם), ואחר כך קולט רשימה של הצבעות של הצופים, המסתיימת במספר 1-. האלגוריתם מודיע מי המועמד אשר קיבל את מספר הקולות הקטן ביותר. ישמו את האלגוריתם בשפת התכנות C#.

פירוק הבעיה לתת-משימות

1. קליטת מספר מועמדים
2. קליטת הצבעות למועמדים תוך מניה של ההצבעות עבור כל מועמד
3. מציאת המונה בעל הערך המינימלי והצגה של המועמד שאליו הוא מתאים

בחירת משתנים

כדי לבצע את תת-משימה 2 עלינו לתפעל במקביל כמה מונים, מונה לכל מועמד. כלומר מספר המונים שווה למספר המועמדים. כמובן שאין אנו יכולים להצהיר על מונה נפרד לכל מועמד: מספר המועמדים אינו ידוע מראש, ואפילו אם היה ידוע, ייתכן כי הוא גדול למדי. בכל מקרה, הצהרה על מונה נפרד לכל מועמד הופכת את התוכנית למסורבלת, לפחות קריאה ולקשה יותר לשינוי (למשל אם מספר המועמדים ישתנה).

לכן ניצור מערך שאורכו כמספר המועמדים, וכל תא בו ייַצג מונה.

נזדקק למשתנים הבאים:

numOfSingers – מטיפוס שלם, שומר את מספר המתמודדים בתחרות
votes – מערך מטיפוס שלם, מערך מונים באורך numOfSingers למניית ההצבעות עבור כל מועמד ומועמד
vote – מטיפוס שלם, שומר הצבעה תורנית מהקלט
min – מטיפוס שלם, לשמירת ההצבעה המצטברת הנמוכה ביותר
minSinger – מטיפוס שלם, שומר את מספר המתמודד שקיבל את מספר ההצבעות הנמוך ביותר

האלגוריתם

לצורך ביצוע תת-משימה 2 יש להשתמש בלולאה, אשר בכל סיבוב בה תיקלט הצבעה נוספת מהקלט, ובהתאם לערכה יגדל ערכו של המונה המתאים. מאחר שמספר ההצבעות אינו ידוע מראש, ומאחר שסוף רשימת ההצבעות מצוין בזקיף, נשתמש בלולאת while שהתנאי בה מתייחס לזקיף.

לצורך ביצוע תת-משימה 3 נחפש את האיבר הקטן ביותר במערך המונים.
שימו ♥: יש להציג כפלט את **מספרו הסידורי** של המתמודד שקיבל את מספר הקולות הקטן ביותר, ולא את מספר הקולות שקיבל. לכן במקרה זה נשתמש בתבנית של מציאת מקום המינימום.

1. קלוט את מספר המתמודדים ב-numOfSingers
2. קלוט מספר מועמד ב-vote
3. **כל עוד** vote ≠ -1 **בצע:**
   3.1. העלה ב-1 את המונה (במערך) של המתמודד שמספרו vote
   3.2. קלוט מספר מועמד ב-vote
4. אתחל את min בערך המונה של המתמודד הראשון
5. אתחל את minSinger ב-1
6. **עבור כל** i שלם מ-2 עד numOfSingers **בצע:**
   6.1. **אם** ערך המונה של מתמודד מספר i קטן מ-min
      6.1.1. השם ב-min את ערך המונה של המתמודד שמספרו i
      6.1.2. השם את i ב-minSinger
7. הצג את ערך minSinger

יישום האלגוריתם

חשוב לשים לב כי המספרים הסידוריים של המועמדים מתחילים ב-1, בעוד שמספור איברי המערך מתחיל מ-0. לכן למשל היישום של הוראה 3.1 באלגוריתם הוא

```
votes[vote - 1]++;
```

והיישום של הוראה 6.1 הוא

```
if (votes[i - 1] < min)
```

התוכנית המלאה

```
/*  קלט: מספר המועמדים בתוכנית "כוכב נולד" והצבעות הצופים למועמדים
    פלט: המספר הסידורי של המתמודד המפסיד
*/
using System;
public class IsraeliIdol
{
    public static void Main()
    {
        // הגדרת משתנים
        int numOfSingers;
        int[] votes;
        int vote;
        int min;
        int minSinger;
```

```
        // קליטת מספר מועמדים והקצאת גודל מערך מתאים
        Console.Write("Enter number of contestants: ");
        numOfSingers = int.Parse(Console.ReadLine());
        votes = new int[numOfSingers];
        // אתחול מערך המונים
        for(int i = 0; i < numOfSingers; i++)
            votes[i] = 0;
        // קליטת בחירת הצופים בלולאת זקיף
        Console.Write("Enter a vote, end the list with -1: ");
        vote = int.Parse(Console.ReadLine());
        while (vote != -1)
        {
            // הוספת קול נוסף למונה המתאים
            votes[vote - 1]++;
            Console.Write("Enter a vote, end the list with -1 ");
            vote = int.Parse(Console.ReadLine());
        }
        // אתחול משתני מינימום ומקום המינימום
        min = votes[0];
        minSinger = 1;
        // חיפוש ערך מינימלי ושמירת ערכו וערך המציין שלו
        for(int i = 2; i <= numOfSingers; i++)
        {
            if (votes[i - 1] < min)
            {
                min = votes[i - 1];
                minSinger = i;
            } // if
        } // for
        // פלט
        Console.WriteLine("contestant number {0} is going home",
                                                         minSinger);
    } //Main
} //class IsraeliIdol
```

שימו ♥: כשם שיש לאתחל משתנה מסוג מונה לאפס, כך גם יש לאתחל כל איבר ואיבר במערך מונים ב-0. אנו מבצעים אתחול כזה במפורש, אף על פי שיכולנו להסתמך על כך שבשפת C# איבריו של מערך מטיפוס מספרי מאותחלים אוטומטית ב-0. אתחול מפורש תורם לבהירות התוכנית.

ועוד שימו ♥: את תבנית מציאת מקום המינימום ניתן גם ליישם בצורה מעט שונה, הנוחה לעבודה עם מערכים. למעשה ביישום זה נשמר רק מקום האיבר המינימלי ולא ערכו, ואנו מסתמכים על כך שבמערך קל לברר את ערכו של איבר כאשר ידוע המציין שלו:

```
// חיפוש ערך מינימלי ושמירת ערך המציין שלו
for(int i = 2; i <= numOfSingers; i++)
    if (votes[i - 1] < votes[minSinger])
        minSinger = i;
Console.WriteLine("contestant number {0} is going home", minSinger);
```

סוף פתרון בעיה 5

להעמקה בתבנית ***מערך מונים*** פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 10.35

עקבו אחר התוכנית `IsraeliIdol` בעזרת טבלת מעקב, עבור הקלט `4 2 3 1 1 2 4 4 2 -1` (משמאל לימין).

שאלה 10.36

בכיתה נערכה הצבעה לנציג הכיתה למועצת תלמידים. כל תלמיד הקליד במחשב את בחירתו על פי המפתח הבא: עבור רותי יוקלד הערך 1, עבור אלי – הערך 2, עבור אביב – הערך 3, ועבור אופיר – הערך 4.

פתחו אלגוריתם המקבל כקלט את מספר התלמידים בכיתה ואת רשימת ההצבעות (הצבעה אחת מכל תלמיד בכיתה) ומציג כפלט את שם המועמד המנצח. ישמו את האלגוריתם בשפת C#.

שאלה 10.37

פתחו אלגוריתם הקולט מספר חיובי שלם, ואחר כך מדמה סדרת הטלות קובייה, שאורכה כערך המספר שנקלט. פלט האלגוריתם הוא מספר הפעמים שהוגרלה כל אחת משש התוצאות האפשריות להטלת קובייה. ישמו את האלגוריתם בשפת התכנות C#.

שאלה 10.38

ששת חברי קבוצה בצופים מצאו דרך חדשה להגרלה. הם מחלקים ביניהם את המספרים מ-1 עד 6, ומטילים קובייה עד אשר מספר כלשהו מוגרל פעמיים. החבר בעל המספר שהוגרל פעמיים הוא הזוכה. פתחו אלגוריתם אשר מדמה את תהליך זריקת הקובייה ומדפיס את המספר הזוכה. ישמו את האלגוריתם בשפת התכנות C#.

שאלה 10.39

בפתרון הבעיה הקודמת השתמשנו במערך מונים, אך בכל מהלך הביצוע ערכיו לא עברו אף פעם את הערך 2. שינוי אחד הערכים במערך ל-2 הביא לסיום האלגוריתם. שנו את הפתרון כך שבמקום מערך מטיפוס שלם, ישתמש במערך מטיפוס בוליאני. ישמו את הפתרון החדש בשפת C#.

שאלה 10.40

במשחק "תפוס את הצבע" כל שחקן בוחר צבע שונה. הקלט הוא מספר השחקנים והצבע שכל שחקן בחר. לאחר מכן עבור כל הקשה על מקש <Enter> מוגרל צבע כלשהו (מתוך הצבעים שנבחרו). המשחק מסתיים כאשר מקישים על האות Q. בסוף המשחק בודקים מהו הצבע שהוגרל הכי הרבה פעמים ומציגים את הצבע הזוכה כפלט. כתבו תוכנית שתדמה את "תפוס את הצבע".

הדרכה: מאתחלים מערך צבעים לפי הקלט שמתקבל מהשחקנים, לדוגמה עבור הקלט: 4, אדום, ירוק, צהוב, סגול, יתקבל המערך הבא:

סגול	צהוב	ירוק	אדום

את הצבעים יש להגריל מתוך הצבעים שנקלטו.

בנוסף, יש להגדיר מערך מונים לספירת ההגרלות לכל צבע.

10.5 מערך צוברים

באותו אופן שמימשנו מערך של מונים, ניתן לממש מערך של צוברים. במערך צוברים כל איבר הוא צובר בפני עצמו, אך יש קשר בין משימות הצבירה שהצוברים השונים במערך אחראים להן. לצורך פתרון השאלה הבאה יש להשתמש במערך צוברים, שהעבודה עמו מהווה אף היא תבנית שימושית.

שאלה 10.41

במשחק קלפים משתתפים ארבעה שחקנים ולהם מספרים סידוריים מ-1 עד 4. בכל סבב מוכרזים השחקנים שזכו במקום הראשון, במקום השני ובמקום השלישי. השחקן הנותר נחשב כמפסיד. המנצח במקום הראשון מקבל 7 נקודות, השחקן שבמקום השני מקבל 3 נקודות, השחקן שבמקום השלישי אינו מקבל נקודות והמפסיד **מאבד** 4 נקודות.

המנצח במשחק כולו הוא זה שקיבל את מרב הנקודות בסיום כל הסבבים.

פתחו אלגוריתם אשר מקבל כקלט את מספר הסבבים, ולאחר מכן מקבל עבור כל סבב את מספרי השחקנים שהגיעו למקום הראשון, השני והשלישי. האלגוריתם מציג כפלט את מספרו של השחקן המנצח במשחק. ישמו את האלגוריתם בשפת C#.

שימו ♥ : מאחר שניתן גם לאבד נקודות, ייתכן ששחקן יגיע למספר נקודות שלילי.

שאלה 10.42

בספריית הווידיאו יש לכל סרט קוד בן 4 ספרות. הספרה השמאלית ביותר מסמנת את קוד המחלקה (בין 1 ל-6), שתי הספרות האמצעיות מסמנות את קוד הסרט (בין 1 ל-99) והספרה הימנית ביותר מסמנת את מספר העותקים בספרייה (בין 1 ל-9).

פתחו אלגוריתם המקבל כקלט רשימה של הקודים של כל הסרטים בספרייה המסתיימת במספר 0. פלט האלגוריתם הוא:

א. קוד המחלקה שיש בה את מספר הסרטים (השונים) הגדול ביותר.

ב. קוד המחלקה שיש בה את מספר עותקי הסרטים הגדול ביותר.

ישמו את האלגוריתם בשפת C#.

שאלה 10.42 עוסקת במציאת איבר שכיח במערך. להעמקה בתבנית ***חישוב שכיח*** פנו לסעיף התבניות המופיע בסוף הפרק.

להעמקה בתבנית ***מערך צוברים*** פנו לסעיף התבניות המופיע בסוף הפרק.

10.6 יעילות מקום

בכל הבעיות שפתרנו בפרק זה נעזרנו במערך, ועבור כל בעיה ראינו את נחיצות השימוש במערך במהלך הפתרון. בסעיף הבא נתמקד באבחנה בין בעיות אשר בפתרונן נחוץ מערך ובין בעיות אשר ניתן לפתור גם ללא מערך, ונבין מדוע לא כדאי להשתמש במערך אם אין בו צורך.

בעיה 6

מטרת הבעיה ופתרונה: הצגת המדד של יעילות מקום, והשוואה של אלגוריתמים ביחס ליעילות מקום. מבט נוסף אל יעילות מבחינת זמן-ביצוע.

ערוצי הכבלים עורכים מדי פעם סקרים כדי לברר את אחוזי הצפייה בתוכניותיהם השונות. פרסום תוצאות סקר כזה כולל רשימת תוכניות יחד עם אחוז צפייה בכל תוכנית.

א. פתחו אלגוריתם אשר הקלט שלו הוא מספר של תוכניות (שלם וחיובי), ולאחר מכן רשימה של אחוזי צפייה בתוכניות. אורך הרשימה כערך הנתון הראשון. הפלט הוא מספר התוכניות שאחוז הצפייה בהן גבוה מממוצע אחוזי הצפייה.

ב. פתחו אלגוריתם אשר הקלט שלו הוא מספר של תוכניות (שלם וחיובי), ולאחר מכן רשימה של אחוזי צפייה בתוכניות. אורך הרשימה כערך הנתון הראשון. הפלט הוא מספר התוכניות שאחוז הצפייה בהן גבוה מ-20%.

ג. האם תשובתכם לסעיף א תשתנה אם ידוע כי רשימת אחוזי הצפייה נתונה בסדר יורד (כלומר תחילה נתון אחוז הצפייה בתוכנית האהודה ביותר, אחר-כך אחוז הצפייה התוכנית האהודה הבאה וכך הלאה, עד אחוז הצפייה בתוכנית שנצפית הכי פחות).

נתחיל בסעיף א.

הבעיה המוגדרת בסעיף א זהה לחלוטין לבעיה 2. יש לקלוט אורך של רשימת נתונים לא ריקה, לקלוט את הרשימה עצמה, לחשב את ממוצע הערכים, ולמנות את הערכים הגדולים מהממוצע. מאחר שכבר פתרנו בעיה זהה לה, לא נחזור כעת על הפתרון בפירוט, רק נזכיר שהפתרון משתמש במערך לשמירת ערכי הקלט (במקרה זה, רשימת אחוזי הצפייה).

נעבור כעת לסעיף ב.

ניתוח הבעיה בעזרת דוגמאות

נניח כי הקלט הוא 31 18 13 21 4. במקרה זה הפלט הוא 2, כי אחוז הצפייה בשתי תוכניות (הראשונה והרביעית) גבוה מ-20%. אין אנו נדרשים לחשב ממוצע של אחוזי הצפייה.

פירוק הבעיה לתת-משימות

ניתן להציג פירוק הדומה לפירוק המתאים לסעיף א:

1. קליטת מספר התוכניות
2. קליטת אחוזי הצפייה ושמירתם
3. השוואת כל אחוז צפייה ל-20 ומנייתו אם הוא גדול מ-20

פירוק זה אכן מורה על שימוש במערך. קודם כל נשמרים הנתונים, ואחר כך הם מעובדים (במקרה זה, מושווים ל-20 ונמנים בהתאם לתוצאת ההשוואה).

? האם ניתן להציג פירוק פשוט יותר, שאינו מחייב שימוש במערך?

כן! ניתן לעבד כל נתון נקלט מיד עם קליטתו, להשוותו ל-20, ולהחליט אם למנות אותו או לא, ובעצם אין צורך בשמירת ערכי הקלט לאחר מכן. לכן נקבל את הפירוק הבא לתת-משימות:

1. קליטת מספר התוכניות
2. קליטת אחוזי הצפיה, ועבור כל אחוז צפייה נקלט, השוואתו ל-20 ועדכון המונה בהתאם

בחירת משתנים

מאחר שכאמור אין צורך במערך נשתמש במשתנים הבאים:

numOfPrograms – מטיפוס שלם, ישמור את מספר התוכניות.

rate – מטיפוס ממשי, ישמור את אחוז הצפייה התורן בקלט.

above20Counter – מטיפוס שלם, מונה לשמירת מספר אחוזי הצפייה הגבוהים מ-20.

האלגוריתם

1. קלוט את מספר התוכניות ב-numOfPrograms
2. אתחל את ערכו של above20Counter ל-0
3. עבור כל i שלם מ-0 עד numOfPrograms בצע:
   3.1. קלוט את אחוז הצפייה התורן ב-rate
   3.2. אם rate > 20
      3.2.1. הגדל את ערכו של above20Counter ב-1
4. הצג את ערכו של above20Counter

אם כך, למרות שהבעיה המוגדרת בסעיף א והבעיה המוגדרת בסעיף ב נראות דומות, הרי לפתרון סעיף א יש צורך במערך ואילו לפתרון סעיף ב אין צורך במערך.

מדוע יש בכך חשיבות? כידוע, בעת ביצוע תוכנית אשר יש בה שימוש במערך, מוקצה שטח זיכרון המספיק עבור כל איברי המערך. מספר תאי הזיכרון המוקצים למערך הינו מרכיב משמעותי בקביעת גודל הזיכרון הכולל הדרוש לביצוע התוכנית. כיוון שמשאבי הזיכרון של המחשב מוגבלים, נשתדל לפתח אלגוריתמים שגודל המקום הדרוש לביצועם הוא מצומצם במידת האפשר. כלומר נשתדל לפתח אלגוריתמים יעילים עד כמה שניתן מבחינת מקום בזיכרון. זאת בנוסף להקפדה על יעילות מבחינת זמן-ביצוע, כפי שהוצג בפרק 8.

כאשר יש שני אלגוריתמים שונים לפתרון אותה בעיה, ובאחד מהם משתמשים במערך ובאחר אין משתמשים במערך, נאמר שהאלגוריתם האחר **יעיל יותר מבחינת מקום בזיכרון**.

נעבור לפתרון סעיף ג.

ניתוח הבעיה בעזרת דוגמאות

גם בסעיף זה יש לפתח אלגוריתם הפותר את הבעיה שהוצגה בסעיף א. אבל שלא כמו בסעיף א, נתון לנו מאפיין נוסף של הקלט: רשימת אחוזי הצפייה נתונה בסדר יורד.

הנה קלט לדוגמה העונה להגדרת הבעיה: 3 15 29 30 41 5.

ממוצע הערכים הנתונים הוא 23.6. הערכים הגבוהים מן הממוצע הם 41, 30 ו-29. ניתן לראות כי מאחר שרשימת הערכים נתונה בסדר יורד, כל הערכים הגבוהים מן הממוצע נמצאים בתחילת הרשימה.

בדיוק כמו בסעיף א, את ההשוואה לממוצע ניתן לבצע רק לאחר חישוב הממוצע, כלומר רק לאחר קליטת כל הערכים.

❓ האם ניתן לפתור סעיף זה ללא מערך?

לא. מאחר שההשוואה לממוצע יכולה להיעשות רק אחרי סיום הקלט, יש לשמור את כל הערכים הנקלטים, ולשם כך נחוץ מערך.

❓ האם נוכל בכל זאת לשפר את הפתרון בשימוש במאפיין הקלט הנוסף?

בעזרת דוגמת הקלט שניתחנו ראינו כי הערכים הגדולים מן הממוצע נמצאים כולם בתחילת הרשימה, ולכן גם יישמרו ראשונים בשלב שמירת הקלט. לכן, בשלב ההשוואה לממוצע לא נצטרך לעבד שוב את כל ערכי הקלט, אלא רק את אלה המופיעים בהתחלה.

לכן, במקום להשתמש בלולאת **for** שבה מספר הפעמים ידוע מראש שתעבור על **כל** האיברים שנשמרו, נוכל להשתמש בלולאת **while**. תנאי הכניסה ללולאה יאפשר את המשך ביצוע הלולאה כל עוד האיבר התורן גדול מהממוצע.

ניתוח זה מראה כי הפתרון שניתן לא יהיה יעיל יותר מהפתרון של סעיף א מבחינת מקום בזיכרון, אבל יהיה בו שיפור מבחינת יעילות זמן: מניית הערכים הגדולים מהממוצע תעבור רק על חלק מהאיברים שנשמרו ולא על כולם.

פירוק הבעיה לתת-משימות

בהתאם לניתוח שערכנו, נפרק את הבעיה באופן הבא:

1. קליטת מספר התוכניות
2. קליטת אחוזי הצפייה, שמירתם וצבירתם
3. חישוב הממוצע
4. מעבר על הערכים הגדולים מן הממוצע תוך כדי מנייתם
5. הצגת ערך המונה

בחירת משתנים

numOfPrograms – מטיפוס שלם, לשמירת מספר התוכניות

rating – מערך מטיפוס ממשי בן numOfPrograms איברים לשמירת אחוזי הצפייה הנתונים, והם יהיו מסודרים בו בסדר יורד

sumOfRating – מטיפוס ממשי, צובר שישמור את סכום אחוזי הצפייה

averageOfRating – מטיפוס ממשי, לשמירת ממוצע אחוזי הצפייה

aboveAverageCounter – מטיפוס שלם, מונה את אחוזי הצפייה הגבוהים מאחוז הצפייה הממוצע.

התוכנית המלאה

```
/*
קלט: מספר של תוכניות (שלם חיובי)
ורשימת אחוזי צפייה בתוכניות, נתונים בסדר יורד
פלט: מספר אחוזי הצפייה הגבוהים מאחוז הצפייה הממוצע
*/
using System;
public class AboveAverage
{
    public static void Main()
    {
        // הגדרת משתנים
        int numOfPrograms;            // מספר התוכניות
        double[] rating;              // מערך אחוזי הצפייה
        double sumOfRating = 0;       // צובר אחוזי הצפייה
        double averageOfRating ;      // ממוצע אחוזי הצפייה
        int aboveAverageCounter = 0;  // מונה הגבוהים מהממוצע
        int i;
        // קלט וצבירה
        Console.Write("Enter number of programs: ");
        numOfPrograms = int.Parse(Console.ReadLine());
        rating = new double[numOfPrograms];
```

```
        for (i = 0; i < numOfPrograms; i++)
        {
            Console.Write("Enter rating percentage: ");
            rating[i] = int.Parse(Console.ReadLine());
            sumOfRating = sumOfRating + rating[i];
        } // for
        // חישוב ממוצע
        averageOfRating = sumOfRating / numOfPrograms;
        // מניית הגבוהים מהממוצע
        i = 0;
        while(rating[i] > averageOfRating)
        {
            i++;
            aboveAverageCounter++;
        } // while
        // פלט
        Console.WriteLine("The rating of {0} programs is above " +
                                        "average", aboveAverageCounter);
        } // Main
} //class AboveAverage
```

שימו ♥ העובדה שאחוזי הצפייה מסודרים בסדר יורד מאפשרת לסיים את לולאת ה-**while** מבלי לעבור על כל איברי המערך. התנאי `rating[i] > averageOfRating` משמש כתנאי כניסה ללולאה ולכן המנייה מסתיימת ברגע שנמצא אחוז צפייה שאינו גבוה מהממוצע.

שאלה 10.43

מדוע מובטח כי ההוראה לביצוע-חוזר-בתנאי, המונה את הערכים הגבוהים מהממוצע, אינה לולאה אינסופית (במילים אחרות מדוע מובטח כי תנאי הכניסה יתקיים בשלב כלשהו)?

סוף פתרון בעיה 6

מפתרון בעיה 6 למדנו לקח חשוב:

יש לנצל עד כמה שניתן את מאפייני הקלט-פלט כדי לנסח אלגוריתם יעיל, הן מבחינת מקום בזיכרון והן מבחינת זמן-ביצוע. ייתכנו שתי בעיות אלגוריתמיות הנראות דומות, לפחות במבט ראשון, ופתרונות יעילים עבורן נבדלים זה מזה הן מבחינת מקום והן מבחינת זמן. בפרט, ייתכן שעבור פתרון בעיה אחת נחוץ מערך ועבור פתרון הבעיה האחרת לא נחוץ מערך.

שאלה 10.44

נתונה רשימת קלט של 40 ציונים. סמנו אם לביצוע כל אחד מהחישובים הבאים נחוץ מערך או לא:

א. מספר הציונים הגבוהים מ-80 .... נחוץ / לא נחוץ
ב. מספר הציונים השווים לציון הראשון ברשימה .... נחוץ / לא נחוץ
ג. מספר הציונים הגבוהים מן הציון האחרון ברשימה .... נחוץ / לא נחוץ
ד. מספר הציונים הנמוכים מן הציון הלפני אחרון ברשימה .... נחוץ / לא נחוץ
ה. מספר הציונים הנמוכים מן הציון הראשון ומן הציון האחרון .... נחוץ / לא נחוץ
ו. מספר הציונים הנמוכים מממוצע הציונים .... נחוץ / לא נחוץ
ז. מספר הציונים הגבוהים מהממוצע של שני הציונים הראשונים .... נחוץ / לא נחוץ

ח. מספר הציונים הגבוהים מהממוצע של שני הציונים האחרונים .... נחוץ / לא נחוץ

שאלה 10.45

בבעיה 3 בפרק, הקלט כלל רשימה של 30 תווים ('T' או 'F') אשר ציינו מצב לוח, ואחר כך מספר שציין מקום של שחקן על הלוח. בפתרון הבעיה נעשה שימוש במערך בן 30 איברים לשמירת מצב הלוח.

נניח שמקום השחקן על הלוח יינתן כקלט **לפני** הרשימה המתארת את מצב הלוח (ולא אחריה). האם גם במקרה זה נחוץ מערך? הסבירו את תשובתכם.

שאלה 10.46

עודד המציא משחק חדש. הוא מציג לפני חבריו קופסת מטבעות סגורה. כל אחד מחבריו רושם ניחוש של מספר המטבעות בקופסה. לאחר מכן עודד סופר את מספר המטבעות בקופסה. המטבעות מחולקים לכל מי שניחש את המספר המדויק של המטבעות. שארית המטבעות נשארת אצל עודד.

א. פתחו אלגוריתם אשר מקבל כקלט מספר המציין את מספר החברים המשתתפים במשחק, אחר כך את רשימת הניחושים, ניחוש עבור כל חבר המשתתף במשחק, ולבסוף את מספר המטבעות שבקופסה. פלט האלגוריתם הוא מספריהם הסידוריים ברשימה של החברים שניחשו את מספר המטבעות המדויק, וכן מספר מטבעות שיקבל כל זוכה.
ישמו את האלגוריתם בשפת C#.
שימו ♥: היזהרו מחלוקה ב-0!

ב. ענו על סעיף א כאשר סדר נתוני הקלט שונה: תחילה נתון מספר המטבעות שבקופסה, אחריו מספר המשתתפים במשחק, ולבסוף רשימת הניחושים.

שאלה 10.47

בכספת של בנק שמורים יהלומים על מדפים הממוספרים מ-1 עד 100.

נתונה כקלט רשימת ערכים המבטאים את מצב מדפי הכספת, כך שהערך ה-i ברשימה מבטא את מצב המדף ה-i (מכיל יהלומים או לא).

ציינו עבור כל אחד מהחישובים הבאים אם נחוץ מערך לצורך ביצועו או לא. נמקו את תשובותיכם.

א. מספר המדפים המכילים יהלומים, ומספר המדפים ללא יהלומים.................. נחוץ / לא נחוץ
נימוק: ____________________

ב. מספר המדפים שמצבם שווה למצב המדף הראשון .... נחוץ / לא נחוץ
נימוק: ____________________

ג. מספר המדפים שמצבם שווה למצב המדף האחרון .... נחוץ / לא נחוץ
נימוק: ____________________

ד. מספרם הסידורי של המדפים שמצבם שווה למצב המדף הראשון .... נחוץ / לא נחוץ
נימוק: ____________________

ה. מספרם הסידורי של המדפים שמצבם שווה למצב המדף האחרון .... נחוץ / לא נחוץ
נימוק: ____________________

בעזרת החומר שלמדנו בפרק 10 ניתן לפתור גם בעיות המשתמשות בתבניות ***הזזה מעגלית בסדרה***, ***הזזה של תת-סדרה*** ו***היפוך של תת-סדרה***. להעמקה בתבניות אלו פנו לסעיף התבניות המופיע בסוף הפרק.

שאלות נוספות

1. במשחק כדורסל ניתן לקלוע סל המזכה את הקבוצה בנקודה אחת, בשתי נקודות או בשלוש נקודות. בכל קבוצה חמישה שחקנים אשר ממוספרים מ-1 עד 5. בעת משחק מוזנות תוצאות הקליעות בזוגות מספרים: מספר השחקן הקולע ומספר הנקודות שקיבל עבור הקליעה. בסיום המשחק מוקלד 0 עבור מספר השחקן. מנהל הקבוצה מעוניין לערוך בדיקות סטטיסטיות על הישגי הקבוצה בעת משחק:
   א. מה מספר השחקן או השחקנים שצברו את מספר הנקודות הגדול ביותר?
   ב. כמה נקודות צברה הקבוצה במשך כל המשחק?
   ג. מה סוג הקליעה הנפוץ ביותר (קליעות שזיכו בנקודה אחת, ב-2 נקודות או ב-3 נקודות)
   ד. איזה שחקן או שחקנים לא צברו שום נקודה במשך כל המשחק?

   לדוגמה: הקלט (משמאל לימין) 0 3 2 2 2 5 1 3 2 2 2 מתאר כי שחקן מספר 2 קלע סל של 2 נקודות, שחקן מספר 3 קלע סל של נקודה 1, שחקן מספר 5 קלע סל של 2 נקודות ושחקן מספר 2 קלע סל של 3 נקודות. הפלט עבור קלט זה הוא: "שחקן מספר 2 צבר את מספר הנקודות הגדול ביותר, במשך כל המשחק הקבוצה צברה 8 נקודות, סוג הקליעה הנפוץ ביותר הוא 2, ושחקנים מספר 1 ו-4 לא צברו אף נקודה במשך המשחק"

   ? חשבו לגבי כל סעיף אם אתם זקוקים למערך מונים, למערך צוברים או שאינכם זקוקים כלל למערך.

2. נתון המערך `arr` ובו ערכים שלמים ונתון הקטע הבא המשתמש במערך מונים:

```
int[] counts = new int[2];
int element;
for (i = 0; i < arr.Length; i++)
{
    element = arr[i];
    counts[element % 2]++;
}
for (i = 0; i < counts.Length; i++)
    Console.WriteLine(counts[i]);
```

   שימו ♥: קטע תוכנית זה משתמש במערך מונים.
   א. מה יוצג כפלט עבור המערך `arr` הבא:

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	Arr[7]
90	68	198	5	11	34	89	6

   ב. תנו דוגמה למערך `arr` בגודל 10, שעבורו יוצגו הערכים: 3 7.
   ג. מהי מטרת קטע התוכנית?

3. כתבו אלגוריתם שהקלט שלו הוא סדרת תווים המסתיימת בתו '*' והפלט שלו הוא מספר המופעים של כל אחת מאותיות הא"ב האנגלי הקטנות.
   שימו ♥: לצורך כתיבת האלגוריתם, עליכם להשתמש במערך מונים.
   א. מהו הביטוי החשבוני שהשתמשתם בו להתאמת אותיות הא"ב למצייני המערך?
   ב. ישמו את האלגוריתם בשפת C#.

4. במוסד לביטוח רפואי נעשה מחקר על צריכת 150 תרופות בידי החולים המבוטחים. לכל תרופה יש מספר סידורי בין 1 ל-150.

א. פתחו אלגוריתם שהקלט שלו הוא סדרת שלשות של ערכים, וכל שלשה מייצגת גיל של מבוטח, מספר סידורי של התרופה שנצרכה ואת הכמות שלה (מספר יחידות). סדרת הקלט מסתיימת עם קליטת הזקיף 1- כגיל המבוטח. הפלט של האלגוריתם הוא קבוצת הגילאים הצורכת את כמות התרופות הכוללת הגדולה ביותר, מבין 4 קבוצות הגילאים הבאות: קבוצת בני 0 עד 10, קבוצת בני 11 עד 30, קבוצת בני 31 עד 50 וקבוצת בני 51 ומעלה. ייתכן שיש יותר מקבוצת גיל אחת הצורכת את כמות התרופות הכוללת הגדולה ביותר. ישמו את האלגוריתם בשפת C#.

שימו ♥: לצורך פתרון סעיף זה אין צורך להבחין בין סוגי התרופות השונים.

ב. הרחיבו את האלגוריתם כך שיוצגו כפלט גם מספרי התרופות שלא נצרכו כלל.

ג. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד שילבתם ביניהן.

5. יותם המדריך החליט לארגן לחניכיו פעולה בסגנון "חפש את המטמון". המטמון הוחבא בבניין שבט הצופים שבו 50 חדרים הממוספרים מ-0 עד 49. בכל חדר נמצא פתק שבו רשום מספר החדר הבא שבו יש להמשיך את החיפוש או נמצא המטמון עצמו. יותם הכין את הפתקים ואת המטמון מבעוד מועד ורצה לפזר אותם בחדרים. לשם כך הוא הכין מערך בשם rooms בגודל 50 תאים. בכל תא במערך רשום מספר בין 1- ל-49, המספר 1- מציין שהמטמון נמצא בחדר זה. כל מספר אחר מכוון לחדר שיש לעבור אליו להמשך החיפוש. כמו כן יותם אומר לחניכיו את מספר החדר הראשון שיש להתחיל ממנו את החיפושים. לדוגמה: עבור 16 חדרים וחיפוש המתחיל בחדר מספר 3:

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
rooms	14	11	10	8	-1	6	-1	2	0	13	4	-1	15	1	7	5

הקבוצה תתחיל בחיפוש בחדר 3, תעבור לחדר 8, משם לחדר 0, וכך הלאה עד אשר תמצא את המטמון בחדר מספר 4. כתבו קטע תוכנית המשתמש במערך rooms ובמספר החדר שבו יש להתחיל את החיפוש ומציג כפלט את מיקומו של המטמון. ניתן להניח שיותם פיזר את הפתקים בצורה כזו שהאלגוריתם מסתיים. יש לסרוק את המערך לפי סדר החיפוש שהכתיב יותם.

סיכום

בפרק זה הוצגו בעיות שפתרונן מצריך שמירה של סדרת ערכים מאותו טיפוס. ניתן לשמור סדרה כזאת כמערך (array).

מערך הוא אוסף סדור של משתנים מאותו טיפוס, הקשורים זה לזה ולהם שם משותף.

איבר במערך הוא משתנה לכל דבר, כלומר ניתן לשמור בו ערכים ולקרוא את הערכים השמורים בו.

שמו של איבר במערך מורכב משם המערך וממספרו הסידורי של האיבר במערך. למשל arr[3]. במקרה זה, שם המערך הוא arr ו-3 הוא **המציין** (אינדקס) שמפנה אותנו לאיבר שנמצא במקום מספר 3 במערך.

באלגוריתמים רבים נוח לבצע **סריקה של מערך**, תוך עיבוד איבריו, באמצעות לולאה.

אם נדרשת **סריקה מלאה** של כל איברי המערך, הרי מספר הסיבובים ידוע מראש ושווה לאורך המערך. גם סריקה לא רציפה, אבל במרווחים ידועים מראש, ניתנת לביצוע באמצעות לולאה שמספר הסיבובים בה ידוע מראש.

אם הסריקה מתבצעת בדילוגים שאינם קבועים ואינה רציפה, או אם ביצועה תלוי בתנאי, נשתמש בהוראה לביצוע-חוזר-בתנאי.

בטבלת מעקב אחר מהלך ביצוע תוכנית הכוללת מערך נכלול עמודות נפרדות עבור איברים שונים של המערך.

מערך יכול לתפקד כ**מערך מונים** או כ**מערך צוברים**, במקרים שיש צורך לתפעל במקביל כמה מונים או צוברים בעלי אופי משותף.

בעת הרצת תוכנית המשתמשת במערך, מוקצים תאים עבור כל איבר במערך. כיוון שמשאבי הזיכרון של המחשב מוגבלים, נשתדל לפתח אלגוריתמים אשר גודל המקום הדרוש לביצועם הוא מצומצם ככל האפשר.

ייתכנו שני אלגוריתמים שונים לפתרון אותה בעיה, אשר באחד יש שימוש במערך ובשני אין. נאמר שהאלגוריתם השני יותר **יעיל מבחינת מקום.**

יש לנצל עד כמה שניתן את מאפייני הקלט-פלט כדי לנסח אלגוריתם יעיל, הן מבחינת מקום בזיכרון והן מבחינת זמן-ביצוע.

ייתכנו שתי בעיות אלגוריתמיות הנראות דומות, לפחות במבט ראשון, אשר פתרונות יעילים עבורן נבדלים זה מזה הן מבחינת מקום והן מבחינת זמן. בפרט, יתכן שעבור פתרון בעיה אחת נחוץ מערך ועבור פתרון הבעיה האחרת לא נחוץ מערך.

סיכום מרכיבי שפת #C שנלמדו בפרק 10

הצהרה על מערך בשפת #C נכתבת כך:

```
טיפוס[] שם המערך;
```

ראשית שם הטיפוס, אחריו סוגריים מרובעים ואז שמו של המערך.

לפני שימוש במערך מוצהר יש לבצע עבורו **הקצאת שטח זיכרון**. הקצאת שטח למערך מתבצעת באמצעות הפעולה **new** שאחריה מופיע טיפוס איברי המערך, ואחריו מופיע בסוגריים מרובעים מספר איברי המערך:

```
new טיפוס[מספר הערכים];
```

ניתן לצרף את ההצהרה ואת ההקצאה להוראה אחת:

```
טיפוס[] שם המערך = new טיפוס[מספר הערכים];
```

בשפת #C כאשר מוקצה שטח זיכרון עבור מערך של איברים מטיפוס פשוט, מתבצע גם **אתחול אוטומטי** של כל איבריו. איברי מערך מספריים (שלמים או ממשיים) יאותחלו ב-0, איברי מערך בוליאני יאותחלו ב-**false**, ואיברי מערך תווי יאותחלו בתו הריק.

בשפת #C **מספור האיברים במערך** מתחיל ב-0, ולכן `arr[0]` מפנה אותנו לאיבר הראשון במערך, ו-`arr[3]` מפנה אותנו לאיבר הרביעי במערך.

לכל מערך מוגדרת **תכונת אורך** בשם `Length` השומרת את אורכו. תכונה זו מקבלת את ערכה בעת הקצאת שטח זיכרון עבור המערך ואינה ניתנת לשינוי לאחר מכן. הפנייה לתכונה נעשית באמצעות סימון הנקודה, למשל כך: `arr.Length`. אין להוסיף סוגריים משום שזו אינה פעולה!

יש להקפיד על פנייה לאיבר באמצעות מציין שערכו איננו חורג מתחום הערכים המותר, כלומר נע בין 0 לבין אורך המערך פחות אחת. **חריגה מגבולות המערך** תגרום לעצירת התוכנית עקב שגיאת ריצה.

סריקת איברי מערך יכולה להיעשות באמצעות לולאת **`for`**, אם ידועות מראש נקודות ההתחלה והסיום של הסריקה, וידועים מרווחי הדילוג (עבור סריקה לא רציפה). בלולאה כזאת, משתנה הבקרה של הלולאה משמש גם כמציין לציון מיקומו של האיבר הנסרק במערך. משתנה הבקרה מאותחל בתחילת הלולאה בערך של המציין של האיבר שהסריקה מתחילה ממנו. ערכו מתקדם בכל צעד בערך הרצוי (לסריקה רציפה הוא מתקדם ב-1 בכל פעם) עד שיגיע למיקומו של האיבר שהסריקה מסתיימת בו .

לסריקה רציפה של כל איברי המערך ניתן להיעזר בתכונת האורך כך שערכי משתני הבקרה ינועו מ-0 עד לאורך המערך פחות אחת, ויגדלו ב-1 בכל סיבוב.

לסריקה שאינה רציפה ובמרווחים שאינם ידועים מראש, או לסריקה התלויה בתנאי, יש להשתמש בלולאת **`while`**.

תבניות – פרק 10

פירוט מלא של התבניות ושל שאלות שבפתרונן יש שימוש בתבניות ניתן למצוא באתר הספר ברשת האינטרנט.

מערך מונים

בחרנו להציג לפניכם שימוש בביצוע-חוזר-בתנאי בתבנית **מערך מונים**, ושימוש בביצוע-חוזר מספר פעמים ידוע מראש בתבנית **מערך צוברים**. אפשר כמובן להחיל את המקרה האחד על האחר ולהיפך בשינוי קל.

שם התבנית: **מערך מונים**

נקודת מוצא: תנאי סיום toEnd, סדרת ערכים, ביטוי חשבוני whichCounter המקשר בין ערך בסדרה למציין של המערך

מטרה: בניית מערך מונים עבור ערכי הסדרה, בעוד משך הבנייה תלוי בביטוי toEnd.

אלגוריתם:

1. אתחל את איברי המערך countElements ב-0
2. השם את הערך הבא בסדרה ב-element
3. **כל עוד** לא מתקיים התנאי toEnd **בצע**:
   3.1. הגדל את countElements[whichCounter] ב-1
   3.2. השם את הערך הבא בסדרה ב-element

מערך צוברים

שם התבנית: **מערך צוברים**

נקודת מוצא: אורך סדרת הערכים limit, סדרת ערכים, ביטוי חשבוני whichSum המקשר בין ערך בסדרה למציין של המערך

מטרה: בניית מערך צוברים עבור ערכי הסדרה שאורכה limit

אלגוריתם:

1. אתחל את ערכי מערך הצוברים
2. בצע limit פעמים:
   2.1. השם את הערך הבא בסדרה ב-element
   2.2. השם את הערך לצבירה ב-value
   2.3. הוסף ל-sumElements[whichSum] את ערכו של value

חישוב שכיח

שם התבנית: **חישוב שכיח**

נקודת מוצא: אורך סדרת הערכים limit, סדרת ערכים, ביטוי חשבוני whichSum המקשר בין ערך בסדרה למציין של המערך

מטרה: הצגה כפלט של הערך השכיח או של הערכים השכיחים בסדרת הקלט שאורכה הוא limit

אלגוריתם:

1. אתחל את ערכי מערך המונים
2. בצע limit פעמים:
   2.1. השם את הערך הבא בסדרה ב-element
   2.2. הגדל את countElements[whichCount] ב-1
3. אתחל את max בערכו של countElements[0]
4. עבור כל i שלם בתחום מ-1 עד אורך המערך countElements פחות 1 בצע:
   4.1. אם countElements[i] > max
      4.1.1. השם ב-max את הערך של countElements[i]
5. עבור כל i שלם בתחום מ-0 עד אורך המערך countElements פחות 1 בצע:
   5.1. אם ערכו של countElements[i] שווה ל-max
      5.1.1. הצג את i כפלט

הזזה מעגלית בסדרה

שם התבנית: **הזזה מעגלית שמאלה בסדרה**
נקודת מוצא: סדרת ערכים במערך elements שאורכו length
מטרה: הזזה מעגלית שמאלה של ערכי המערך
אלגוריתם:
1. *השם ב-temp את* elements[0]
2. *עבור כל* i *שלם בתחום 0 עד* length-2 *בצע:*
   2.1. *השם ב-*elements[i] *את הערך של* elements[i+1]
3. *השם ב-*elements[length-1] *את הערך של* temp

שם התבנית: **הזזה מעגלית ימינה בסדרה**
נקודת מוצא: סדרת ערכים במערך elements שאורכו length
מטרה: הזזה מעגלית ימינה של ערכי המערך
אלגוריתם:
1. *השם ב-temp את* elements[length-1]
2. *עבור כל* i *שלם בתחום* length-1 *עד 1 (בסדר יורד) בצע:*
   2.1. *השם ב-*elements[i] *את הערך של* elements[i-1]
3. *השם ב-*elements[0] *את הערך של* temp

הזזה של תת-סדרה

שם התבנית: **הזזה של תת-סדרה שמאלה**
נקודת מוצא: סדרת ערכים במערך elements באורך length, מקום k במערך ($0 \le k < length-1$)
מטרה: הזזה שמאלה של התת-סדרה הנמצאת במקומות k+1..length-1 למקומות k..length-2
אלגוריתם:
1. *עבור כל* i *שלם בתחום מ-*k *עד* length-2 *בצע:*
   1.1. *השם ב-*elements[i] *את הערך של* elements[i+1]

שם התבנית: **הזזה של תת-סדרה ימינה**
נקודת מוצא: סדרת ערכים במערך elements באורך length, מקום k במערך ($0 < k \le length-1$)
מטרה: הזזה ימינה של התת-סדרה הנמצאת במקומות k-1 .. 0 למקומות k .. 1
אלגוריתם:
1. *עבור כל* i *שלם בתחום מ-*k *עד 1 (בסדר יורד) בצע:*
   1.1. *השם ב-*elements[i] *את הערך של* elements[i-1]

היפוך סדר האיברים בסדרה

שם התבנית: **היפוך סדר האיברים בסדרה**
נקודת מוצא: סדרת ערכים במערך elements
מטרה: היפוך סדר הערכים במערך שאורכו length
אלגוריתם:

1. השם ב-limit את מנת החלוקה של length פריטים לשתי קבוצות
2. עבור כל שלם i בתחום מ-0 עד limit-1 בצע:
2.1. החלף את הערכים של elements[i] ושל elements[length-i-1]

פרק 11 – מחלקות ועצמים: הרחבה והעמקה

בפרקים הקודמים ראינו כי בשפת #C ניתן להשתמש בטיפוסים מורכבים הנקראים מחלקות (**classes**). משתנים מטיפוסים אלה נקראים עצמים (**objects**). בפרק 9 ראינו שימוש בעצמים מהמחלקה **string** המוגדרת בשפת #C.

בפרק זה נלמד כיצד להגדיל את אוצר הטיפוסים שעומדים לרשותנו, כלומר כיצד להגדיר מחלקות חדשות בעצמנו ולהשתמש בעצמים ממחלקות אלו.

11.1 מחלקה - הגדרה ושימוש

בעיה 1

מטרת הבעיה ופתרונה: הכרת המושגים מחלקה, תכונות, פעולות ועצמים. הגדרת מחלקה בסיסית, יצירת עצם ושימוש בו.

> אפשר לייצג זמן באמצעות שעות ודקות. כתבו תוכנית הקולטת מהמשתמש זמן בשני ערכים: שעה (מספר בין 0 ל-23) ודקה (מספר בין 0 ל-59) ומציגה את הזמן בפורמט סטנדרטי, למשל עבור השעה אחת עשרה ועשרה הפלט יהיה: 11:10.

השאלה עוסקת בזמן. אפשר להסתכל על כל זמן כעל ישות המתאפיינת באמצעות שני נתונים – אחד מייצג את השעה והשני את הדקה. אם כך, ניתן להגדיר טיפוס חדש (מחלקה) בשם "זמן" אשר יאגד בתוכו את תכונות הזמן. לאחר שנגדיר את הטיפוס "זמן" נוכל ליצור עצמים (הנקראים גם מופעים) המייצגים זמנים שונים. למופעים של הטיפוס החדש, כלומר לכל עצם מהטיפוס זמן, יהיו התכונות שעה ודקה אשר מתארות זמן מסוים.

בנוסף להגדרת התכונות (**attributes**), אנו צריכים להגדיר את הפעולות (**methods**) שנרצה להפעיל על עצמים מסוג זמן. מתוך תיאור הבעיה אנו מבינים שאנחנו זקוקים לפעולה המעדכנת את הזמן לפי ערכים המתקבלים מהמשתמש, כלומר אנחנו נרצה לעדכן את ערכי תכונות הזמן כך שכל עצם מסוג זמן ייצג זמן מסוים כלשהו כגון: שמונה ועשרה, שתים עשרה ארבעים וחמש וכו'. פעולה נוספת המתבקשת מתיאור הבעיה היא פעולה המחזירה את הייצוג הסטנדרטי של הזמן. כלומר בהינתן עצם מסוג זמן נרצה לקבל ממנו מחרוזת בפורמט המבוקש, למשל 8:10 או 12:45.

לצורך פתרון הבעיה, נגדיר תחילה את הטיפוס החדש, את המחלקה *זמן* ולאחר מכן נמשיך בשלבי פיתוח הפתרון.

הגדרת המחלקה *זמן*

כפי שראינו, לעצמים יש תכונות המאפיינות אותם ופעולות השייכות להם. בבואנו להגדיר מחלקה חדשה עלינו להגדיר את התכונות ואת הפעולות שתהיינה לעצמים מהמחלקה.

הגדרת התכונות

כדי להגדיר את התכונות של זמן, עלינו לקבוע מהו הטיפוס המתאים לכל תכונה. שעה היא תכונה שערכה יכול להיות מספר שלם בין 0 ל-23, ודקה היא תכונה שערכה יכול להיות מספר שלם בין 0 ל-59. לכן נגדיר אותן מטיפוס שלם.

בהתאם להחלטה על הטיפוסים נבחר משתנים לייצוג התכונות.

בחירת משתנים לייצוג התכונות

- **hour** – משתנה מטיפוס שלם המייצג את מרכיב השעות בזמן.
- **minute** – משתנה מטיפוס שלם המייצג את מרכיב הדקות בזמן.

הגדרת הפעולות

כאמור, על פי הגדרת הבעיה, אפשר לבצע על זמן את הפעולות הבאות.

- **עדכון הזמן** – פעולה זו מקבלת שני פרמטרים: אחד מייצג את השעות והשני את הדקות, והיא מעדכנת את רכיבי הזמן בהתאם.
- **החזרת הייצוג הסטנדרטי של הזמן** – פעולה זו מחזירה מחרוזת המייצגת את הזמן. אפשר לבנות את המחרוזת באמצעות שרשור התכונות עם סימן של נקודתיים ביניהן:

```
hour + ":" + minute
```

הגדרנו את תכונות הזמן ואת פעולותיו כפי שנובעות מהגדרת הבעיה. נראה כעת כיצד לממש הגדרות אלו בשפת C#.

מימוש המחלקה

כזכור מהאופן שאנו מגדירים את המחלקה הראשית, הגדרת מחלקה בשפת C# מתחילה תמיד במילה השמורה `class` ולאחר מכן נכתב שם המחלקה. בשפת C# מקובל להתחיל שם מחלקה באות גדולה. פרטי ההגדרה מופיעים בתוך סוגריים מסולסלים.

אם כן, כך נראית מסגרת הגדרת מחלקה בשם `Time`:

```
public class Time
{
        // גוף המחלקה //
} // class Time
```

מאפיין הגישה **public** מציין שהמחלקה היא ציבורית ולכן ניתנת לשימוש מכל מחלקה אחרת. וכך מתוך הפעולה הראשית (הנמצאת במחלקה אחרת) נוכל ליצור עצמים מסוג "זמן" ולהשתמש בהם.

בתוך תחום ההגדרה, כלומר בתוך הסוגריים המסולסלים, נפרט את המרכיבים השונים: תכונות ופעולות.

מימוש התכונות:

```
public class Time
{
    private int hour;          //מכיל את השעות
    private int minute;        //מכיל את הדקות
} // class Time
```

הגדרה של תכונה נעשית בדומה להצהרה על משתנה רגיל: שם הטיפוס (למשל **int**) ואחריו שם התכונה (למשל, `hour`). המילה השמורה **private**, המופיעה בתחילת כל הצהרה מציינת כי התכונה היא פרטית, ואין אליה גישה ישירה מחוץ למחלקה. לכן מתוך הפעולה הראשית (המכונה Main) לא נוכל להתייחס ישירות לתכונת השעות ולתכונת הדקות של עצם מסוג *זמן*. ניסיון להתייחסות ישירה כזאת גורם לשגיאת הידור. לעומת זאת, **פעולות** של זמן המוגדרות בתוך תחום המחלקה *זמן* יכולות להתייחס **לתכונות** של הזמן כאל משתנים לכל דבר. מכך נובע שכל התייחסות לתכונות של עצם נעשית אך ורק דרך הפעולות של אותו העצם.

אתחול התכונות:

בדומה למשתנה רגיל, אפשר גם לספק ערכים התחלתיים לתכונות, למשל:

```
private int hour = 8;
private int minute = 0;
```

במקרה זה ערכו ההתחלתי של עצם מסוג זמן יהיה השעה שמונה. לאחר מכן נוכל להשתמש בפעולת העדכון כדי לשנות את הזמן. במקרה שלא מאתחלים את התכונות במפורש, שפת C# מאתחלת תכונות מטיפוס מספרי בערך אפס, תווים מאותחלים בתו מיוחד המייצג תו ריק, תכונות בוליאניות בערך **false** ומערכים ועצמים בערך מיוחד **null**.

ההצהרות על התכונות יכולות להופיע בכל מקום בתוך תחום המחלקה. אנו ננהג לרשום את ההצהרות בראש המחלקה ואחריהן נרשום את הפעולות.

מימוש הפעולות:

נפנה כעת למימוש הפעולות. נתחיל בפעולה לעדכון הזמן המקבלת את השעה ואת הדקה של עצם מסוים ומעדכנת את תכונותיו. מימוש הפעולה `SetTime` בשפת C# נעשה כך:

```
public void SetTime(int h, int m)
{
    hour = h;
    minute = m;
}
```

נסביר את מרכיבי הפעולה.

השורה הראשונה היא **כותרת הפעולה**. היא מהווה למעשה את תעודת הזהות של הפעולה, ומציגה לפני המשתמש את כל המידע הדרוש לצורך שימוש בפעולה. למעשה, כבר ראינו כותרות של פעולות: בפרק 9, הכרנו כמה פעולות של עצמים מסוג **string**, וכדי שנדע כיצד להשתמש בפעולות אלו הצגנו בטבלה עבור כל פעולה את שורת הכותרת שלה. שורה זו גילתה לנו פרטים חשובים: את שם הפעולה, אילו פרמטרים היא מצפה לקבל ואיזה סוג ערך היא מחזירה. כותרת הפעולה `SetTime` מודיעה כי היא מצפה לקבל שני פרמטרים מסוג **int** הנקראים `h` ו-`m` (כפי שמציינים הסוגריים), ושהיא אינה מחזירה ערך כלשהו (כך מציינת המילה **void**). המילה השמורה **public** בתחילת שורת הכותרת מציינת כי הפעולה `SetTime` מוגדרת כפעולה ציבורית של עצם מסוג *זמן*. פעולה ציבורית של עצם היא פעולה שאפשר להפעיל על העצם, ולזמן אותה מכל מקום בתוכנית. כך נוכל מתוך הפעולה הראשית להפעיל את הפעולה `SetTime` על עצמים מסוג *זמן*, כפי שנדגים מיד.

גוף הפעולה תחום כרגיל בתוך סוגריים מסולסלים. בתוך הסוגריים אנו כותבים את ההוראות המממשות את עדכון תכונות הזמן. שימו לב שהפעולה `SetTime` משתמשת בתכונות הפרטיות שהצהרנו עליהן, ושרק דרך פעולות שהגדרנו במחלקה ניתן לגשת לתכונותיה המוגדרות כ-**private**.

באופן דומה נגדיר את פעולת החזרת הייצוג הסטנדרטי של הזמן:

```
public string GetTime()
{
    return hour + ":" + minute;
}
```

כותרת הפעולה מציינת ששם הפעולה הוא `GetTime`, הפעולה לא מקבלת פרמטרים (כפי שמציינים הסוגריים הריקים), אך היא מחזירה ערך מסוג **string**.

גוף הפעולה כולל משפט **return** שתפקידו לסיים את ביצוע הפעולה ולהחזיר לנקודה שממנה הופעלה הפעולה את הערך הרשום בביטוי המופיע אחרי המילה **return**. במקרה זה תוחזר המחרוזת `hour + ":" + minute` המייצגת את הזמן. טיפוס הביטוי המוחזר בהוראה **return** חייב להיות זהה לטיפוס המופיע בכותרת הפעולה (במקרה זה **string**). במקרה של פעולות שלא מחזירות ערך כמו הפעולה `SetTime` אין צורך במשפט **return**, והחֲזרה מהפעולה מתבצעת עם ההגעה לסוף תחום הפעולה.

הנה ההגדרה המלאה של המחלקה *זמן*:

```
/*
  מחלקת זמן
*/
public class Time
{
    // הגדרת התכונות
    private int hour = 0;      // שעות
    private int minute = 0;    // דקות
    // פעולה לעדכון הזמן
    public void SetTime(int h, int m)
    {
        hour = h;
        minute = m;
    }
    //פעולה להחזרת הייצוג הסטנדרטי של הזמן//
    public string GetTime()
    {
        return hour + ":" + minute;
    }
} // class Time
```

הגדרת הפעולה הראשית

כתבנו מחלקה המגדירה את הטיפוס זמן וכעת עלינו לכתוב את התוכנית לפתרון הבעיה. התוכנית תיכתב במחלקה נפרדת בתוך הפעולה הראשית (המכונה Main) כפי שאנו יודעים.

פירוק הבעיה לתת-משימות

תיאור הבעיה מוביל לפירוק הבא לתת-משימות:

1. יצירת עצם מסוג זמן
2. קליטת נתוני זמן (שעה ודקה) ועדכון העצם מסוג זמן
3. הצגת הייצוג הסטנדרטי של הזמן

פתרון המשימה יהיה דומה לתוכניות שכתבנו בפרקים הקודמים. אבל במקום להשתמש רק בטיפוסים המוגדרים מראש בשפה, נשתמש גם במחלקה החדשה שבנינו, המחלקה *זמן*. אם כך, נעבור לשלב בחירת המשתנים:

בחירת משתנים

tm – עצם מהמחלקה Time.

h – השעה כפי שניתן על-ידי המשתמש.

m – הדקה כפי שניתן על-ידי המשתמש.

מימוש

כמו שלמדנו בפרקים קודמים, בניגוד למשתנים מטיפוסים פשוטים הנוצרים באופן אוטומטי עם ההצהרה עליהם, כאשר אנו משתמשים בעצמים עלינו קודם כל ליצור אותם. תהליך יצירת עצם כולל הקצאת מקום בזיכרון עבור העצם ואתחול של תכונותיו. יצירת עצם נעשית באמצעות ההוראה **new**, למשל כך:

```
Time tm;
tm = new Time();
```

אחרי שיצרנו את העצם tm, נוכל להפעיל עליו את הפעולות שהוגדרו עבורו. הפעלת פעולה מתבצעת באמצעות סימון הנקודה, למשל, המשפט הבא מעדכן את הזמן בעצם tm:

```
tm.SetTime(h,m);
```

שימו ♥: לא ניתן להתייחס באמצעות סימון הנקודה, לתכונותיו של העצם tm מתוך הפעולה הראשית, כי אלו הוגדרו כפרטיות (**private**). לכן הביטויים tm.hour ו-tm.minute אינם חוקיים.

הנה המימוש המלא של המחלקה הראשית:

```
/*
        המחלקה הראשית המשתמשת במחלקה זמן
*/
using System;
public class UseTime
{
    public static void Main()
    {
        // הגדרת משתנים
        Time tm;
        int h;
        int m;
        //יצירת עצם מסוג זמן
        tm = new Time();
        // קלט
        Console.Write("Enter the hour (0..23): ");
        h = int.Parse(Console.ReadLine());
        Console.Write("Enter the minute (0..59): ");
        m = int.Parse(Console.ReadLine());
        // עדכון הזמן והצגת פלט
        tm.SetTime(h,m);
        Console.WriteLine("The time is: {0}", tm.GetTime());
    } // Main
} // class UseTime
```

סוף פתרון בעיה 1

נסכם את הנלמד בפתרון בעיה 1:

לצורך פתרון הבעיה הגדרנו טיפוס חדש, את המחלקה זמן. לכן, שלא כמו התוכניות שכתבנו עד כה שכללו רק מחלקה אחת והיא המחלקה הראשית, בתוכנית זו יש שתי מחלקות: האחת מגדירה את הטיפוס החדש Time והשנייה היא המחלקה הראשית המגדירה את הפעולה Main שניתן להריץ במחשב. הפעולה Main יוצרת עצם מסוג Time ומפעילה את פעולותיו.

שימו ♥ : נהוג בשפת C# לשים כל מחלקה בקובץ נפרד אף על פי שניתן לשים יותר ממחלקה אחת בקובץ.

- **הגדרת מחלקה** היא למעשה הגדרה של טיפוס חדש. הגדרת מחלקה כוללת הגדרה של תכונות ושל פעולות עבור עצמים של המחלקה, כלומר עבור משתנים מהטיפוס החדש.
- **תכונות של עצם** הן משתנים מטיפוסים כלשהם, המאפיינים את העצם.
- **פעולות של עצם** אף הן משויכות לעצם, והן פועלות בדרך כלל על תכונות העצם.
- כדי לא לחשוף את אופן המימוש הפנימי של עצמים, נקפיד להגדיר את התכונות כ**פרטיות**. כלומר מחוץ למחלקה לא ניתן להתייחס אליהן ישירות.
- פעולות יוגדרו בדרך כלל כ**ציבוריות**.

- **הגדרת מחלקה בשפת C#** נכתבת באופן הבא:

```
public class שם המחלקה
{
    // הגדרת התכונות
    .
    .
    // הגדרת הפעולות
    .
    .
}
```

- המילה השמורה **public** מציינת שהמחלקה פתוחה לשימוש ציבורי. כלומר שמחלקות אחרות יכולות ליצור עצמים מטיפוס המחלקה ולהפעיל את הפעולות שלהם.
- המילה השמורה **class** מציינת את הגדרתה של מחלקה חדשה בשפה. לאחר ציון המילה **class** נציין את שם המחלקה. מקובל להתחיל שם מחלקה באות גדולה. אם שם המחלקה מורכב מכמה מילים, הן נכתבות צמודות כאשר כל מילה מתחילה באות גדולה ושאר האותיות הן קטנות.
- **הגדרת תכונות בשפת C#** נכתבת בדומה להצהרה על משתנים. הצהרה של תכונה של עצם כוללת את הרשאת הגישה לתכונה, את הטיפוס של התכונה ולאחר מכן את שמה. הרשאת גישה פרטית לתכונות מוגדרת באמצעות המילה השמורה **private**, למשל:

```
private int hour;
```

- אפשר לאתחל תכונה בעת הצהרתה, למשל:

```
private int hour = 0;
```

במקרה שלא מאתחלים את התכונות במפורש, שפת C# מאתחלת תכונות מטיפוס מספרי בערך אפס, תווים מאותחלים בתו מיוחד המייצג תו ריק, תכונות בוליאניות בערך **false** ומערכים ועצמים בערך מיוחד **null**.

- ניתן לגשת לתכונות פרטיות של עצם רק מתוך פעולות העצם – פעולות המוגדרות באותה המחלקה.

♦ **הגדרת פעולה בשפת C#** נכתבת באופן הבא:

```
public המוחזר-הערך-טיפוס שם-הפעולה(רשימת פרמטרים)
{
    גוף הפעולה
}
```

♦ **כותרת הפעולה**

כותרת הפעולה נותנת למשתמש את כל המידע הדרוש לצורך שימוש בפעולה. כותרת הפעולה כוללת את הרכיבים הבאים:

הרשאת גישה לפעולה

רכיב זה קובע מי רשאי להפעיל את הפעולה של העצם. משמעות המילה **public** היא כי הפעולה יכולה להיות מופעלת מכל מחלקה אחרת בתוכנית.

טיפוס הערך המוחזר

כאשר הפעולה איננה מחזירה דבר נכתוב **void** כטיפוס הערך המוחזר. בכל מקרה אחר, נכתוב את שם הטיפוס, והוא יכול להיות טיפוס פשוט בשפה (כמו **int**), מערך (כמו **int[]**) או שם מחלקה (כמו **string**).

שם הפעולה

שם הפעולה הוא שם המתאר את הפעולה המתבצעת. נהוג ששם הפעולה יתחיל באות גדולה. אם השם מורכב מכמה מילים, הן נכתבות צמודות כאשר כל מילה מתחילה באות גדולה ושאר האותיות הן קטנות.

הפרמטרים

כאשר פעולה לא מגדירה פרמטרים הסוגריים נותרים ריקים. בכל מקרה אחר הרשימה מורכבת מזוגות המופרדים בפסיקים. כל זוג מתאר פרמטר אחד וכולל את טיפוס הפרמטר ואת שמו.

♦ **גוף הפעולה**

את גוף הפעולה יש לתחום בסוגריים מסולסלים (הסימנים {...}).

בגוף הפעולה נכתוב את רצף ההוראות שמבצע את מטרת הפעולה, כלומר מיישם את האלגוריתם של הפעולה. בגוף הפעולה ניתן להצהיר על משתנים ולהשתמש בכל ההוראות הקיימות בשפת C#. כגון משפטי תנאי, לולאות, השמות ועוד. במקרה שמצהירים על משתנים, משתנים אלה מוּכָּרִים רק בגוף הפעולה, הם נהרסים עם סיום ביצוע הפעולה ולא ניתן להשתמש בהם מחוץ לה.

שימו ♥: בגוף הפעולה אפשר להתייחס ישירות לכל התכונות של העצם שמופעלת עליו הפעולה. אמנם התכונות הן פרטיות, כלומר מוחבאות משאר חלקי התוכנית, אולם הן ידועות וגלויות לכל פעולות העצם המוגדרות במחלקה. לכן מתוך הפעולות מותרת התייחסות ישירה לתכונות.

♦ **הוראת חזרה return**

כאשר ביצועה של ההוראה האחרונה בפעולה מסתיים, הפעולה כולה מסתיימת. אבל ניתן גם לגרום במפורש לסיום הביצוע באמצעות הוראת החזרה **return**. הוראה זו גורמת ליציאה מיידית מהפעולה. כאשר הפעולה אמורה להחזיר ערך, הוראת **return** כוללת גם ביטוי שיוחזר עם סיום הביצוע. טיפוס הביטוי המוחזר מהפעולה חייב להיות תואם לטיפוס הערך המוחזר שמוגדר בכותרת הפעולה.

שימו ♥: למען קריאות המחלקה חשוב מאוד לצרף להגדרה הערות המתארות את המחלקה, את תכונותיה ואת פעולותיה.

- כדי **להשתמש במחלקה** ניצור ממנה עצם במחלקה אחרת (למשל במחלקה הראשית).
- כדי **ליצור עצם** נצהיר עליו ונקצה לו מקום בזיכרון באמצעות ההוראה **new**, למשל:

```
Time tm;
tm = new Time();
```

- כדי **להפעיל את פעולות העצם** (לזמן את פעולות העצם) נשתמש בסימון הנקודה, למשל:

```
tm.SetTime(h,m);
```

יש לשים לב להתאמה בין אופן השימוש בפעולה של העצם לבין כותרת הפעולה: אם הפעולה מצפה לקבל פרמטרים יש להקפיד על סדר הפרמטרים כפי שמופיע בכותרת הפעולה, ועל התאמה של טיפוס הפרמטרים המועברים לטיפוסים המפורטים בכותרת הפעולה; אם הפעולה מחזירה ערך יש להקפיד על התאמה של טיפוס הערך המוחזר לטיפוס הביטוי שמשולב בו הערך.

שאלה 11.1

א. הוסיפו למחלקה Time פעולה בשם Increment המקדמת את הזמן בדקה. למשל לאחר הפעולה Increment הזמן 10:52 ישתנה ל- 10:53, והזמן 23:59 ישתנה ל- 0:0.

ב. כתבו פעולה ראשית הקולטת מהמשתמש נתוני זמן (שעה ודקה) ומספר שלם k. התוכנית תקדם את הזמן ב-k דקות (באמצעות הפעולה שהוגדרה בסעיף הקודם) ותציג את השעה שהתקבלה.

בעיה 2

מטרת הבעיה ופתרונה: היכרות עם פעולה בונה (**constructor**).

בשיעורי הנדסה בכיתה ד' המורה לימדה כיצד לחשב שטח והיקף של מלבן. כעת רוצה המורה לבחון את תלמידיה. המבחן כולל 20 שאלות שבכל אחת מהן התלמיד נדרש לחשב את שטחם ואת היקפם של מלבנים אשר אורכיהם ורוחביהם נתונים. פתחו וממשו תוכנית שתעזור למורה לחשב את התשובות למבחן. התוכנית תקלוט עבור כל שאלה את האורך ואת הרוחב של המלבן ותציג את שטחו ואת היקפו.

מתוך תיאור הבעיה ניתן לזהות שהשאלה עוסקת במלבנים. אפשר להסתכל על כל מלבן כעל ישות המתאפיינת בשני נתונים – אורך ורוחב, ומתאפיינת בשתי פעולות – אחת לחישוב השטח והאחרת לחישוב ההיקף. אם כך, ניתן להגדיר מחלקה בשם מלבן אשר יאגד בתוכו את תכונות המלבן ואת הפעולות של המלבן. לאחר שנגדיר את הטיפוס מלבן נוכל ליצור מלבנים שונים, כלומר עצמים, שכל אחד מהם ייצג מלבן אחר. לכל מופע (עצם) של הטיפוס מלבן, יהיו גם התכונות אורך ורוחב אשר מתארות את המלבן וגם הפעולות שטח והיקף שניתן להפעיל על המלבן.

לפתרון הבעיה נגדיר תחילה את המחלקה *מלבן* ולאחר מכן נמשיך בשלבי פיתוח הפתרון.

הגדרת המחלקה *מלבן*

הגדרת התכונות

אורך ורוחב הן יחידות מידה. לכן נגדיר אותן מטיפוס ממשי. בהתאם לכך נבחר את המשתנים הבאים לתיאור התכונות:

- **length** – משתנה מטיפוס ממשי המייצג את אורך המלבן.
- **width** – משתנה מטיפוס ממשי המייצג את רוחב המלבן.

הגדרת הפעולות

על פי הגדרת הבעיה, מלבן יכול לבצע את הפעולות הבאות:

- **חישוב שטח** – פעולה זו מחשבת ומחזירה את שטח המלבן. בפעולה זו מוגדרת משימה אלגוריתמית, עם נקודת מוצא (אורך ורוחב המלבן) ופלט מבוקש (שטח). ניתן לחשב את שטח המלבן באמצעות הביטוי הבא:

width * length

- **חישוב היקף** – פעולה זו מחשבת ומחזירה את היקף המלבן. הנה הביטוי לחישוב ההיקף:

2 * width + 2 * length

הגדרנו את הפעולות שטח והיקף. פעולות אלו משתמשות בתכונות אורך המלבן ורוחב המלבן. כדי לאתחל את תכונות המלבן בגדלים של מלבן מסוים ניתן להגדיר פעולה מיוחדת המיועדת לאתחול תכונות העצם. פעולה זו נקראת **פעולה בונה** והיא מתבצעת מיד עם יצירת עצם מסוג מלבן. אם כך, נוסיף את הפעולה הבאה:

- **פעולה בונה -** פעולה זו תקבל כפרמטרים את אורך המלבן ואת רוחבו ותאתחל את התכונות בהתאם.

הגדרנו את תכונות המלבן ואת פעולותיו כפי שנובעות מהגדרת הבעיה. כעת נממש את ההגדרות האלו בשפת #C.

מימוש המחלקה

נגדיר מחלקה בשם `Rectangle`. בתוך תחום המחלקה נפרט את המרכיבים השונים:

התכונות אורך ורוחב:

```
private double length; //מכיל את אורך המלבן
private double width; // מכיל את רוחב המלבן
```

הפעולות שטח והיקף:

```
public double Area()
{
   return width * length;
}
public double Perimeter()
{
   return 2 * width + 2 * length;
}
```

הפעולה הבונה:

כאמור הפעולה הבונה היא פעולה מיוחדת המיועדת לאתחול תכונות העצם. פעולה זו תופעל בעת יצירת המופע והיא מחזירה הפניה לעצם שנוצר. הגדרת הפעולה הבונה דומה להגדרת פעולה רגילה כמו חישוב שטח וחישוב היקף שהוצגו לעיל, פרט לכך ששמה זהה לשם המחלקה ולא רושמים את טיפוס הערך המוחזר (גם לא **void**). הפעולה הבונה, כמו כל פעולה, יכולה לקבל פרמטרים.

אם כך, נוכל להגדיר עבור המחלקה מלבן פעולה בונה ששמה Rectangle, והיא תקבל את האורך ואת הרוחב של המלבן. נשתמש בה כדי לאתחל את תכונותיו של עצם מסוג מלבן.

הנה פעולה בונה עבור עצמים מסוג *מלבן* :

```
public Rectangle(double aLength, double aWidth)
{
    length = aLength;
    width = aWidth;
}
```

שימו ♥ : השם של הפעולה הבונה חייב להיות זהה לשם המחלקה, ובכותרת שלה אין טיפוס לערך מוחזר.

יש מקרים שבהם משתמשים בשמות פרמטרים זהים לשמות התכונות. הדבר נפוץ במיוחד בפעולות שהפרמטרים בהן מתייחסים בצורה ישירה לתכונות. למשל ניתן לכתוב את הכותרת של הפעולה הבונה באופן הבא :

```
public Rectangle(double length, double width)
```

נרצה שהפרמטר length יאתחל את התכונה length, ושהפרמטר width יאתחל את התכונה width. מכיוון ששמות הפרמטרים זהים לשמות התכונות, אנו צריכים דרך להבחין ביניהם. אם בתוך הפעולה נתייחס ישירות למזהים length ו-width אז נתייחס למעשה לפרמטרים length ו-width ולא לתכונות באותם השמות. אם כך, כיצד נתייחס לתכונות?

בשפת C# קיימת בתוך כל פעולה של עצם הפניה בשם **this** המפנה לעצם עצמו – העצם שהפעולה שלו הופעלה. כלומר אם נרשום בתוך הפעולה את הביטוי **this**.length אז נתייחס לתכונה length של העצם שאנו יוצרים ולא לפרמטר length.

אם כך, נוכל לכתוב את הפעולה הבונה כך :

```
public Rectangle(double length, double width)
{
    this.length = length;
    this.width = width;
}
```

באופן כללי, כאשר שם של פרמטר זהה לשם של תכונה, נשתמש בקידומת **this** כדי לציין את התכונה.

הנה ההגדרה המלאה של המחלקה *מלבן* :

```
/*
 מחלקת מלבן
*/
public class Rectangle
{
    // הגדרת התכונות
    private double width;      // רוחב
    private double length;     // אורך
    //פעולה בונה
    public Rectangle(double length, double width)
    {
        this.length = length;
        this.width = width;
    }
```

```
    // פעולת שטח
    public double Area()
    {
        return width * length;
    }
    //פעולת היקף
     public double Perimeter()
    {
        return 2 * width + 2 * length;
    }
} // class Rectangle
```

הגדרת הפעולה הראשית

כתבנו מחלקה המגדירה את הטיפוס מלבן וכעת עלינו לכתוב את הפעולה הראשית לפתרון הבעיה.

פירוק הבעיה לתת-משימות

תיאור הבעיה מוביל לפירוק הבא לתת-משימות:

עבור כל שאלה מ-20 שאלות המבחן:

1. קליטת אורך ורוחב המלבן ויצירת עצם מסוג מלבן
2. חישוב שטח המלבן (באמצעות פעולת השטח) והצגתו
3. חישוב היקף המלבן (באמצעות פעולת ההיקף) והצגתו

בחירת משתנים

rec – עצם מהמחלקה Rectangle. מייצג את המלבן עבור כל שאלה.

length – אורך המלבן כפי שניתן על-ידי המשתמש.

width – רוחב המלבן כפי שניתן על-ידי המשתמש.

האלגוריתם

1. בצע 20 פעמים:
   1.1 קלוט את אורך המלבן ב-length
   1.2 קלוט את רוחב המלבן ב-width
   1.3 צור עצם מסוג Rectangle בשם rec ואתחל אותו באורך וברוחב שנקלטו מהמשתמש.
   1.4 הפעל את פעולת שטח המלבן על rec והצג את התוצאה
   1.5 הפעל את פעולת היקף המלבן על rec והצג את התוצאה

מימוש

כפי שראינו בבעיה הקודמת, כאשר אנו משתמשים בעצם עלינו קודם כל להקצות עבורו מקום חדש בזיכרון ולאתחל את תכונותיו. הקצאת המקום מתבצעת באמצעות ההוראה **new** והאתחול מתבצע באמצעות קריאה לפעולה הבונה, למשל כך:

```
Rectangle rec;
rec = new Rectangle (5,2.5);
```

ואכן, עם הקצאת המקום לעצם `rec` מופעלת הפעולה הבונה ששמה `Rectangle`. פעולה זו מבצעת את האתחולים הדרושים לצורך תחילת עבודה תקינה עם העצם שנוצר. משום כך היא נקראת **פעולה בונה**.

הנה המימוש המלא של המחלקה הראשית:

```
/*
        המחלקה הראשית המשתמשת במחלקה מלבן
*/
using System;
public class TestSolver
{
    public static void Main()
    {
        // הגדרת משתנים
        Rectangle rec;
        double length;
        double width;
        for (int i = 0; i < 20; i++)
        {
            //  קלט
            Console.Write("Enter the rectangle length: ");
            length = double.Parse(Console.ReadLine());
            Console.Write("Enter the rectangle width: ");
            width = double.Parse(Console.ReadLine());
            //הקצאת מלבן
            rec = new Rectangle(length,width);
            // ביצוע חישובים והצגת פלט //
            Console.WriteLine("The rectangle area is: {0}",
                                                      rec.Area());
            Console.WriteLine("The rectangle perimeter is: {0}",
                                                  rec.Perimeter());
        } // for
    } // Main
} // class TestSolver
```

סוף פתרון בעיה 2

בפתרון בעיה זו הכרנו גם את הפעולה הבונה.

- **הפעולה הבונה** של עצם מופעלת מיד אחרי שההוראה **new** מקצה עבורו שטח בזיכרון.
- **שמה של פעולה בונה** הוא תמיד כשם המחלקה.
- כמו כל פעולה, גם פעולה בונה יכולה לקבל פרמטרים. **פרמטרים** אלה משמשים לאתחול תכונות העצם.
- במקרים ששם של פרמטר זהה לשם של תכונה נוכל להתייחס לתכונה באמצעות הקידומת **this** המציינת התייחסות לתכונות של העצם הנוכחי – העצם שמופעלת עליו הפעולה. אפשר להשתמש ב-**this** מתוך כל פעולה המוגדרת בעצם ולא רק מתוך הפעולה הבונה.
- אם איננו מגדירים פעולה בונה כלשהי במחלקה, שפת C# מספקת פעולה בונה ריקה חסרת פרמטרים. פעולה בונה זו היא ברירת המחדל במחלקה שלא הגדרנו עבורה פעולה בונה. ברגע שנגדיר פעולה בונה במחלקה, היא תחליף את הפעולה הבונה שסופקה כברירת מחדל.

שאלה 11.2

הגדירו וממשו את המחלקות הבאות בשפת C#:

המחלקה	תכונות	פעולות
חתול	- שם - סוג - אוהב או לא אוהב מים - אוהב או לא אוהב ליילל	- פעולה בונה המאתחלת את תכונות העצם. - האם צמא: עבור חתול שאוהב מים הפעולה תחזיר **true** אחרת יוחזר **false**. - יללה: עבור חתול שאוהב ליילל הפעולה תחזיר את המחרוזת "מיאו מיאו מיאו". עבור חתול שאינו אוהב ליילל הפעולה תחזיר את המחרוזת "מיאו". - בירור פרטי החתול: הפעולה תחזיר את המחרוזת: שמי <שם החתול> והסוג שלי הוא <סוג החתול>.
ילד	- שם - גיל	- פעולה בונה המאתחלת את תכונות העצם. - בירור פרטים אישיים: הפעולה תחזיר את המחרוזת: שמי <שם הילד> וגילי הוא <גיל הילד>. - יום הולדת: הפעולה תקדם את גיל הילד באחד.
מעגל	- רדיוס	- פעולה בונה המאתחלת את תכונות העצם. - שטח: הפעולה תחזיר את שטח המעגל. - היקף: הפעולה תחזיר את היקף המעגל. **הדרכה**: השתמשו בקבוע `Math.PI`

שאלה 11.3

כתבו תוכנית המשתמשת במחלקת החתול שהוגדרה בשאלה הקודמת. התוכנית תקלוט מהמשתמש נתונים עבור חתולים (שם, סוג, האם הוא אוהב מים והאם הוא אוהב ליילל). עבור כל חתול, יוצג כפלט: פרטי החתול (שם וסוג), היללה שלו והאם הוא צמא.

התוכנית תסתיים כאשר המשתמש יקליד מחרוזת ריקה עבור שם החתול.

11.2 פעולות גישה

בעיה 3

מטרת הבעיה ופתרונה: היכרות עם פעולות גישה המחזירות ומעדכנות את ערכי תכונות העצם.

לאמא ארנבת שני ארנבונים: ארני וברני. בכל יום שישי מודדת אמא ארנבת את אורך אוזני גוריה ואת משקלם. עזרו לאמא ארנבת לנהל את מדידותיה. הגדירו וממשו מחלקה בשם "ארנבון" בה יישמרו נתוני כל ארנבון: שמו, אורך אוזניו ומשקלו. שימו לב שאורך אוזני הארנבון אינו יכול לקטון.

כתבו תוכנית שנתית המשמשת למדידת הארנבונים. תחילה התוכנית תיצור את שני הארנבונים ארני וברני, ולאחר מכן היא תקלוט עבור כל שבוע (52 שבועות) את אורך האוזניים ואת המשקל של כל ארנבון. אם אמא ארנבת שגתה בהכנסת הנתונים והכניסה אורך אוזניים קצר יותר מהנוכחי, תציג התוכנית הודעת שגיאה ותבקש להכניס שוב את הנתונים. בסוף השנה תדפיס התוכנית את הנתונים העדכניים של כל ארנבון.

הגדרת המחלקה *ארנבון*

לצורך פתרון הבעיה נזדקק למחלקה המגדירה ארנבון.

הגדרת התכונות

על פי הגדרת הבעיה, לעצם מהמחלקה ארנבון יהיו התכונות: שם ארנבון, אורך אוזניים ומשקל. בהתאם לכך נבחר את המשתנים הבאים לתיאור התכונות:

- **name** – מחרוזת המייצגת את שם הארנבון.
- **earsLength** - מספר מטיפוס שלם המייצג את אורך האוזניים של הארנבון.
- **weight** - מספר מטיפוס שלם המייצג את משקל הארנבון.

הגדרת הפעולות

- **פעולה בונה** – הפעולה הבונה תקבל כפרמטר את שם הארנבון, ותאתחל את תכונת שם הארנבון. מה לגבי אתחול שאר תכונות המחלקה, משקל ואורך אוזניים? בתחילה, אין נתונים עבור משקל ועבור אורך אוזני הארנבון. הנתונים לגבי תכונות אלו יתקבלו רק לאחר המדידה הראשונה. לכן הפעולה הבונה תאתחל את שם הארנבון לפי הפרמטר שהתקבל ותאתחל את שתי התכונות האחרות בערך התחלתי אפס.

פעולות לעדכון תכונות המחלקה – עבור כל ארנבון לאחר כל מדידה ועבור כל תכונה, נרצה לעדכן את ערך התכונה כפי שנמדד. כיצד נעשה זאת? כמובן שלפעולה הראשית המנהלת את המדידות אין גישה ישירה לתכונות של הארנבונים משום שתכונות אלו מוגדרות כפרטיות. אם כך, יש צורך בפעולות גישה המעדכנות ערך של תכונה, כלומר בפעולה עבור כל תכונה שנרצה לעדכן:

- **SetWeight** – הפעולה תקבל כפרמטר את משקלו של הארנבון כפי שנמדד ותעדכן את משקלו. הפעולה איננה מחזירה ערך.

- **SetEarsLength** – הפעולה תקבל כפרמטר את אורך אוזניו של הארנבון כפי שנמדד, ותעדכן אותו. שימו לב, לפי הגדרת הבעיה אורך אוזני הארנבון אינו יכול לקטון, כלומר עלינו לעדכן את התכונה אורך אוזני הארנבון רק אם האורך שהתקבל כפרמטר גדול יותר מאורך האוזניים הנוכחי של הארנבון. לפי הגדרת הבעיה, אם אמא ארנבת שגתה בהכנסת הנתונים והכניסה אורך אוזניים קצר יותר, תבקש ממנה הפעולה הראשית להכניס את הנתון שוב. כיצד תדע הפעולה הראשית שהאורך שהתקבל קצר מהאורך השמור? נשתמש בצינור הגישה גם בכיוון השני, הפעולה תחזיר ערך בוליאני המציין אם העדכון התבצע או לא.

פעולות גישה המחזירות תכונות של העצם – בסוף השנה תציג הפעולה הראשית עבור כל ארנבון את אורך אוזניו ואת גובהו. מכיוון שלפעולה הראשית המנהלת את המדידות אין גישה ישירה לתכונות של הארנבונים משום שתכונות אלו מוגדרות כפרטיות, נגדיר פעולות גישה לתכונות והן יחזירו את ערכן של התכונות. בדרך כלל פעולות הגישה הן פעולות שאינן מקבלות פרמטרים אלא רק מחזירות ערך מטיפוס התכונה.

- **GetWeight** – הפעולה איננה מקבלת פרמטרים והיא מחזירה את משקלו של הארנבון.

- **GetEarsLength** – הפעולה איננה מקבלת פרמטרים והיא מחזירה את אורך אוזניו של הארנבון.

מימוש המחלקה

```
/*
המחלקה ארנבון
*/
public class Rabbit
{
    private string name;
    private int weight;
    private int earsLength;
    //פעולה בונה מקבלת את שם הארנבון
    public Rabbit(string name)
    {
        this.name = name;
        weight = 0;
        earsLength = 0;
    }
    //פעולה המעדכנת את משקלו של הארנבון
    public void SetWeight(int weight)
    {
        this.weight = weight;
    }
    //פעולה המעדכנת את אורך אוזני הארנבון רק אם האורך שהתקבל גדול
    // מהאורך השמור. מחזירה אמת אם העדכון בוצע, שקר אחרת
    public bool SetEarsLength(int earsLength)
    {
        if (this.earsLength > earsLength)
            return false;
        this.earsLength = earsLength;
        return true;
    }
    //פעולת גישה המחזירה את משקלו של הארנבון
    public int GetWeight()
    {
        return weight;
    }
    //פעולת גישה המחזירה את אורך אוזניו של הארנבון
    public int GetEarsLength()
    {
        return earsLength;
    }
}//class Rabbit
```

שימו ♥ לאופן שמיושמת הפעולה `SetEarsLength`: בפעולה זאת תיתכן יציאה (באמצעות המילה **return**) באמצע הפעולה, בלי להגיע עד סופה. אם אורך האוזניים שהתקבל כפרמטר קטן מהאורך השמור, ביצוע הפעולה מסתיים מיד ומחזיר את הערך שקר. לא משנה אילו הוראות כתובות לאחר התנאי, בכל מקרה הן לא יבוצעו. המילה השמורה **return** גורמת ליציאה מיידית מהפעולה. כל ההוראות שנכתבות לאחריה אינן מבוצעות; פעולת העדכון במקרה זה.

הגדרת הפעולה הראשית

הגדרנו מחלקה לייצוג הטיפוס ארנבון, וכעת עלינו להמשיך לפיתוח הפעולה הראשית האחראית על המדידות.

פירוק לתת-משימות

את משימתה של הפעולה הראשית ניתן לפרק לתת-משימות באופן הבא:

1. יצירת הארנבונים: ארני וברני
2. עבור כל אחד מ-52 השבועות בשנה:
   2.1. קליטת משקלו של ארני ועדכונו
   2.2. קליטת אורך אוזניו של ארני ועדכונו, חזרה על כך כל עוד העדכון לא בוצע (פעולת העדכון החזירה **false**)
   2.3. קליטת משקלו של ברני ועדכונו
   2.4. קליטת אורך אוזניו של ברני ועדכונו, חזרה על כך כל עוד העדכון לא בוצע (פעולת העדכון החזירה **false**)
3. הצגה כפלט: בסוף השנה משקלו של ארני הוא <משקלו של ארני> ואורך אוזניו הוא <אורך אוזניו של ארני>
4. הצגה כפלט: בסוף השנה משקלו של ברני הוא <משקלו של ברני> ואורך אוזניו הוא <אורך אוזניו של ברני>

בחירת משתנים

rabit1 – עצם מטיפוס המחלקה ארנבון, מייצג את ארני.

rabit2 – עצם מטיפוס המחלקה ארנבון, מייצג את ברני.

currentWeight – משתנה מטיפוס שלם, משמש לקליטת משקל הארנבונים.

currentEarsLength – משתנה מטיפוס שלם, משמש לקליטת אורך אוזני הארנבונים.

לא נתעכב על כתיבת האלגוריתם המלא הנובע כמעט באופן ישיר מהפירוק לתת-משימות שניתן לעיל.

מימוש

```
/*
    המחלקה הראשית המשמשת למדידת הארנבונים
*/
using System;
public class RabbitsDevelopment
{
    public static void Main()
    {
        Rabbit rabbit1 = new Rabbit("Arni");
        Rabbit rabbit2 = new Rabbit("Barni");
        int currentWeight;
        int currentEarsLength;
        //עבור כל אחד מהשבועות בשנה נעדכן את נתוני הארנבונים
        for(int i = 0; i < 52; i ++)
        {
            Console.Write("Enter Arni's weight: ");
            currentWeight = int.Parse(Console.ReadLine());
            rabbit1.SetWeight(currentWeight);
            Console.Write("Enter Arni's ears length: ");
            currentEarsLength = int.Parse(Console.ReadLine());
            //כל עוד העדכון לא בוצע, נבקש להכניס שוב את האורך
            while (!rabbit1.SetEarsLength(currentEarsLength))
```

```
         {
            Console.Write("ReEnter Arni's ears length:");
            currentEarsLength = int.Parse(Console.ReadLine());
         }
         Console.Write("Enter Barni's weight: ");
         currentWeight = int.Parse(Console.ReadLine());
         rabbit2.SetWeight(currentWeight);
         Console.Write("Enter Barni's ears length:");
         currentEarsLength = int.Parse(Console.ReadLine());
         //כל עוד העדכון לא בוצע, נבקש להכניס שוב את האורך
         while (!rabbit2.SetEarsLength(currentEarsLength))
         {
            Console.Write("ReEnter Barni's ears length:");
            currentEarsLength = int.Parse(Console.ReadLine());
         }
      }
      //פלט
      Console.WriteLine("At the end of the year Arni's weight " +
                              "is: {0}, Arni's ears length is: {1}",
                      rabbit1.GetWeight(), rabbit1.GetEarsLength());
      Console.WriteLine("At the end of the year Barni's weight " +
                             "is: {0}, Barni's ears length is: {1}",
                      rabbit2.GetWeight(), rabbit2.GetEarsLength());
   }//Main
}// RabbitsDevelopment
```

סוף פתרון בעיה 3

פתרון בעיה 3 כלל פעולות גישה לעדכון ערכי תכונות ולהחזרתם. מאחר שתכונות של עצם מוגדרות בדרך כלל כפרטיות, לא ניתן לגשת אליהן ישירות מחוץ למחלקה. אם כך, גישה לתכונות העצם מחוץ למחלקה נעשית באמצעות **פעולות גישה**.

אנו לא חייבים לספק פעולות גישה לכל תכונה. פעולה המעדכנת ערך של תכונה נכתבת רק עבור תכונות שנרצה שהמשתמש בָּעֶצֶם יוכל לעדכן. פעולה כזו מקבלת את הערך החדש של התכונה כפרמטר. היא בדרך כלל לא מחזירה ערך, או שהיא מחזירה משתנה בוליאני המציין אם פעולת העדכון התבצעה בהצלחה. נהוג ששמה של הפעולה יהיה שרשור של המילה Set ולאחריו שם התכונה. למשל עבור התכונה weight שם פעולת העדכון יהיה SetWeight.

פעולת גישה המחזירה ערך של תכונה נכתבת רק עבור תכונות שאת ערכן נרצה שמשתמש חיצוני יוכל לקבל. פעולה כזו אינה מקבלת פרמטרים, אלא מחזירה ערך מטיפוס התכונה. נהוג ששמה של הפעולה יהיה שרשור של המילה Get לשם התכונה. למשל עבור התכונה weight שם פעולת הגישה להחזרת ערך התכונה יהיה GetWeight.

פעולות גישה משמשות צינור גישה בטוח אל תכונות העצם מחוץ למחלקה. פעולת עדכון יכולה לקבוע את חוקיות הערכים המעדכנים את התכונה (בדומה לפעולת העדכון `SetEarsLength`). בנוסף לכך, גם אם המתכנת שינה את ייצוג התכונה (למשל את שמה), עדיין לא ישתנו הערך המוחזר מפעולת הגישה והפרמטר לפעולת העדכון, וכך שאר חלקי התוכנית המשתמשים בפעולות הגישה לא יהיו מושפעים מהשינויים. בשל כך התוכנית כולה תהיה עמידה יותר בפני שינויים.

שאלה 11.4

כתבו מחלקה המגדירה טיפוס קלמר, כמתואר בשתי הטבלאות הבאות :

התכונה	טיפוס	תיאור
מספר העטים	שלם	מספר העטים שנמצאים בקלמר
מספר העפרונות	שלם	מספר העפרונות שנמצאים בקלמר

הפעולה	טיפוס ערך מוחזר	רשימת פרמטרים	תיאור
פעולה בונה		מספר העטים ומספר העפרונות בקלמר	הפעולה מאתחלת את מספר העטים ואת מספר העפרונות בקלמר
הוספת עט	void	אין	פעולה שמגדילה את מספר העטים בקלמר ב-1
הוספת עיפרון	void	אין	פעולה שמגדילה את מספר העפרונות בקלמר ב-1
אובדן עט	void	אין	פעולה שמקטינה את מספר העטים בקלמר ב-1
אובדן עיפרון	void	אין	פעולה שמקטינה את מספר העפרונות בקלמר ב-1
החזרת מספר העטים	מספר שלם	אין	פעולת גישה שמחזירה את מספר העטים בקלמר
החזרת מספר העפרונות	מספר שלם	אין	פעולת גישה שמחזירה את מספר העפרונות בקלמר

כתבו תוכנית שתקלוט את מספר העטים ואת מספר העפרונות בקלמר ותיצור עצם מסוג קלמר. לאחר מכן התוכנית תקלוט מספר המציין כמה אירועים עבר הקלמר, כאשר אירוע יכול להיות הוספה או איבוד של עט או עיפרון. לאחר מכן התוכנית תקלוט תיאור של כל האירועים, כאשר עבור כל אירוע ייקלט קודם כל סוגו כמספר שלם (1 – הוספה, 2 – איבוד), ולאחר מכן ייקלט סוג החפץ המשתתף באירוע, גם כן כמספר שלם (1 – עט, 2 – עפרון).

פלט התוכנית יהיה מספר העטים ומספר העפרונות בקלמר לאחר כל האירועים שעברו עליו.

למשל, אם נתוני הקלמר ההתחלתיים הם 4 10 (כלומר, 4 עטים ו-10 עפרונות), מספר האירועים הוא 3, ותיאור האירועים הוא 2 2 1 1 2 2 (כלומר, אבד עיפרון, נוסף עט, אבד עיפרון), אז פלט התוכנית צריך להיות הודעה המציינת כי בקלמר יש 5 עטים ו-8 עפרונות.

שאלה 11.5

כתבו מחלקה המגדירה את הטיפוס מעגל. תכונות של עצם מהמחלקה הן צבע המעגל ורדיוס המעגל. הפעולות שניתן לבצע על עצם מהמחלקה הן חישוב שטח, חישוב היקף והחזרת צבע המעגל.

כתבו תוכנית עזר לציור מעגלים. התוכנית תקלוט מהמשתמש את צבע המעגל ואת רדיוסו ותציג כפלט :

שטח המעגל הוא <שטח המעגל> היקף המעגל הוא <היקף המעגל> וצבעו הוא <צבע המעגל>

התוכנית תסתיים כאשר הצבע הנקלט יהיה שחור.

שאלה 11.6

פתחו וממשו מערכת לניהול מספרת כלבים. המערכת קולטת מספר שלם n ואחריו פרטים של n כלבים הבאים להסתפר במספרה. המערכת מציגה עבור כל כלב את סוג התספורת שיסופר בה. פרטים של כלב כוללים: שם כלב, אורך שיער (קצר או ארוך) וסוג נביחה ("הב", "הב-הב" או "הב-הב-הב"). סוגי התספורות האפשריים הם: גילוח, תספורת קצרה ותספורת רגילה.

המספרה מחליטה עבור כל כלב את סוג התספורת המתאימה לו: אם לכלב יש שיער קצר ונביחתו היא "הב" התספורת היא גילוח. אם לכלב יש שיער ארוך ונביחתו היא "הב הב" או "הב הב הב" הוא יסופר בתספורת קצרה. אחרת התספורת היא באורך רגיל.

הדרכה: התכונות של עצם מסוג כלב הם: שם, אורך שיער ונביחה. פעולות של עצם מסוג כלב הם: פעולה בונה ופעולות גישה המחזירות את ערכי התכונות. הפעולה הראשית תיצור עצמים מסוג כלב לפי נתונים המתקבלים מהקלט ותציג עבור כל כלב את סוג התספורת המתאימה לו.

בעיה 4

מטרת הבעיה ופתרונה: העמקה בעצמים

כתבו תוכנית המדמה משחק קובייה לזוג שחקנים. כל שחקן מטיל שתי קוביות בתורו. כל שחקן צובר את הנקודות מהטלות הקוביות שלו. "סיבוב" במשחק מורכב מתור של שחקן ראשון ואחריו תור של שחקן שני. המנצח הוא הראשון שמגיע ל-100 נקודות או יותר. אם שני השחקנים הגיעו ל-100 נקודות או יותר באותו "הסיבוב" נכריז על תיקו. אם שחקן מטיל דאבל (כגון 6-6) השחקן השני מקבל ניקוד כפול בתור הבא.

כתבו מחלקה המגדירה שחקן. לעצם מטיפוס שחקן נשמור את הניקוד המצטבר, ופעולותיו הן "שחק" ושתי פעולות גישה המאפשרות לברר את ניקוד השחקן ואם ניצח.

הגדרת המחלקה *שחקן*

נזדקק למחלקה המגדירה שחקן.

הגדרת התכונות

על פי הגדרת הבעיה, לעצם מהמחלקה שחקן תהיה תכונה השומרת את הניקוד המצטבר של השחקן. בהתאם לכך נבחר את המשתנה הבא לתיאור התכונה:

- **points** – משתנה מטיפוס שלם מייצג את הניקוד המצטבר שצבר השחקן עד כה.

הגדרת הפעולות

- **פעולה בונה** – הפעולה הבונה לא תקבל פרמטרים אלא רק תאתחל את הניקוד המצטבר.
- **Play** – הפעולה מקבלת פרמטר בוליאני הקובע אם לשחק כתור רגיל או כתור בניקוד כפול. הפעולה מדמה את זריקת שתי הקוביות, מעדכנת את הניקוד המצטבר של השחקן ומחזירה אם יצא דאבל.
- **GetPoints** – פעולת גישה המחזירה את הניקוד המצטבר של השחקן.
- **IsWin** – הפעולה איננה מקבלת פרמטרים והיא מחזירה **true** אם השחקן צבר 100 נקודות או יותר ו-**false** אחרת.

מימוש המחלקה

```
/*
המחלקה שחקן
*/
public class Player
{
    private int points;
    //פעולה בונה לשחקן//
    public Player()
    {
        points = 0;
    }
    // הפעולה מקבלת פרמטר בוליאני הקובע אם לשחק תור כרגיל או כתור
    // בניקוד כפול. הפעולה מדמה את זריקת הקוביות ומעדכנת את הניקוד
    // המצטבר של השחקן. הפעולה מחזירה אם יצא דאבל
    public bool Play(bool doublePoints)
    {
        Random rnd = new Random();
        int die1 = rnd.Next(1,7);
        int die2 = rnd.Next(1,7);
        int playPoints = die1 + die2;
        if (!doublePoints)
            points = points + die1 + die2;
        else
            points = points + 2 * (die1 + die2);
        return (die1 == die2);
    }
    //מחזירה את מספר הנקודות שצבר השחקן//
    public int GetPoints()
    {
        return points;
    }
    //מחזירה אמת אם הניקוד המצטבר גדול או שווה ל-100//
    public bool IsWin()
    {
        return (points >= 100);
    }
} // class Player
```

הגדרת הפעולה הראשית

כתבנו מחלקה המגדירה את הטיפוס שחקן, וכעת עלינו להמשיך לפיתוח הפעולה הראשית המדמה את המשחק.

פירוק לתת-משימות

את משימתה של הפעולה הראשית ניתן לפרק לתת-משימות באופן הבא:

1. יצירת השחקנים
2. דימוי המשחק

בחירת משתנים

player1 – משתנה מטיפוס שחקן, מייצג את שחקן 1.

player2 – משתנה מטיפוס שחקן, מייצג את שחקן 2.

doub – משתנה מטיפוס בוליאני מייצג האם השחקן זרק מספר כפול.

שימו ♥ לכך שלא ניתן לקרוא למשתנה double מפני שהמילה **double** היא מילה שמורה בשפה המציינת טיפוס ממשי. באופן כללי לא ניתן לקרוא למשתנים במילים שמורות בשפה.

כיצד נממש את התת-משימה השנייה?

האלגוריתם

1. כל עוד שחקן 1 לא ניצח וגם שחקן 2 לא ניצח בצע:
   1.1 אם לשחקן 2 יצא דאבל
      1.1.1 שחקן 1 – שחק כפול
   1.2 אחרת
      1.2.1 שחקן 1 - שחק
   1.3 אם לשחקן 1 יצא דאבל
      1.3.1 שחקן 2 – שחק כפול
   1.4 אחרת
      1.4.1 שחקן 2 – שחק
2. אם שחקן 1 ניצח וגם שחקן 2 ניצח
   2.1 הצג כפלט "תיקו"
3. אחרת אם שחקן 1 ניצח
   3.1 הצג כפלט "שחקן 1 ניצח"
4. אחרת
   4.1 הצג כפלט "שחקן 2 ניצח"

מימוש

```
/*
       המחלקה הראשית המממשת את המשחק
*/
using System;
public class DiceGame
{
    public static void Main()
    {
        Player player1 = new Player();
        Player player2 = new Player();
        bool doub = false;
        while (!player1.IsWin() && !player2.IsWin())
        {
             // שחקן ראשון משחק
             doub = player1.Play(doub);
             Console.WriteLine("player1 sum: {0}",
                                                player1.GetPoints());
             // שחקן שני משחק
             doub = player2.Play(doub);
```

```
            Console.WriteLine("player2 sum: {0}",
                                              player2.GetPoints());
        }
        //הצגת הפלט
        if (player1.IsWin() && player2.IsWin())
            Console.WriteLine("tie");
        else if (player1.IsWin())
            Console.WriteLine("player 1 won");
        else
            Console.WriteLine("player 2 won");
    } // Main
} // class DiceGame
```

סוף פתרון בעיה 4

כפי שכבר ציינו, בעת הגדרת מחלקה ניתן לספק פעולות גישה המאפשרות לעדכן תכונות (Set) ולאחזר את ערכן (Get). לעתים נרצה לאפשר גישה חלקית בלבד או לא לאפשר גישה בכלל. למשל בפתרון בעיה 4, לתכונה points הוגדרה פעולת גישה המחזירה את ערכה, אך לא הוגדרה פעולת גישה המעדכנת את ערכה. הסיבה לכך היא, שאנו רוצים להגן על התכונה מפני עדכון מבחוץ. רק עצם מסוג שחקן יכול לעדכן את הניקוד שלו.

ההחלטה לאילו תכונות לאפשר גישה דרך פעולות גישה ואילו תכונות לא לחשוף כלל, מושפעת מהגדרת הבעיה ובפרט מתפקידן של התכונות במשימה שיש לפתור. הגדרה מבוקרת ומושכלת של פעולות גישה יכולה לסייע לנו בהגנה על הנתונים מפני שינוי בלתי מבוקר, וכן בהסתרת מידע פרטי מפני שאר חלקי התוכנית.

שאלה 11.7

במשחק "צבעי קלפים" מקבל כל משתתף 5 קלפים מכל צבע. הצבעים הם כחול, אדום וירוק. במשחק משתתפים 4 שחקנים. כל משתתף בתורו, זורק קלף בצבע כלשהו ולוקח קלף אחר מהקופה. המנצח הוא המשתתף הראשון שצבר 10 קלפים מאותו צבע.

פתחו וממשו תוכנית לניהול משחק "צבעי הקלפים". התוכנית תשתמש במחלקה "שחקן".

בכל תור התוכנית תפעיל על השחקן התורן את פעולת זריקת קלף, תגריל עבורו את צבע הקלף שיקבל מהקופה, ותעדכן את קלפיו בהתאם. התוכנית תסתיים כאשר אחד השחקנים ינצח.

הדרכה: לעצם מטיפוס המחלקה "שחקן" יש שלוש תכונות: עבור כל אחד משלושת הצבעים יש תכונה השומרת את מספר הקלפים שיש לשחקן מצבע זה. לעצם ממחלקה זו יש שתי פעולות: הוספת קלף וזריקת קלף. פעולת הוספת הקלף תקבל כפרמטר את צבע הקלף שיש להוסיף ותעדכן את התכונה המתאימה. פעולת זריקת קלף תחזיר את צבע הקלף שבחר השחקן לזרוק. הפעולה תבחר את הצבע שממנו יש לשחקן הכי פחות קלפים (אם יש יותר מצבע אחד כזה, יוחזר אחד מהם). יש כמובן צורך גם בפעולות גישה, כדי שהתוכנית תוכל לדעת את מצב הקלפים של השחקנים כדי שניתן יהיה לקבוע ניצחון. חשבו: מה צריכה לבצע הפעולה הבונה של עצם ממחלקה זו?

שאלה 11.8

במזללה "אוכל טעים" יש תפריט קבוע הכולל שתייה, מנה עיקרית ותוספת. השתייה יכולה להיות שתייה בכוס קטנה, בינונית או גדולה. המנה העיקרית יכולה להיות אחת מ-8 אפשרויות המשתנות כל יום, ולכן מסמנים אותה במספר בין 1 ל-8. התוספת היא צ'יפס, פירה או סלט. ארוחה רגילה כוללת שתייה בכוס בינונית, מנה עיקרית שמספרה 1 ופירה.

בצהרי כל יום נכנסים 20 סועדים למזללה. עובד הדלפק שואל את הסועד התורן באיזו ארוחה הוא מעוניין. אם הסועד אומר כי הוא מעוניין בארוחה רגילה הוא מקבל את הארוחה הרגילה. אחרת העובד שואל את הסועד איזה גודל של כוס שתייה, איזו מנה עיקרית ואיזו תוספת הוא רוצה להזמין. לפני ההגשה חוזר העובד על פרטי הארוחה המוזמנת, ומוסיף "בתיאבון".
פתחו וממשו תוכנית לניהול המזללה. הפעולה הראשית של התוכנית תשתמש במחלקה "ארוחה".

הדרכה: הגדירו מחלקת ארוחה הכוללת את התכונות: גודל שתייה, מנה עיקרית ותוספת. הפעולה הבונה של עצם מהמחלקה תקבל את פרטי הארוחה ותעדכן אותם. לצורך עמידה בדרישת החזרה על פרטי הארוחה המוזמנת לפני ההגשה יש להגדיר במחלקת הארוחה גם פעולה המחזירה כמחרוזת את תיאור הארוחה.

שאלה 11.9
בשקית סוכריות יש סוכריות מארבעה צבעים שונים (אדום, צהוב, ירוק וחום). קיימות שקיות בשני גדלים: שקית קטנה ושקית גדולה. בשקית קטנה יש 20 סוכריות ובשקית גדולה 28.
אם מספר הסוכריות מכל צבע בשקית שווה, תיקרא השקית **מאוזנת**. אם מספר הסוכריות מכל צבע בשקית שונה, תיקרא השקית **בלתי מאוזנת**.
פתחו וממשו אלגוריתם, שיקלוט את מספר השקיות המיוצרות במפעל ביום מסוים, ולאחר מכן יקלוט עבור כל שקית את תכולתה. תכולת השקית תיקלט כמחרוזת המורכבת מהתווים r (אדום), y (צהוב) g (ירוק) ו-b (חום) המייצגים את צבעי הסוכריות בשקית. האלגוריתם יציג כפלט את מספר השקיות הקטנות ואת מספר השקיות הגדולות שיוצרו במפעל באותו יום ואת מספר השקיות הבלתי מאוזנת מבין אלו שיוצרו באותו יום (ללא קשר לגודלן).

הדרכה: כתבו מחלקה המגדירה שקית סוכריות, ולמחלקה פעולה בונה המקבלת מחרוזת המכילה את כל הצבעים של הסוכריות בשקית. לעצם מטיפוס שקית סוכריות יהיו התכונות הבאות: תכונה אחת עבור כל צבע לשמירת מספר הסוכריות בצבע זה, ותכונה לשמירת גודל השקית (קטנה או גדולה). הפעולה הבונה תאתחל את ערכי התכונות. הגדירו את הפעולות הנדרשות לצורך קבלת המידע המבוקש.

11.3 תכונות מורכבות

בעיה 5

מטרת הבעיה ופתרונה: היכרות עם תכונות מורכבות של עצמים: מערך כתכונה.

במשחק ניחושים קיים טופס ובו רשומים 5 מספרים שונים בין 1 ל-100 וכן תיאור הפרס שזוכים בו אם מנחשים את כל המספרים הרשומים על הטופס. במשחק זה הפרס הוא מיליון דולר. פתחו וממשו תוכנית המתארת את משחק הניחושים. ראשית התוכנית תיצור טופס ניחושים. לאחר מכן התוכנית תקלוט מהשחקן מספר ותבדוק אם הוא קיים בטופס. אם המספר קיים ועדיין לא נוחש, התוכנית תציג כפלט את ההודעה "ניחוש נכון", אחרת תוצג ההודעה "ניחוש שגוי". התוכנית תסתיים לאחר שהשחקן ינחש נכונה את כל המספרים בטופס, או לאחר 10 ניחושים. אם השחקן ניחש נכונה את כל המספרים בטופס, התוכנית תציג כפלט את ההודעה "ניצחת: זכית ב<הפרס>", ואחרת, תוצג ההודעה "הפסדת: לא זכית ב<הפרס>".

לצורך פתרון הבעיה נכתוב מחלקה המגדירה טופס למשחק הניחושים.

הגדרת המחלקה *טופס ניחושים*

הגדרת התכונות

על-פי הגדרת הבעיה נגדיר לעצם מהמחלקה טופס ניחושים את התכונות: פרס ומספרים. התכונה מספרים מייצגת את רשימת המספרים בטופס שאותם המשתתף צריך לנחש.

כיצד נשמור את רשימת המספרים? נשמור מערך של מספרים שלמים שגודלו 5. אולם לא נוכל להסתפק ברשימת המספרים מכיוון שעבור כל מספר עלינו לדעת אם המשתתף כבר ניחש מספר זה או לא. לכן, המחלקה תכיל עוד תכונה. תכונה זו תהיה מערך בוליאני וגם הוא בגודל 5, שיחווה עבור כל מספר אם ניחשו אותו או לא. כלומר אם ניחשו את המספר השלישי אז האיבר השלישי במערך הבוליאני יהיה **true**.

שימו ♥: כאשר מגדירים מחלקה, התכונות יכולות להיות מכל טיפוס שהוא, גם מערך. הגדרת מערך כתכונה נעשית כהגדרת כל תכונה מטיפוס אחר.

נבחר את המשתנים הבאים לתיאור התכונות:

- **prize** – מטיפוס מחרוזת, מייצג את הפרס שזוכים בו אם מנחשים את כל המספרים בטופס.
- **numbers** – מערך של שלמים שונים בין 1 ל-100, מייצג את המספרים שהמשתתף צריך לנחש.
- **guessedNumbers** – מערך בוליאני, מייצג עבור כל מספר אם כבר ניחשו אותו או לא.
- **SIZE** – קבוע מטיפוס שלם, מייצג את מספר המספרים הנמצאים בטופס, במקרה זה 5.

הגדרת הפעולות

- **פעולה בונה** – הפעולה הבונה תקבל כפרמטר את הפרס ותאתחל אותו. מה לגבי אתחול המספרים? הפעולה הבונה תגריל את המספרים בטופס ותאתחל את מערך המספרים בהתאם. בהתחלה עדיין לא ניחשו אף מספר ולכן המערך הבוליאני guessedNumbers יאותחל כולו ב-**false**. שימו לב, עלינו לדאוג לכך שכל המספרים שנגריל יהיו שונים זה מזה. לצורך כך, נעזר במערך בוליאני שישמור עבור כל מספר מ-1 עד 100 אם כבר הגרילו אותו או לא. מערך זה הוא משתנה עזר לצורך הגרלת המספרים, לכן נגדיר אותו כמשתנה מקומי בפעולה הבונה ולא כתכונה של העצם.
  שימו ♥: פעולת ההגרלה נמשכת עד שמגרילים 5 מספרים שונים, אנו מניחים שפעולה זו תסתיים.
- **בדיקת ניחוש** – לאחר שהמשתתף מנחש מספר, הוא מקבל הודעה אם הוא צדק בניחושו או טעה. נגדיר פעולה IsGoodGuess המקבלת מספר כפרמטר. הפעולה תחזיר **true** אם המספר מופיע בטופס ועדיין לא נוחש ו-**false** אחרת. בנוסף לכך, אם המספר מופיע בטופס ועדיין לא נוחש, הפעולה תעדכן ל-**true** את התא המתאים במערך guessedNumbers.
- **בדיקת ניצחון** – כיצד נדע אם המשתתף ניחש את כל המספרים? נגדיר פעולה בוליאנית בשם IsWin. הפעולה תחזיר **true** אם המשתתף ניחש את כל המספרים בטופס, כלומר כל ערכי התאים במערך guessedNumbers הם **true**, אחרת יוחזר **false**.
- **פעולת גישה להחזרת הפרס** – על מנת להדפיס את הפרס בסוף המשחק נגדיר פעולת גישה GetPrize, להחזרת הפרס. פעולה זו אינה מקבלת ערכים והיא מחזירה את הפרס.

מימוש המחלקה

```
/*
המחלקה טופס-למשחק-הניחושים
*/
public class GuessForm
{
    private string prize;      //שומר את הפרס
    private int[] numbers;     //מערך המספרים שאותם צריך לנחש
    private bool[] guessedNumbers;
    //מערך המחווה לגבי כל מספר האם ניחשו אותו כבר או לא//
    private const int SIZE = 5;  //קבוע לציון מספר המספרים שבטופס
    //  פעולה בונה - מקבלת את הפרס, מגרילה את המספרים בטופס
    //  ומאתחלת את המערך המחווה אם המספרים נוחשו
    public GuessForm(string prize)
    {
        this.prize = prize;
        numbers = new int[SIZE];
        guessedNumbers = new bool[SIZE];
        Random rnd = new Random();
        //מערך עזר בוליאני המשמש לבחירת מספרים שונים//
        bool[] nums = new bool[101];
        for (int i = 0; i <= 100; i++)
            nums[i] = false;
        int r;
        for (int i = 0; i < SIZE; i++)
        {
            r = rnd.Next(1,101);
            while (nums[r] == true)
                 r = rnd.Next(1,101);
            nums[r] = true;
            numbers[i] = r;
            guessedNumbers[i] = false;
        }
    }
    //פעולת גישה המחזירה את הפרס//
    public string GetPrize()
    {
        return prize;
    }
    //פעולה הבודקת האם כל המספרים בטופס נוחשו//
    //מחזירה אמת אם כן//
    public bool IsWin()
    {
        for (int i = 0; i < SIZE; i++)
             if (guessedNumbers[i] == false)
                  return false;
        return true;
    }
    //פעולה המקבלת כפרמטר מספר ובודקת האם הוא נמצא בין המספרים בטופס,//
    //מעדכנת את מערך המספרים שכבר נוחשו ומחזירה אמת אם המספר נמצא//
    //בטופס ועדיין לא נוחש, אחרת שקר//
```

```
    public bool IsGoodGuess(int guess)
    {
        for (int i = 0; i < SIZE; i++)
        {
            if (numbers[i] == guess)
            {
                if (! guessedNumbers[i])
                {
                     guessedNumbers[i] = true;
                     return true;
                }
                else
                    return false;
            }
        }
        return false;
    }
}//class GuessForm
```

שימו ♥: בפעולה הבונה אנו מצהירים על משתנים כמו rnd וכמו nums המשמשים להגרלת המספר ולבדיקת כפילויות. משתנים אלה מוצהרים בתוך התחום של הפעולה הבונה. משתנים כאלה נקראים **משתנים מקומיים** (לוקאליים), משום שהם קיימים רק בתוך הפעולה. פעולות אחרות, אפילו של אותה מחלקה, לא יכולות להתייחס אליהם.

כאשר הפעולה מסתיימת המשתנים המקומיים שלה אינם קיימים יותר. כאשר תופעל הפעולה שוב יוקצה שוב שטח זיכרון עבור כל משתנה מקומי (שאינו בהכרח אותו שטח שהוקצה בהפעלה קודמת).

הדבר דומה למשתנה הבקרה שאנו מגדירים בתוך לולאת **for**. למשל בהוראה:

```
for (int i = 0; i < SIZE; i++)
```

מוצהר משתנה i המשמש בתוך הלולאה, אך מפסיק להתקיים עם סיומה.

למעשה בתוך כל תחום המוגדר בסוגריים מסולסלים (כמו למשל גוף של לולאה או גוף של הוראה לביצוע-בתנאי) ניתן להצהיר על משתנים מקומיים לאותו תחום. משתנים אלה מפסיקים להתקיים עם סיום ביצוע התחום.

הגדרת הפעולה הראשית

פירוק לתת-משימות

המשימה של הפעולה הראשית היא לממש את משחק הניחושים. את משימתה של הפעולה הראשית ניתן לפרק לתת-משימות, באופן הבא:

1. יצירת טופס ניחושים ואתחולו בפרס המתאים
2. המשחק עצמו
3. הצגת הפלט

בחירת משתנים

prize– משתנה מטיפוס מחרוזת, מייצג את הפרס.

guessForm – עצם מטיפוס "טופס ניחושים", מייצג את הטופס למשחק

counter – משתנה מטיפוס שלם, שומר את מספר הניחושים עד כה
guess – משתנה מטיפוס שלם, לתוכו נקלט הניחוש הנוכחי

האלגוריתם

1. צור עצם מטיפוס "טופס ניחושים" ואתחל את הפרס
2. כל עוד לא ניחשו כל המספרים וגם מונה הניחושים קטן מ-10 בצע:
   2.1 הגדל את מונה הניחושים ב-1
   2.2 קלוט מהמשתמש את הניחוש והשם ב-guess
   2.3 אם הניחוש מופיע בטופס
      2.3.1 הצג כפלט "ניחוש נכון"
   2.4 אחרת
      2.4.1 הצג כפלט "ניחוש שגוי"
3. אם כל המספרים ניחשו
   3.1 הצג כפלט: ניצחת: זכית ב<הפרס>
4. אחרת
   4.1 הצג כפלט: הפסדת: לא זכית ב<הפרס>

מימוש

```
/*
     המחלקה הראשית הממששת את משחק הניחושים
*/
using System;
public class GuessGame
{
    public static void Main()
    {
        string prize = "1 million dollars";
        GuessForm guessForm = new GuessForm(prize);
        int guess;
        int i = 0;
        while ((i < 10) && (guessForm.IsWin() == false))
        {
            i++;
            Console.Write("Please enter your guess: ");
            guess = int.Parse(Console.ReadLine());
            if (guessForm.IsGoodGuess(guess))
                Console.WriteLine("Good Guess");
            else
                Console.WriteLine("Bad Guess");
        }
        if (guessForm.IsWin())
            Console.WriteLine("You win: {0}",guessForm.GetPrize());
        else
            Console.WriteLine("You lose:{0}",guessForm.GetPrize());
    }// Main
}// GuessGame
```

סוף פתרון בעיה 5

שאלה 11.10

בתחרות קפיצה לגובה כל משתתף קופץ 5 קפיצות. כתבו מחלקה המגדירה קופץ לגובה. עצם מהמחלקה ישמור את שמו של הקופץ ואת תוצאות הקפיצות שלו.

פתחו וממשו תוכנית לאימון בקפיצה לגובה. התוכנית תקלוט את שמו של הקופץ. לאחר מכן תבקש התוכנית מהקופץ להכניס בכל פעם את תוצאת קפיצתו. (שימו לב שתוצאה של קפיצה לגובה היא מספר ממשי, למשל 1.15 מטר).

התוכנית תציג כפלט את שמו של הקופץ יחד עם תוצאת הקפיצה הטובה ביותר שלו.

שאלה 11.11

עזרו למורה נחמה להציג את הממוצע הכיתתי במקצועות חשבון ועברית. כתבו מחלקה המגדירה את ציוני הכיתה. בכיתה 20 תלמידים. עבור כל תלמיד המורה תכניס את ציונו בחשבון ואחר כך בעברית.

הדרכה: הגדירו במחלקה את הפעולות הבאות: פעולה לעדכון נתוני תלמיד המקבלת מספר סידורי של תלמיד ואת ציוניו בשני המקצועות, ופעולות המחזירות את הממוצע הכיתתי בכל מקצוע.

שאלה 11.12

במסעדת "יאמיאמי" מגישים: סלט, ספגטי, שניצל, המבורגר, מרק ועוגה. בכל יום בשעה 13:00 מגיעים סועדים רבים למסעדה. הטבח מוכן להתחיל לבשל רק לאחר שכל ההזמנות נעשו. כתבו מחלקה המגדירה את ניהול ההזמנות במסעדת "יאמיאמי". עצם מהמחלקה יכיל את רשימת המאכלים שמגישים במסעדה ובעבור כל מאכל כמה סועדים הזמינו אותו. פתחו וממשו תוכנית שתקלוט מכל סועד את הזמנתו (מספר בין 1 ל-6 המציין בהתאמה את המנות האפשריות) ותציג כפלט את כל המנות שצריך הטבח להכין וכמה מכל מנה. סוף הקלט יצוין בהזמנת מנה שמספרה הוא 0.

בעיה 6

מטרת הבעיה ופתרונה: היכרות עם מבני נתונים בסיסיים של עצמים (מערך).

כתבו תוכנית לניהול תחרות קפיצה לגובה. התוכנית תקלוט את מספר המשתתפים בתחרות. לאחר מכן ייקלטו נתוני המשתתפים: עבור כל קופץ לגובה תקלוט התוכנית את שמו של הקופץ, את מספר תעודת הזהות שלו ואת גובה קפיצתו.

התוכנית תציג כפלט את שמותיהם של כל זוגות המתחרים שגובה קפיצתם זהה, ואת גובה הקפיצה.

למשל, אם בתחרות יש 4 משתתפים, ולהם הנתונים הבאים:

שם	**ת"ז**	**גובה קפיצה**
פלוטו	1111	1.25
מיני	1112	2.02
דונלד	1120	1.25
מיקי	1121	1.25

אז בפלט צריכים להיות מוצגים הזוגות הבאים:

- פלוטו ודונלד קפצו לגובה 1.25
- פלוטו ומיקי קפצו לגובה 1.25

הגדרת המחלקה *קופץ לגובה*

הגדרת התכונות

על פי הגדרת הבעיה לעצם מהמחלקה קופץ לגובה יש התכונות הבאות: שם מלא, ת״ז וגובה קפיצה.

כיצד נשמור את מספר תעודת הזהות של הקופץ? מכיוון שמספר תעודת הזהות משמש כמזהה, כלומר הוא איננו מספר שאמורים לבצע עליו פעולות חשבוניות, נשמור את מספר תעודת הזהות כמחרוזת.

נבחר את המשתנים הבאים לתיאור התכונות:

- **name** – מחרוזת המייצגת את שם הקופץ.
- **id** – מחרוזת המייצגת את מספר תעודת הזהות של הקופץ.
- **jumpHeight** – משתנה מטיפוס ממשי המייצג את גובה הקפיצה של הקופץ.

הגדרת הפעולות

- **פעולה בונה** – הפעולה הבונה תקבל כפרמטרים את שם הקופץ ואת מספר תעודת הזהות שלו ותאתחל את התכונות המתאימות. מה לגבי אתחול גובה הקפיצה של הקופץ? בהתחלה הקופץ עדיין לא קפץ. לכן גובה הקפיצה יאותחל ב-0.
- **עדכון גובה קפיצה** – לאחר שהקופץ יקפוץ נרצה לעדכן את גובה הקפיצה שלו. לכן נרצה להגדיר פעולת גישה SetJumpHeight המעדכנת את גובה הקפיצה.
- **קבלת גובה קפיצה** – לצורך מציאת כל זוגות הקופצים שגובה קפיצתם זהה, נגדיר פעולת גישה GetJumpHeight, המחזירה את גובה הקפיצה של הקופץ.
- **קבלת שם קופץ** – על מנת להציג את שמותיהם של הקופצים שגובה קפיצתם זהה נגדיר פעולת גישה GetName המחזירה את שמו של הקופץ.

מימוש המחלקה

```
/*
מחלקת קופץ לגובה
*/
public class Jumper
{
      // תכונות הקופץ
      private string name;          // שם
      private string id;            // ת"ז
      private double jumpHeight;    // גובה קפיצה
      // הפעולה הבונה
      public Jumper(string name, string id)
      {
            this.name = name;
            this.id = id;
            jumpHeight = 0;
      }
      // פעולת גישה: מחזירה את שם הקופץ
      public string GetName()
```

```
        {
             return name;
        }
        //פעולת גישה: מחזירה את גובה הקפיצה של הקופץ
        public double GetJumpHeight()
        {
             return jumpHeight;
        }
        //פעולת גישה: מעדכנת את גובה הקפיצה של הקופץ
        public void SetJumpHeight(double jumpHeight)
        {
             this.jumpHeight = jumpHeight;
        }
} // class Jumper
```

הגדרת הפעולה הראשית

פירוק לתת-משימות

המשימה של הפעולה הראשית היא לקלוט את מספר הקופצים בתחרות. לאחר מכן, לקלוט את הנתונים של הקופצים ולהציג את שמותיהם של כל זוגות המתחרים שגובה קפיצתם זהה ואת גובה הקפיצה. את משימתה של הפעולה הראשית ניתן לפרק לתת-משימות, באופן הבא:

1. קליטת מספר הקופצים
2. קליטת הנתונים של הקופצים
3. הצגת כל זוגות המתחרים שגובה קפיצתם זהה והצגת גובה הקפיצה.

בחירת משתנים

את הקופצים נשמור במערך שבו כל איבר יהיה מטיפוס "קופץ לגובה". אלה המשתנים הדרושים:

jumpers – מערך מטיפוס "קופץ לגובה", מייצג את הקופצים לגובה אשר משתתפים בתחרות.
jumpersNum – משתנה מטיפוס שלם, שומר את מספר הקופצים.
jumpHeight – משתנה מטיפוס ממשי, שומר את גובה הקפיצה.
jumperName – מחרוזת, שומרת את שם הקופץ.
jumperId – מחרוזת, שומרת את מספר תעודת הזהות של הקופץ.

האלגוריתם

משימת קליטת נתוני הקופצים דומה למשימות קלט מבעיות קודמות. ההבדל היחיד הוא שבמקרה זה הקופצים הם עצמים ולא משתנים פשוטים. לכן עבור כל קופץ ניצור עצם מטיפוס "קופץ לגובה" ונאתחל אותו בשם ובמספר תעודת הזהות שהתקבלו עבורו כקלט. העצם שנוצר יישמר במערך הקופצים. לאחר מכן נקלוט את גובה הקפיצה עבור הקופץ ונעדכן אותו.

? כיצד נבצע את התת-משימה השלישית? כיצד נמצא את כל זוגות המתחרים אשר גובה קפיצתם זהה?

נעבור על המערך, ועבור כל קופץ נשווה את גובה קפיצתו לגובה קפיצתם של הקופצים האחרים. כלומר נעבור על המערך בלולאה מקוננת. הלולאה החיצונית תגדיר קופץ ועבור כל קופץ נבצע לולאה פנימית, שעוברת על רשימת הקופצים ומחפשת קופצים אחרים שגובה קפיצתם זהה לזה שלו. נרצה להשוות כל זוג קופצים פעם אחת בלבד. אחרי שהשווינו את גובה הקפיצה של הקופץ

הראשון לזה של הקופץ השני, לא נרצה להשוות אחר כך את גובה הקפיצה של הקופץ השני לזה של הראשון. לכן הלולאה הפנימית לא תתחיל בכל פעם מהקופץ הראשון, אלא מהקופץ שבא אחרי הקופץ התורן בלולאה החיצונית.

האלגוריתם ליישום התת-משימה השנייה הוא:

1. עבור כל i מ-0 עד מספר הקופצים פחות 2 בצע:
   1.1. עבור כל j מ-i+1 עד מספר הקופצים פחות 1 בצע:
      1.1.1. אם גובה הקפיצה של הקופץ ה-i במערך שווה לגובה הקפיצה של הקופץ ה-j במערך
         1.1.1.1. הצג את שם הקופץ ה-i והקופץ ה-j במערך ואת גובה קפיצתם.

מימוש

```
/*
    מחלקה ראשית המממשת תחרות קפיצה לגובה
*/
using System;
public class JumperContest
{
    public static void Main()
    {
        // הגדרה והקצאת משתנים
        int jumpersNum;
        Jumper[] jumpers;
        double jumpHeight;
        string jumperName;
        string jumperId;
        Console.Write("Enter number of jumpers: ");
        jumpersNum = int.Parse(Console.ReadLine());
        jumpers = new Jumper[jumpersNum];
        for (int i = 0; i < jumpersNum; i++)
        {
            Console.Write("Enter Jumper Name: ");
            jumperName = Console.ReadLine();
            Console.Write("Enter Jumper Id: ");
            jumperId = Console.ReadLine();
            jumpers[i]= new Jumper(jumperName, jumperId);
            //יצירת כל איבר במערך בתורו
            Console.Write("Enter Jump Height: ");
            jumpHeight = int.Parse(Console.ReadLine());
            jumpers[i].SetJumpHeight(jumpHeight);
        }
        // פלט
        for (int i = 0; i < jumpersNum-1; i++)
        {
            for (int j = i+1; j < jumpersNum; j++)
            {
                if (jumpers[i].GetJumpHeight() ==
                        jumpers[j].GetJumpHeight())
                {
```

```
                    Console.WriteLine("{0} and {1} jumped the same
                        height: {2}", jumpers[i].GetName(),
                        jumpers[j].GetName(),
                        jumpers[i].GetJumpHeight());
                } // if
            } // for j
        } // for i
    } // Main
} // class JumperContest
```

סוף פתרון בעיה 6

בפתרון בעיה 6 ראינו שניתן להגדיר מערך מכל טיפוס שהוא, בפרט מערך של עצמים ממחלקות שהגדרנו אנחנו. הגדרת מערך מטיפוס מחלקה מסוימת נעשה כהגדרת כל מערך מטיפוס אחר. עם זאת מערך של עצמים הוא למעשה מערך של הפניות לעצמים, ולכך יש חשיבות רבה, כפי שנדגים כעת. נתבונן בקטע התוכנית הבא:

```
Jumper[] jumpers = new Jumper[10];
Jumper j = new Jumper("Yoyo","99999");
jumpers[0] = j;
```

כתוצאה מביצוע קטע זה נוצר עצם אחד מסוג קופץ, אך לעצם זה יש שתי הפניות – המשתנה `j` ו-`jumpers[0]` – שניהם מפנים לאותו הקופץ ששמו `Yoyo`.
אם נעדכן את גובה הקפיצה באופן הבא:

```
j.SetJumpHeight(1.85);
```

נוכל לראות את השינוי גם דרך `jumpers[0]` כי הוא מפנה לאותו העצם.
בעקבות הוראת הפלט הבאה:

```
Console.WriteLine("jump height = {0}", jumpers[0].GetJumpHeight());
```

יוצג הפלט:

```
jump height = 1.85
```

מדוע? הרי לכאורה לא ביצענו שינוי באיברי המערך `jumpers`! הסיבה היא שהמערך `jumpers` הוא מערך של עצמים. כפי שאמרנו; כאשר איברי המערך הם מטיפוס מחלקה כלשהי, המערך הוא מערך של **הפניות** לעצמים. כלומר כל איבר במערך הוא הפניה לעצם. בקטע התוכנית שלעיל התבצע שינוי בעצם ש-`j` מפנה אליו. מכיוון שהתא הראשון במערך מפנה לאותו העצם, כל שינוי שנעשה באמצעות `j` ייראה גם מתוך המערך, ולחילופין כל שינוי שנעשה באמצעות `jumpers[0]` ייראה דרך `j`.

אם נרצה למנוע יצירת שתי הפניות לאותו העצם, נוכל ליצור את העצמים כפי שעשינו בפתרון הבעיה שהוצג לעיל:

```
jumpers[i] = new Jumper("Yoyo","99999");
```

בצורה זו ההפניה היחידה לקופץ נמצאת במערך.

שאלה 11.13

פתחו וממשו תוכנית למשחק בול פגיעה. התוכנית תגריל מספר בן 4 ספרות (ללא חזרות) והמשתמש ינסה לנחש את המספר. עבור כל ניחוש התוכנית תציג כמה ספרות הן "בול" וכמה ספרות הן רק "פגיעה".

נגדיר "בול" כשהמשתמש ניחש את הספרה במקומה הנכון ו"פגיעה" כשניחש ספרה המופיעה במספר שהוגרל אך לא במיקומו הנכון.

לדוגמא: המספר שבחרה התוכנית הוא 3456

המספר שניחש המשתמש הוא 2465

לכן הפלט יהיה: בול אחד (הספרה 4) ו- 2 פגיעות (הספרות 5 ו 6)

התוכנית תמשיך לקלוט מהמשתמש מספרים עד אשר ינחש את המספר, ורק אז תסתיים.

הדרכה: כתבו מחלקה המגדירה את המספר הסודי שרוצים לנחש. הפעולה הבונה תגריל מספר בן 4 ספרות ותשמור אותו במערך של שלמים בגודל 4 – כל תא במערך ייצג ספרה במספר. הגדירו במחלקה, פעולה המקבלת כפרמטר מספר בן 4 ספרות, בודקת כמה מהספרות הן בול וכמה קליעה ומחזירה מחרוזת עם התוצאה. במקרה שהניחוש נכון תוחזר המחרוזת "finished".

שאלה 11.14

בתחרות ניווטים הוחלט שהמשתתפים שיעלו לשלב הגמר הם אלה שהגיעו ליעד בזמן הנמוך מזמן ההגעה הממוצע של כל המשתתפים.

עליכם לפתח אלגוריתם שיקלוט את מספר המשתתפים, ולכל משתתף את הנתונים הבאים: שם, כתובת, מספר תעודת זהות וזמן ההגעה ליעד. האלגוריתם יציג כפלט את פרטי המשתתפים שיעלו לשלב הגמר, את כתובותיהם ואת מספרי תעודות הזהות שלהם.

א. הגדירו את המחלקה המתאימה למשתתף בתחרות, את כל התכונות ואת פעולות הגישה הנדרשות.

ב. פתחו אלגוריתם שהקלט שלו הוא זמן ההגעה ליעד של כל משתתף והפלט שלו הוא רשימת המשתתפים שזמן ההגעה שלהם ליעד נמוך מזמן ההגעה הממוצע.

ג. כתבו תוכנית הקולטת את מספר המשתתפים בתחרות. ממשו את האלגוריתם שפיתחתם בסעיף ב, בשימוש במחלקה שהגדרתם בסעיף א.

שאלה 11.15

כתבו תוכנית למציאת הילדים שלהם יש שמות ייחודיים בכיתה. התוכנית תקלוט את מספר הילדים בכיתה. לאחר מכן, תקלוט התוכנית עבור כל ילד את שמו הפרטי, את המקצוע האהוב עליו ואת גילו. התוכנית תציג את הנתונים של הילדים אשר שמותיהם ייחודיים בכיתה. ילד הוא בעל שם ייחודי אם בכיתה אין עוד ילד בשם זהה.

שאלה 11.16

א. נתונה התוכנית הבאה:

```
using System;
public class Animal
{
    private string kind = ""; // סוג
    public void SetKind(string Kind)
    {
        this.kind = Kind;
    }
    public string GetKind()
    {
        return kind;
    }
} // class Animal
```

```
public class MyAnimals
{
    public static void Main()
    {
         Animal ani = new Animal();
         Animal [] myAnimals = new Animal[3];
         ani.SetKind("dog");
         myAnimals[0] = ani;
         ani.SetKind("cat");
         myAnimals[1] = ani;
         ani.SetKind("fish");
         myAnimals[2] = ani;
         Console.WriteLine("My animals are {0} {1} {2}",
              myAnimals[0].GetKind(), myAnimals[1].GetKind(),
              myAnimals[2].GetKind());
    } // Main
} // class MyAnimals
```

מה הפלט שיוצג בסוף התוכנית? השלימו:

```
My animals are _________  _________  ________
```

ב. תקנו את הפעולה הראשית (אך לא את הוראת הפלט) כך שבסוף התוכנית תוצג ההודעה:

```
My animals are dog cat fish
```

שאלה 11.17

נתון קטע התוכנית הבא:

```
int[] intArray = new int[1];
int i = 10;
intArray[0] = i;
Console.WriteLine("intArray[0] is {0}", intArray[0]);
i = 5;
Console.WriteLine("intArray[0] is {0}", intArray[0]);
```

כתבו מהו הפלט של קטע התוכנית.

בעיה 7

מטרת הבעיה ופתרונה: היכרות עם מחלקת שירות ועם פעולות סטטיות.

> פתחו וממשו מחשבון המבצע פעולות מתמטיות מתקדמות הכוללות: העלאת מספר בחזקה, בדיקה אם מספר הוא חזקה של מספר אחר וחישוב סכום הספרות של מספר.
> פתחו תוכנית לשימוש במחשבון. התוכנית תציג לפני המשתמש את הפעולות שהמחשבון יכול לבצע. המשתמש יבחר את הפעולה הרצויה ויספק את הקלט המתאים לפעולה. בסיום ביצוע הפעולה תציג התוכנית פלט הכולל את פרטי התרגיל שבוצע ואת התוצאה. למשל עבור העלאת 2 בחזקת 5 יוצג הפלט: 2^5=32.

הגדרת המחלקה מחשבון

על פי הגדרת הבעיה נראה שלעצם מסוג מחשבון אין תכונות כלשהן. מחשבון יכול לבצע פעולות חשבוניות על מספרים הניתנים לו כפרמטרים. עד כה ראינו מחלקות המורכבות מתכונות ומפעולות. אולם יש מקרים שמחלקה תכיל רק פעולות. במקרים כאלו, נקרא למחלקה "מחלקת

שירות", מפני שהיא מכילה אוסף של שירותים (פעולות) שאינם פועלים על עצם כלשהו. פעולות אלה יכולות לקבל פרמטרים ולהחזיר ערך אך הן לא פועלות על תכונות. פעולות אלה נקראות **פעולות מחלקה** (class methods) או **פעולות סטטיות** (static methods).

המחלקה Math שהשתמשנו בה בעבר, היא מחלקת שירות הכוללת פעולות סטטיות המאפשרות לבצע חישובים מתמטיים שונים. כדי להשתמש בפעולות סטטיות איננו צריכים ליצור עצם מהמחלקה אלא להשתמש במחלקה עצמה ולהפעיל עליה ישירות את הפעולות, למשל:

```
int x = Math.Sqrt(4);
```

לעומת זאת כאשר רוצים להפעיל פעולה של עצם, כלומר פעולה שאיננה סטטית, אנו יוצרים עצם מהמחלקה (באמצעות ההוראה **new**) ועליו אנו מפעילים את הפעולה, למשל:

```
Random rnd = new Random();
int val = rnd.Next(6);
```

במחלקת שירות כל הפעולות מוגדרות כפעולות סטטיות. כלומר כעת כותרות הפעולות ייראו כך:

```
public static טיפוס-הערך-המוחזר שם-הפעולה(רשימת פרמטרים)
```

כעת, ניצור מחלקת שירות בשם מחשבון.

הגדרת הפעולות

- **העלאת מספר בחזקה**: הפעולה תקבל שני פרמטרים מטיפוס שלם, המייצגים בסיס ומעריך חיובי. הפעולה תחשב ותחזיר את הבסיס בחזקת המעריך. החישוב יתבצע באמצעות פעולת הכפל ולא באמצעות הפעולה Pow של המחלקה Math.
- **האם מספר הוא חזקה של מספר אחר**: הפעולה תקבל שני פרמטרים מטיפוס שלם ותבדוק אם המספר הראשון הוא חזקה של המספר השני. אם כן הפעולה תחזיר **true** אחרת יוחזר **false**. למשל אם התקבלו המספרים 25 ו-5 יוחזר הערך **true** כי 25 הוא חזקה של 5.
- **חישוב סכום הספרות של מספר**: הפעולה תקבל מספר שלם ותחזיר את סכום ספרותיו.

שימו ♥: מכיוון שזוהי מחלקת שירות אין צורך בפעולה בונה.

מימוש המחלקה

```
/*
מחלקת שירות: מחשבון
*/
public class Calculator
{
    //פעולה המקבלת בסיס ומעריך ומחזירה את הבסיס בחזקת המעריך
    public static int Power(int b, int exponent)
    {
        if (exponent == 0)
            return 1;
        int result = b;
        for (int i = 1; i < exponent; i++)
            result = result * b;
        return result;
    }
    // :פעולה המקבלת שני מספרים שלמים ומחזירה
    // האם הראשון הוא חזקה של השני
    public static bool IsPower(int result, int b)
    {
```

```
        double div = result;
        if (result == 1)
             return true;
        if (result % b != 0)
             return false;
        while(result > 1)
        {
             div = div / b;
             result = result / b;
        }
        return (div == 1);
    }
    //פעולה המקבלת מספר שלם ומחזירה את סכום ספרותיו
    public static int DigitSum(int num)
    {
        int sum = 0;
        while (num != 0)
        {
             sum = sum + num % 10;
             num = num / 10;
        }
        return sum;
    }
}// Calculator
```

שימו ♥: לצורך ביצוע הפעולה DigitSum השתמשנו בתבנית "פירוק מספר לספרותיו".

הגדרת הפעולה הראשית

המשימה של הפעולה הראשית היא להציג את הפעולות שניתן לבצע במחשבון, לקלוט את הפעולה שמבקש המשתמש לבצע, לקלוט מהמשתמש את הפרמטרים הנחוצים לביצוע הפעולה ולהציג את הפרמטרים שנקלטו ואת תוצאת הפעולה.

בחירת משתנים

operation – משתנה מטיפוס שלם, מייצג את הפעולה החשבונית שהמשתמש רוצה לבצע.

num – משתנה מטיפוס שלם, מייצג את הפרמטר הראשון לפעולה שנבחרה.

num1 – משתנה מטיפוס שלם, מייצג (במקרה הצורך) את הפרמטר השני לפעולה שנבחרה.

מימוש

```
/*
       מחלקה ראשית המשתמשת במחלקת המחשבון
*/
using System;
public class CalculatorTest
{
    public static void Main()
    {
        int operation;
        int num;
        int num1;
```

```
        Console.WriteLine("Please enter the number of the "+
                          "operation you would like to perform:");
        Console.WriteLine("1: power");
        Console.WriteLine("2: is power");
        Console.WriteLine("3: digit sum");
        operation = int.Parse(Console.ReadLine());
        switch (operation)
        {
            case 1:
                Console.Write("Please enter the base number: ");
                num = int.Parse(Console.ReadLine());
                Console.Write("Please enter the exponent: ");
                num1 = int.Parse(Console.ReadLine());
                Console.WriteLine("{0}^{1}={2}", num, num1,
                                  Calculator.Power(num,num1));
            break;
            case 2:
                Console.Write("Please enter the result: ");
                num = int.Parse(Console.ReadLine());
                Console.Write("Please enter the base: ");
                num1 = int.Parse(Console.ReadLine());
                if (Calculator.IsPower(num,num1))
                    Console.WriteLine("{0} is a power of {1}", num
                                                         , num1);
                else
                    Console.WriteLine("{0} is not a power of {1}",
                                                      num, num1);
            break;
            case 3:
                Console.Write("Please enter a number: ");
                num = int.Parse(Console.ReadLine());
                Console.WriteLine("The digit sum of {0} is {1}",
                                num, Calculator.DigitSum(num));
            break;
            default:
                Console.WriteLine("Wrong option");
            break;
        }// Switch
    }// Main
}// ClaculatorTest
```

סוף פתרון בעיה 7

שימו ♥: הפעולות במחלקת המחשבון מוגדרות כפעולות סטטיות, לכן השתמשנו בהן **ללא** יצירת עצם מסוג מחשבון. זימון פעולות סטטיות נעשה ישירות דרך שם המחלקה:

```
שם המחלקה.שם הפעולה(פרמטרים)
```

למשל:

```
Calculator.DigitSum(num)
```

נסתכל על מימוש הפעולה DigitSum במחלקה "מחשבון". הפעולה מקבלת פרמטר בשם num שהוא המספר וסוכמת את ספרותיו. לצורך ביצוע הפעולה, חילקנו שוב ושוב את num ב-10 עד שערכו היה שווה לאפס.

בפעולה הראשית, קולטים ערך למשתנה num ומעבירים אותו כפרמטר לפעולה DigitSum. לאחר סיום הפעולה המשתנה num מוצג בפלט וניתן לראות שערכו לא השתנה. מדוע? כאשר אנו מעבירים משתנה מטיפוס פשוט כפרמטר, ערכו של המשתנה הוא זה שמועבר ולא המשתנה עצמו. כלומר, כל שינוי בתוך הפעולה על הפרמטר משנה את הפרמטר בלבד אך לא את המשתנה שהועבר. לצורה זו של העברת פרמטרים קוראים **call by value** כי **הערך** של המשתנה הוא זה שמועבר ולא המשתנה עצמו. נרחיב בנושא זה בבעיה 9 בפרק זה.

שאלה 11.18

א. נתונות המחלקות הבאות:

```
using System;
public class Doubler
{
    public static void DoubleIt(int num)
    {
         num = num * 2;
         Console.WriteLine("Doubled: num = {0}", num);
    }
}// Doubler

public class DoublerTest
{
    public static void Main()
    {
         int num = 2;
         Console.WriteLine("num is {0}", num);
         Doubler.DoubleIt(num);
         Console.WriteLine("num after double is {0}", num);
    } // Main
}//DoublerTest
```

מה הפלט שיוצג כתוצאה מביצוע הוראת הפלט האחרונה בתוכנית? השלימו:

```
num after double is _________
```

זכרו שפרמטרים ב-C# מועברים על פי ערך (call by value) ולכן ערכו של `num` לא ישתנה כתוצאה מביצוע הפעולה.

ב. הדרך היחידה שבה הפעולה `DoubleIt` יכולה להכפיל את ערכו של `num` היא בהחזרת הערך `num*2` והשמת הערך המוחזר במשתנה `num`. שנו את הפעולה `DoubleIt` (ואת אופן הפעלתה), כך שהפעולה הראשית תקבל את ערכו של num מוכפל ב-2. כלומר, כתוצאה מביצוע הוראת הפלט האחרונה בתוכנית תוצג ההודעה:

```
num after double is 4
```

שאלה 11.19

פתחו וממשו מחלקת שירות למחרוזות. פעולות המחלקה הן:

1. האם פלינדרום – פעולה המקבלת מחרוזת ומחזירה אם היא פלינדרום. מחרוזת היא פלינדרום כאשר רצף התווים משמאל לימין זהה לרצף התווים מימין לשמאל. למשל המחרוזות "aba" ו-"xyyx" הן פלינדרום ואילו המחרוזת "abab" אינה פלינדרום.
2. האות השכיחה ביותר – פעולה המקבלת מחרוזת ומחזירה את האות השכיחה ביותר במחרוזת. אם יש כמה כאלה תוחזר הראשונה מביניהן.

3. הפיכת המחרוזת – פעולה המקבלת מחרוזת ומחזירה את המחרוזת ההפוכה.

בעיה 8

מטרת הבעיה ופתרונה: העברת עצם כפרמטר לפעולה.

החייזר בונבון הלך לאיבוד בגלקסיה, מלך הגלקסיה מצא אותו והוא מוכן להחזירו רק למי שבונבון יזהה כקרוב משפחתו. רבים באו בטענות שהם קרובי משפחתו של בונבון.

חייזר קובע שחייזר אחר הוא קרוב משפחה רק אם לשניהם שם משפחה זהה, שניהם הגיעו מאותו כוכב ולשניהם מאכל אהוב זהה.

שם המשפחה של בונבון הוא אלף, הוא הגיע מכוכב מלמק והמאכל האהוב עליו הוא פיצה.

פתחו וממשו תוכנית המנהלת את מציאת קרוב המשפחה של בונבון. התוכנית תקלוט את שמו של הטוען לקרבה, את הכוכב שהוא בא ממנו ואת המאכל האהוב עליו.

אם בונבון יזהה קרוב משפחה מבין אחד הטוענים לקרבה, התוכנית תודיע: קרוב המשפחה נמצא, תודה על השתתפותכם. אם לאחר 50 קרובים שנבדקו לא נמצאה התאמה, התוכנית תודיע שהחיפוש הסתיים.

הגדרת המחלקה *חייזר*

הגדרת התכונות

על פי הגדרת הבעיה לעצם מהמחלקה חייזר (כלומר לבונבון ולקרוביו) יש התכונות הבאות:
שם פרטי, שם משפחה, כוכב ומאכל אהוב.
נבחר את המשתנים הבאים לתיאור התכונות:

- **firstName** – מחרוזת. שמו הפרטי של החייזר.
- **lastName** – מחרוזת. שם המשפחה של החייזר.
- **planet** – מחרוזת. כוכב המוצא של החייזר.
- **favoriteFood** – מחרוזת. המאכל האהוב על החייזר.

הגדרת הפעולות

- **פעולה בונה** – הפעולה תקבל כפרמטרים את שם החייזר, את הכוכב שהגיע ממנו ואת המאכל האהוב עליו. הפעולה הבונה תאתחל את תכונות החייזר לפי הפרמטרים שהתקבלו.
- **האם קרוב משפחה** – לכל חייזר הטוען לקרבת משפחה צריך לבדוק אם הוא אכן קרוב משפחה לפי שם משפחתו, לפי הכוכב שהגיע ממנו ולפי המאכל האהוב עליו. כלומר הפעולה מקבלת כפרמטר חייזר אחר הטוען לקרבה – היא תחזיר **true** אם שניהם קרובי משפחה ו-**false** אחרת.

מימוש המחלקה

```
/*
מחלקת חייזר
*/
public class Alien
{
    private string firstName;
    private string lastName;
    private string planet;
    private string favoriteFood;
    public Alien (string firstName, string lastName, string planet,
                                            string favoriteFood)
    {
          this.firstName = firstName;
          this.lastName = lastName;
          this.planet = planet;
          this.favoriteFood = favoriteFood;
    }
    public bool IsRelative(Alien relative)
    {
          if (lastName==relative.lastName && planet==relative.planet
                              && favoriteFood==relative.favoriteFood)
               return true;
          else
               return false;
    }
}// Alien
```

הגדרת הפעולה הראשית

המשימה של הפעולה הראשית היא לקלוט בכל פעם את הטוען לקרבה לבונבון ולבדוק אם הוא אכן קרוב המשפחה של בונבון. אם כן, להציג את הפלט קרוב משפחה נמצא, תודה על השתתפותכם. אחרת להמשיך ולקלוט את הטוען לקרבה הבא בתור.

בחירת משתנים

bonbon – עצם מטיפוס חייזר. מייצג את החייזר בונבון אלף שהגיע ממלמק והמאכל האהוב עליו הוא פיצה.
relative – עצם מטיפוס חייזר. מייצג בכל פעם את הטוען לקרבה משפחתית לבונבון.
firstName – משתנה מטיפוס מחרוזת, מכיל את השם של הטוען לקרבה.
lastName – משתנה מטיפוס מחרוזת, מכיל את שם המשפחה של הטוען לקרבה .
planet – משתנה מטיפוס מחרוזת, מכיל את הכוכב ממנו הגיע הטוען לקרבה.
favoriteFood – משתנה מטיפוס מחרוזת, מכיל את המאכל האהוב על הטוען לקרבה.
foundRelative – משתנה בוליאני, קובע אם החייזר שנבדק הוא קרוב משפחה של בונבון

מימוש

```
/*
 מחלקה ראשית לחיפוש קרוב משפחה של בונבון
*/
using System;
public class AlienFamily
{
    public static void Main()
    {
         string firstName;
         string lastName;
         string planet;
         string favoriteFood;
         bool foundRelative = false;
         Alien bonbon = new Alien("Bonbon","Alf","Melmack","pizza");
         Alien relative;
         int counter = 0;
         while (!foundRelative && counter < 50)
         {
              Console.Write("---Looking for a relative--- ");
              counter ++;
              Console.Write("Enter first name: ");
              firstName = Console.ReadLine();
              Console.Write("Enter last name: ");
              lastName = Console.ReadLine();
              Console.Write("Where do you come from? ");
              planet = Console.ReadLine();
              Console.Write("What is your favorite food? ");
              favoriteFood = Console.ReadLine();
              relative = new
                   Alien(firstName,lastName,planet,favoriteFood);
              foundRelative = bonbon.IsRelative(relative);
         }
         if (foundRelative)
              Console.WriteLine("A relative was found. " +
                                  "Thank you for your participation");
         else
              Console.WriteLine("The search ended with no luck");
    }// Main
}// AlienFamily
```

סוף פתרון בעיה 8

בפתרון בעיה 8 העברנו עצם מסוג חייזר לפעולה IsRelative. כך, ניתן להעביר לפעולות גם עצמים כפרמטרים ולא רק משתנים.

שאלה 11.20

בכל בוקר מוציא מר גרבון גרב ממגירת הגרביים ומתקשה למצוא לו בן-זוג תואם. פתחו תוכנית שתעזור למר גרבון למצוא בן-זוג תואם. התוכנית תקלוט ממר גרבון את הגרב (גודל, צבע ודוגמה) ולאחר מכן תקלוט את בן-הזוג שהוא מנסה להתאים לגרב. התוכנית תסתיים כאשר יימצא גרב תואם. הגדירו מחלקה בשם "גרב" בעלת התכונות גודל, צבע ודוגמה. הגדירו פעולה במחלקה

שתקבל גרב אחר ותחליט אם הגרביים הם זוג. עליכם להחליט לפי שיקול דעתכם אילו טיפוסי נתונים מתאימים לתכונות גודל, צבע ודוגמה.

שאלה 11.21

הגדירו מחלקה המגדירה כיתה. תכונות הכיתה הן מספר התלמידים בכיתה ושם המורה. הגדירו לכיתה פעולה בשם "איחוד": הפעולה תקבל כפרמטר עצם אחר מהמחלקה כיתה ו"תאחד" אותו עם הכיתה הנוכחית. איחוד כיתה עם כיתה אחרת נעשה באמצעות עדכון מספר התלמידים בכיתה הנוכחית לסכום התלמידים בשתי הכיתות ושרשור שם המורה של הכיתה שהתקבלה כפרמטר לשם המורה של הכיתה הנוכחית. נתוני הכיתה שהתקבלה כפרמטר לא ישתנו.

בעיה 9

מטרת הבעיה ופתרונה: העברת עצם כפרמטר לפעולה – העמקה.

> ניתן להגדיר חשבון בנק באמצעות מחלקה הכוללת את התכונות הבאות: שם בעל החשבון והיתרה בחשבון. הפעולות שניתן לבצע בחשבון בנק הן: הפקדה, משיכה והעברה. פעולת ההעברה מקבלת כפרמטר את חשבון הבנק שרוצים אליו להעביר כסף ואת סכום הכסף להעברה.
>
> את החשבון מאתחלים באמצעות פעולה בונה המקבלת את שם בעל החשבון ואת היתרה ההתחלתית הנמצאת בחשבון.
>
> פתחו וממשו את המחלקה חשבון בנק.

הגדרת המחלקה *חשבון בנק*

הגדרת התכונות

על פי הגדרת הבעיה לעצם מהמחלקה חשבון בנק יש התכונות הבאות: שם בעל החשבון ויתרה. נבחר את המשתנים הבאים לתיאור התכונות:

- **name** – מחרוזת המייצגת את שם בעל החשבון.
- **balance** – משתנה מטיפוס ממשי המייצג את היתרה בחשבון.

הגדרת הפעולות

- **פעולה בונה** – הפעולה הבונה תקבל כפרמטרים את שם בעל החשבון ואת היתרה. הפעולה הבונה תאתחל את התכונות לפי הפרמטרים שהתקבלו.
- **משיכה** – הפעולה תקבל כפרמטר את סכום הכסף שרוצים למשוך מהחשבון ותעדכן את היתרה בהתאם.
- **הפקדה** – הפעולה תקבל כפרמטר את סכום הכסף שרוצים להפקיד לחשבון ותעדכן את היתרה בהתאם.
- **העברה** – פעולה המעבירה כסף לחשבון אחר. הפעולה תקבל כפרמטר עצם המייצג את החשבון שרוצים אליו להעביר כסף ואת סכום הכסף שיש להעביר.
- **החזר יתרה** – הפעולה תחזיר את היתרה בחשבון.

מימוש המחלקה

```
/*
 מחלקת חשבון בנק
*/
public class BankAccount
{
    private double balance;
    private string name;
    //פעולה בונה
    public BankAccount(string name, double balance)
    {
         this.name = name;
         this.balance = balance;
    }
    //הפקדה
    public void Deposit(double amount)
    {
         balance = balance + amount;
    }
    //משיכה
    public void Withdraw(double amount)
    {
         balance = balance - amount;
    }
    //העברה
    public void Transfer(BankAccount otherAccount, double amount)
    {
         Withdraw(amount);
         otherAccount.Deposit(amount);
    }
    //החזר יתרה
    public double GetBalance()
    {
         return balance;
    }
}// class BankAccount
```

קטע הקוד הבא מתאר העברה של 100 שקלים מחשבון ba1 לחשבון ba2

```
BankAccount ba1 = new BankAccount("ba1", 500);
BankAccount ba2 = new BankAccount("ba2", 200);
//נוכל להעביר 100 שקלים מחשבון 1 לחשבון 2
ba1.Transfer(ba2,100);
```

כלומר כעת בחשבון ba1 היתרה היא 400 שקלים ובחשבון ba2 היתרה היא 300 שקלים. כיצד? אמנם בשפת C# כאשר מעבירים פרמטר מטיפוס פשוט לפעולה ומשנים אותו, ערכו של המשתנה שהועבר כפרמטר אינו משתנה אך כאשר מעבירים עצם כפרמטר לפעולה, עוברת למעשה **ההפניה** אל העצם. באמצעות הפניה זו ניתן לבצע פעולות על העצם שהועבר כפרמטר ובכך לשנות אותו.

סוף פתרון בעיה 9

נסתכל על יישום הפעולה Transfer במחלקה "חשבון בנק". הפעולה מקבלת פרמטר מסוג חשבון בנק ומפעילה עליו את הפעולה Deposit עם סכום הכסף שהתקבל אף הוא כפרמטר. בניגוד לטיפוס פשוט (כפי שראינו במחלקת המחשבון), כאשר מעבירים עצם כפרמטר, עוברת ההפניה אל העצם (הפניה אל המקום בזיכרון ששם שמור העצם) ולא עותק של העצם. מסיבה זו, פעולות המופעלות על הפרמטר מופעלות על העצם עצמו ולא על עותק של העצם.

שאלה 11.22

במשחק "הדִיגֶר" קיים יצור בשם דיגר והוא אוסף ומאבד יהלומים.

א. כתבו מחלקה המגדירה דיגר המכיל את התכונה מספר יהלומים, את פעולת הגישה המחזירה את מספר היהלומים, ואת הפעולות "אסוף יהלום" ו"אבד יהלום". הפעולה "אסוף יהלום" מוסיפה 1 למניין היהלומים של הדיגר והפעולה "אבד יהלום" מפחיתה 1 ממספר היהלומים שיש לדיגר. כל דיגר מתחיל את חייו עם עשרה יהלומים.

ב. כעת יכול הדיגר לאכול דיגרים אחרים. כאשר דיגר אוכל דיגר אחר הוא מקבל (אוסף) את כל היהלומים שהיו לדיגר שאכל. הדיגר שנאכל מאבד את כל היהלומים שלו. הגדירו את פעולת "אכול דיגר".
הדרכה: פעולה זו תקבל דיגר כפרמטר.

שאלה 11.23

נתונה המחלקות Doubler ו-DoublerTest הבאות:

```
using System;
public class Doubler
{
    private double num;
    public Doubler(double num)
    {
        this.num = num;
    }
    public double GetNum()
    {
        return num;
    }
    public void DoubleNum()
    {
        num = num * 2;
        Console.WriteLine("Doubled = {0}", num);
    }
    public void DoubleIt(Doubler it)
    {
        it.DoubleNum();
        Console.WriteLine("Doubled: it = {0}", it.GetNum());
    }
}// Doubler

public class DoublerTest
{
    public static void Main()
    {
        Doubler doubler1 = new Doubler(5);
        Doubler doubler2 = new Doubler(4);
```

```
        Console.WriteLine("doubler1 is {0}", doubler1.GetNum());
        doubler1.DoubleNum();
        Console.WriteLine("doubler1 after DoubleNum is {0}",
                                              doubler1.GetNum());
        Console.WriteLine("doubler2 is {0}", doubler2.GetNum());
        doubler1.DoubleIt(doubler2);
        Console.WriteLine("doubler2 after DoubleIt is {0}",
                                              doubler2.GetNum());
    } // Main
}//DoublerTest
```

מה הפלט שיוצג בסוף התוכנית? השלימו:

```
doubler1 after DoubleNum is  _________
doubler2 after DoubleIt is _________
```

בעיה 10

מטרת הבעיה ופתרונה: הצגה של החזרת מערך מפעולה

> מוכר בחנות מקבל כבונוס 10% מסכום שלוש המכירות הגבוהות ביותר שמכר באותו חודש. כתבו תוכנית הקולטת את ערכה הכספי של כל מכירה ומכירה שהמוכר מכר בחודש מסוים. התוכנית תציג את שלוש המכירות הגבוהות ביותר ואת סכום הבונוס.
>
> כתבו מחלקה המגדירה מוכר. התכונות של מוכר הם שמו ושלוש המכירות הגבוהות ביותר שמכר. הפעולות המוגדרות עבור מוכר הן: הוספת מכירה, פעולה לחישוב הבונוס ופעולה להחזרת שלוש המכירות הגבוהות ביותר.

הגדרת המחלקה *מוכר*

נזדקק למחלקה המגדירה מוכר.

הגדרת התכונות

על פי הגדרת הבעיה, לעצם מהמחלקה מוכר יהיו התכונות: שם ושלוש המכירות הגבוהות ביותר באותו החודש. בהתאם לכך נבחר את המשתנים הבאים לתיאור התכונות:

- **name** – משתנה מטיפוס מחרוזת, מייצג את שם המוכר.
- **bonusSales** – מערך מטיפוס ממשי, מכיל את שלוש המכירות הגבוהות ביותר.

הגדרת הפעולות

- **פעולה בונה** – הפעולה הבונה תקבל כפרמטר את שמו של המוכר ותאתחל אותו. בנוסף לכך, הפעולה תקצה את מערך המכירות ותאפס אותו.
- **AddSale** – הפעולה תקבל כפרמטר ערך ממשי המייצג מכירה. הפעולה תבדוק אם מכירה זו מועמדת להיות אחת משלוש המכירות הגבוהות בַּחודש, אם כן ערכו יישמר במערך bonusSales.
- **CalculateBonus** – הפעולה איננה מקבלת פרמטרים והיא מחזירה את סכום הבונוס שהמוכר זכאי לו לפי הנוסחה: 10% מסכום המכירות במערך bonusSales.
- **GetBonusSales** – הפעולה מחזירה את המערך של שלוש המכירות הגבוהות בחודש.

מימוש המחלקה

```
/*
המחלקה מוכר
*/
public class Seller
{
    private double[] bonusSales;
    private string name;
    //הפעולה הבונה
    public Seller(string name)
    {
         this.name = name;
         bonusSales = new double[3];
         for (int i = 0; i < bonusSales.Length; i++)
              bonusSales[i] = 0.0;
    }
    //פעולה להוספת מכירה
    public void AddSale(double sale)
    {
         //מציאת המכירה הקטנה ביותר
         double min = bonusSales[0];
         int minIndex = 0;
         for (int i = 1; i < bonusSales.Length; i++)
         {
              if (min > bonusSales[i])
              {
                   min = bonusSales[i];
                   minIndex = i;
              }
         }
         //אם המכירה הנוכחית גדולה ממנה מעדכנים את המערך
         if (sale > min)
              bonusSales[minIndex] = sale;
    }
    //פעולה המחזירה את הבונוס שהמוכר זכאי לו
    public double CalculateBonus()
    {
         double bonus = 0;
         for (int i = 0; i < bonusSales.Length; i++)
              bonus = bonus + bonusSales[i] * 0.1;
         return bonus;
    }
    //פעולה המחזירה את מערך שלוש המכירות הגבוהות של החודש
    public double[] GetBonusSales()
    {
         // העתקת מערך המכירות והחזרתו
         double[] sales = new double[bonusSales.Length];
         for (int i = 0; i < bonusSales.Length; i++)
              sales[i] = bonusSales[i];
         return sales;
    }
} // Seller
```

שימו ♥: הפעולה `GetBonusSales` מחזירה **עותק** של המערך `bonusSales` ולא את המערך עצמו. הסיבה לכך היא שמערך ב-C# ממומש באמצעות הפניה, ולכן אם נחזיר את ההפניה למערך `bonusSales` (`return bonusSales;`), תוכל הפעולה הראשית לשנות את תוכן המערך בצורה ישירה באמצעות הפניה זו. הדבר אינו רצוי משום שהמערך `bonusSales` הוגדר כתכונה פרטית של המוכר, ואנו רוצים לוודא שערכיו יתעדכנו בצורה מבוקרת, אך ורק לפי האלגוריתם שקבענו בפעולה `AddSale`. בצורה זו אנחנו מגנים על הנתונים השמורים בעצם מוכֵר מפני שינויים מבחוץ. כל שינוי שיתבצע על המערך המוחזר לא ישפיע על נתוני המכירות בעצם מוכֵר.

הגדרת הפעולה הראשית

הפעולה הראשית תיצור עצם מסוג מוכר ותקלוט את נתוני המכירות שלו בחודש מסוים. סיום הנתונים יסומן במכירה שערכה הוא 0. התוכנית תחשב ותציג את הבונוס ואת שלוש המכירות הגבוהות ביותר של המוכר:

```
/*
       מחלקה ראשית המשתמשת במחלקה Seller
*/
using System;
public class SellerBonus
{
    public static void Main()
    {
         // יצירת עצם מסוג מוכר
         Seller seller = new Seller("selli");

         // קלט ועדכון המכירות
         double sale;
         Console.Write("Enter a sale: ");
         sale = double.Parse(Console.ReadLine());
         while (sale > 0.0)
         {
              seller.AddSale(sale);
              Console.Write("Enter a sale, Enter 0 to end: ");
              sale = double.Parse(Console.ReadLine());
         }
         // חישוב בונוס והצגת פלט
         double bonus;
         bonus = seller.CalculateBonus();
         Console.WriteLine("The seller bonus is {0}", bonus);
         double[] sales;
         sales = seller.GetBonusSales();

         // output the sales
         Console.Write("The bonus Sales are: ");
         for (int i = 0; i < sales.Length; i++)
              Console.Write(" {0} ", sales[i]);
         Console.WriteLine();
    } // Main
} // class SellerBonus
```

סוף פתרון בעיה 10

שאלה 11.24

בתחרות קפיצה לרוחק כל משתתף קופץ 5 קפיצות. כתבו מחלקה המגדירה קופץ לרוחק. עצם מהמחלקה ישמור את שמו של הקופץ ואת תוצאות הקפיצות שלו.
פתחו וממשו תוכנית לאימון בקפיצה לרוחק. התוכנית תקלוט את שמו של הקופץ. לאחר מכן תבקש התוכנית מהקופץ להכניס בכל פעם את תוצאת קפיצתו. (שימו לב שתוצאה של קפיצה לרוחק היא מספר ממשי, למשל 3.15 מטר).
התוכנית תציג כפלט את שמו של הקופץ, את קפיצותיו ואת ממוצע הקפיצות.

שאלה 11.25

כתבו מחלקה המגדירה ספר טלפונים אלקטרוני (כמו בטלפון נייד). ספר טלפונים מכיל מערך של מחרוזות. כל מחרוזת מורכבת משם וממספר טלפון. הניחו שהמערך ממוין לפי השמות. הגדירו פעולה המקבלת מחרוזת ומחזירה מערך המכיל את כל המחרוזות שמתחילות במחרוזת זו.
לדוגמא: נניח שבמערך המחרוזות 3 מחרוזות:
"דונלד 4444"
"מיקי 1234"
"מיני 998877"
כאשר הפרמטר לפעולה יהיה "מיק" תחזיר הפעולה מערך המכיל את המחרוזות:
"מיקי 1234"
כאשר הפרמטר לפעולה יהיה "מ" תחזיר הפעולה מערך המכיל את המחרוזות:
"מיקי 1234"
"מיני 998877"

סיכום

בפרק זה למדנו כיצד להגדיר מחלקות חדשות ולהשתמש בעצמים ממחלקות אלו.

הגדרה של **מחלקה** היא למעשה הגדרה של טיפוס חדש. הגדרת **מחלקה** כוללת הגדרה של תכונות ושל פעולות עבור **עצמים** של המחלקה, כלומר עבור משתנים מטיפוס המחלקה.

- **תכונות של עצם** הן משתנים מטיפוסים כלשהם (כולל מערכים ואף עצמים כגון מחרוזות), המאפיינים את העצם. בדרך כלל נגדיר את התכונות כ**פרטיות (private)** על מנת להגן עליהן מפני שינוי לא מבוקר מבחוץ וכדי לא לחשוף את אופן הייצוג הפנימי של העצם.
- **פעולות של עצם** הן פעולות המשויכות לעצם, והן פועלות בדרך כלל על תכונות העצם. את הפעולות נגדיר בדרך כלל כ**ציבוריות (public)**. **פעולות של עצם** יכולות לקבל פרמטרים ולהחזיר ערך.
- **פרמטרים** לפעולה יכולים להיות משתנים מכל טיפוס שהוא. ראינו שכאשר משתנים מטיפוסים פשוטים כגון **int** מועברים כפרמטרים, עובר רק הערך של המשתנה כפרמטר. כלומר שינוי של משתנה כזה בתוך הפעולה **איננו משנה** את ערכו של המשתנה מחוץ לפעולה. לעומת זאת, כאשר מערכים ועצמים מועברים כפרמטרים, עוברת ההפניה אל שטח הזיכרון שהעצם נמצא בו. באמצעות הפניה זו ניתן לשנות את תוכן המערך או העצם מתוך הפעולה. מכיוון שהשינוי מתבצע על המערך או על העצם עצמו ולא על עותק שלו, שינוי כזה תקף גם אחרי סיום הפעולה.

- **החזרת ערך מפעולה** יכולה להיות מטיפוס כלשהו. אופן החזרת הערך דומה לאופן העברת הפרמטרים, כלומר במקרה של ערך מטיפוס פשוט, מוחזר העותק שלו ובמקרה של ערך מטיפוס מערך או עצם, מוחזרת ההפניה אליו. לכן, אם קיימת תכונה שהיא מערך, נשקול להחזיר עותק של המערך ולא הפניה למערך עצמו, כדי שלא תהיה גישה ישירה למערך מבחוץ.
- **פעולה בונה** היא פעולה מיוחדת המופעלת מיד אחרי שמוקצה עבור העצם שטח בזיכרון, באמצעות ההוראה **new**. פעולה בונה מחזירה הפניה לעצם שנוצר, אך בכותרתה אין לכלול טיפוס ערך מוחזר. שמה של הפעולה הבונה חייב להיות זהה לשם המחלקה. כמו כל פעולה, גם פעולה בונה יכולה לקבל פרמטרים. פרמטרים אלה משמשים לאתחל את תכונות העצם.
- **פעולות גישה** הן פעולות המשמשות צינור גישה בטוח אל תכונות העצם מחוץ למחלקה. יש שני סוגים של פעולות גישה: פעולות המחזירות ערך של תכונה (Get) ופעולות המעדכנות ערך של תכונה (Set).
- פעולות של עצם יכולות להתייחס ישירות לתכונות של העצם וכן לקרוא לפעולות אחרות של העצם.
- כדי להתייחס לתכונה ששמה זהה לשם של פרמטר, יש להשתמש בקידומת **this** לפני שם התכונה.
- **יצירת עצם** כוללת את השלבים הבאים: הצהרה על משתנה מטיפוס המחלקה, הקצאת זיכרון ואתחול באמצעות הפעולה הבונה.
- **שימוש בעצם** כולל הפעלת הפעולות הציבוריות שלו.
- ניתן לשמור עצמים במערך, או במילים אחרות מערך שכל תא בו מפנה לעצם.
- מחלקה המכילה רק פעולות ללא תכונות נקראת "מחלקת שירות". היא מכילה אוסף של שירותים (פעולות) שאינם פועלים על עצם כלשהו. פעולות אלה יכולות לקבל פרמטרים ולהחזיר ערך אך הן לא פועלות על תכונות של עצם. פעולות אלה נקראות פעולות מחלקה **(class methods)** או פעולות סטטיות **(static methods)**.

סיכום מרכיבי שפת #C שנלמדו בפרק 11

הגדרת מחלקה בשפת #C

הגדרת מחלקה נכתבת באופן הבא:

```
public class שם המחלקה
{
	// הגדרת התכונות
	.
	.
	// הגדרת הפעולות
	.
	.
}
```

- המילה השמורה `public` מציינת שהמחלקה פתוחה לשימוש ציבורי. כלומר שמחלקות אחרות יכולות ליצור עצמים מטיפוס המחלקה.

- כדי להגדיר מחלקה נצהיר עליה באמצעות המילה השמורה **class**. לאחר ציון המילה **class** נציין את שם המחלקה. מקובל להתחיל שם מחלקה באות גדולה. אם שם המחלקה מורכב מכמה מילים, הן נכתבות צמודות ללא רווח, באותיות קטנות, וכל מילה מתחילה באות גדולה.

הגדרת תכונות בשפת C#

הגדרת תכונות נכתבת בדומה להצהרה על משתנים:

```
הרשאת-הגישה טיפוס-התכונה שם-התכונה
private double length;
```

- הרשאת הגישה **private** מציינת שהתכונה היא פרטית, כלומר היא מוכרת רק בתוך תחום המחלקה, ורק מתוכו ניתן להתייחס אליה.

הגדרת פעולות בשפת C#

הגדרת פעולה נכתבת באופן הבא:

```
הרשאת-הגישה טיפוס-הערך-המוחזר שם-הפעולה (רשימת פרמטרים)
{
      גוף הפעולה
}
```

לדוגמה:

```
public double GetLength()
{
      return length;
}
```

- הרשאת הגישה **public** מציינת שהפעולה היא ציבורית, כלומר היא מוכרת מחוץ לתחום המחלקה.
- הטיפוס-המוחזר מציין את טיפוס הערך שתחזיר הפעולה.
- במקרה של פעולה שאינה מחזירה ערך יש לרשום **void** במקום הערך החוזר.
- רשימת הפרמטרים מופיעה בסוגריים והיא כוללת זוגות המורכבים מטיפוסים ומשמות משתנים המופרדים בפסיקים.
- המילה השמורה **retrun** מציינת את הוראת החזרה מהפעולה. במקרה שהפעולה מחזירה ערך, יש לכלול ב-**return** ערך או ביטוי מהטיפוס המוחזר.
- פעולה סטטית היא פעולה השייכת למחלקה ולא לעצם כלשהו. על מנת להגדיר פעולה סטטית נוסיף את המילה **static** להגדרת הפעולה, למשל:

```
public static int DigitSum (int num)
```

הפעולה הבונה

הפעולה הבונה מוגדרת באופן הבא:

```
הרשאת-הגישה שם-המחלקה (רשימת פרמטרים)
{
}
```

- הפעולה הבונה משמשת לאתחול תכונות העצם.
- שפת C# מאתחלת באופן אוטומטי תכונות שלא אותחלו במפורש.

- ♦ כאשר לא מגדירים פעולה בונה כלשהי, שפת C# מספקת **כברירת מחדל** פעולה בונה ריקה חסרת פרמטרים.

שימו ♥: למען קריאות המחלקה חשוב מאוד לצרף להגדרתה הערות המתארות את המחלקה, את תכונותיה ואת פעולותיה.

שימוש בעצמים

מחלקה מגדירה טיפוס, וכדי להשתמש בו ניצור עצם מטיפוס המחלקה. יצירת עצם (למשל מתוך הפעולה הראשית) נעשית באופן הבא:

```
Time tm = new Time();
```

- ♦ יצירת העצם כוללת הצהרה על העצם, הקצאת מקום בזיכרון באמצעות ההוראה **new** ואתחול תכונות העצם באמצעות הפעולה הבונה.
- ♦ הפעלת פעולות העצם נעשית בסימון הנקודה באופן הבא:

```
tm.SetTime(8,30);
```

- ♦ יש לשים לב להתאמה בין אופן השימוש בפעולה של העצם לבין כותרת הפעולה: אם הפעולה מקבלת פרמטרים יש להקפיד לספק פרמטרים לפי הסדר שבו הם מופיעים בכותרת הפעולה. יש להקפיד על התאמה של טיפוס הפרמטרים המועברים לטיפוסים המפורטים בכותרת הפעולה; אם הפעולה מחזירה ערך יש להקפיד על התאמה של טיפוס הערך המוחזר לטיפוס הביטוי שמשולב בו הערך.
- ♦ זימון פעולה סטטית נעשה ישירות דרך שם המחלקה:

```
שם המחלקה.שם הפעולה(פרמטרים)
```

פרק 12 – תבניות אלגוריתמיות

12.1 מערך דו-ממדי

מערך דו-מְמַדִּי מייצג מבנה מתמטי הנקרא מטריצה, ובוודאי מוכר לכם מתשבצים, מלוח שחמט ומסודוקו. המבנה מורכב משורות ומעמודות. הנה דוגמה למערך דו-ממדי ובו ארבע שורות וחמש עמודות:

סוס	מתנה	כוכב	רכבת	נחש
חתול	כלב	ציפור	מכונית	ילד
ילדה	בית	עציץ	מטוס	שמש
עלה	פרח	גבעול	שמש	ירח

המערכים שהכרנו עד כה היו מערכים חד-ממדיים.
מערך חד-ממדי הוא בעצם מקרה פרטי של מערך דו-ממדי ובו שורה אחת בלבד. במערך דו-ממדי ניתן לפנות לכל איבר בציון השורה והעמודה שלו.

כיצד סורקים מערך דו-ממדי?

כזכור מפרק 10, לצורך פנייה לאיבר במערך חד-ממדי ציינו את מספר התא שאליו היינו מעוניינים לפנות. כדי לעבור על כל איבריו של מערך חד-ממדי נוח להשתמש בלולאה שמשתנה הבקרה שלה משמש כמציין של תא תורן במערך. כדי לפנות לאיבר במערך **דו-ממדי** יש לציין הן את מספר **השורה** והן את מספר **העמודה** של האיבר. למשל על מנת לציין את התא שמכיל את המילה "כלב" נפנה למערך בשורה השנייה בטור הרביעי (משמאל). כיצד נציין את התא שמכיל את המילה "ילדה"?

כדי לעבור על כל איבריו של מערך דו-ממדי עלינו להשתמש בלולאות מקוננות (לולאה בתוך לולאה): הלולאה החיצונית תעבור על פני השורות, והלולאה הפנימית תסרוק עבור כל שורה את כל איבריה. בסריקה כזאת נשתמש **בשני** משתני הבקרה, אחד של הלולאה החיצונית והאחר של הלולאה הפנימית, לצורך ציון התא התורן.

התבוננו באלגוריתם הבא שבו אנו קולטים ערכים במערך דו-ממדי ובו N שורות ו-M עמודות:

1. *עבור כל* i *שלם בין 1 ל-*N *בצע*
   1.1 *עבור כל* j *שלם בין 1 ל-*M *בצע*
      1.1.1 *קלוט ערך והשם את הערך במערך בשורה ה-*i *ובעמודה ה-*j

כיצד מצהירים על מערך דו-ממדי?

הצהרה על מערך דו-ממדי והקצאת זיכרון עבורו נעשים בדומה להצהרה ולהקצאת זיכרון עבור מערך חד-ממדי. כדי להצהיר על מערך דו-ממדי בשם `numbers` המכיל מספרים שלמים נכתוב:

```
int[,] numbers;
```

הסוגריים המרובעים והפסיק ביניהם מציינים כי `numbers` הוא מערך דו-ממדי ו-**`int`** מציין שאיבריו הם מטיפוס שלם.

כדי להקצות זיכרון עבור מערך דו-ממדי יש להשתמש בהוראה **new**. הקצאת זיכרון עבור 3 שורות ו-5 עמודות, תתבצע באופן הבא:

```
numbers = new int[3,5];
```

כעת המשתנה numbers **מפנה** לשטח הזיכרון שהוקצה עבור המערך. כתמיד, ניתן לאחד את ההצהרה ואת ההקצאה להוראה אחת, כך:

```
int[,] numbers = new int[3,5];
```

כפי שציינו בפרק הדן במערכים חד-ממדיים, מומלץ להשתמש בקבועים לצורך הגדרת גודל המערך, למשל נוכל להצהיר על המערך numbers באופן הבא:

```
const int NUM_OF_ROWS = 3;
const int NUM_OF_COLS = 5;
int[,]numbers = new int[NUM_OF_ROWS,NUM_OF_COLS];
```

בדומה לפנייה אל איבר במערך חד-ממדי, גם במערך דו-ממדי הפנייה נעשית באמצעות סוגריים מרובעים, אך במקרה של מערך דו-ממדי, הפנייה צריכה להתייחס גם למספר השורה של התא המבוקש וגם למספר העמודה. גם כאן, בדומה למערך חד-ממדי, מספור השורות והעמודות מתחיל מאפס. למשל, numbers[2,3] מתייחס לתא הנמצא **בשורה** השלישית (שמספרה הוא 2) **ובעמודה** הרביעית (שמספרה הוא 3).

הפעולה GetLength(i) מאפשרת לנו לברר את ממדי המערך הדו-ממדי באופן הבא: הערך של numbers.GetLength(0) מציין את מספר השורות במערך numbers, במקרה זה 3, ואילו הערך של numbers.GetLength(1) מציין את מספר האיברים בכל שורה, במקרה זה 5.

התבוננו בקטע התוכנית הבא המציג את איברי המערך הדו-ממדי numbers בצורת טבלה:

```
for(int i = 0; i < numbers.GetLength(0); i++)
{
     for(int j = 0; j < numbers.GetLength(1); j++)
          Console.Write("{0} ", numbers[i,j]);
     Console.WriteLine();
}
```

נסכם את המושגים הבסיסיים הנוגעים למערכים דו-ממדיים ולעבודה עמם בשפת C#:

- **מערך דו-ממדי בשפת C#** הוא מבנה נתונים, המורכב משורות ומעמודות.
- **הקצאת זיכרון** עבור מערך דו-ממדי מתבצעת באמצעות ההוראה **new**.
- **ציון** איברי מערך דו-ממדי (בדומה למערך חד-ממדי) מתחיל משורה 0 ומעמודה 0.
- **גישה** לאיבר במערך דו-ממדי נעשית בציון השורה והעמודה של האיבר הרצוי, למשל: matrix[מספר שורה, מספר עמודה]
- **סריקת** מערך דו-ממדי מתבצעת באמצעות לולאות מקוננות.
- **הפעולה** GetLength(0) של מערך דו-ממדי מחזירה את מספר השורות במערך דו-ממדי והפעולה GetLength(1) מחזירה את מספר האיברים בכל שורה, כלומר את מספר העמודות.

בעיה 1

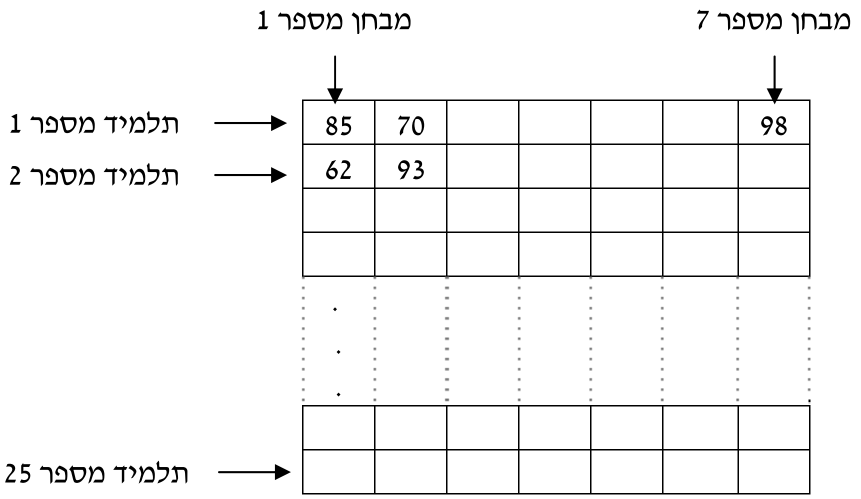
מטרת הבעיה ופתרונה: הצגת מערך דו-ממדי.

> בכיתת מדעי המחשב 25 תלמידים. במהלך השנה התקיימו בדיוק 7 מבחנים. פתחו וישמו אלגוריתם שיקבל כקלט את ציוני שבעת המבחנים של כל תלמיד. בנוסף, האלגוריתם יקלוט מספר המציין מבחן (מספר בין 1 ל-7), ומספר המציין תלמיד (מספר בין 1 ל-25), ויציג כפלט:
>
> 1. את ממוצע ציוני שבעת המבחנים של התלמיד שמספרו התקבל כקלט.
> 2. את ממוצע הציונים של כל 25 התלמידים במבחן שמספרו התקבל כקלט.
> 3. את הציון הגבוה ביותר שהיה במבחן כלשהו במהלך השנה במדעי המחשב.

כיצד נשמור את המידע הדרוש?

ישנם 25 תלמידים ולכל תלמיד 7 ציונים, ניתן לחשוב על כך בתור טבלה בת 25 שורות – שורה עבור כל תלמיד, ו-7 עמודות – עמודה עבור כל ציון, כך שבכל שורה יהיו שבעת הציונים של תלמיד מסוים.

התבוננו באיור הבא:



סידור זה של הציונים במערך דו-ממדי יאפשר לסרוק את הנתונים של תלמיד מסוים ושל מבחן מסוים. סריקת ציוני תלמיד תתבצע באמצעות סריקת השורה המתאימה לו במערך, ואילו סריקת ציוני מבחן מסוים תתבצע באמצעות סריקת העמודה המתאימה לו במערך.

הגדרת המחלקה *ציוני התלמידים*

נזדקק למחלקה המייצגת את ציוני התלמידים, ובה מערך דו-ממדי.

הגדרת התכונות

- **grades** – מערך דו-ממדי של שלמים המכיל את ציוני התלמידים בכיתה.

הגדרת הפעולות

- **פעולה בונה** – הפעולה תקבל כפרמטר את מספר התלמידים בכיתה, ואת מספר המבחנים ותקצה זיכרון עבור מערך הציונים.
- **עדכון ציון** – הפעולה תקבל את מספר התלמיד, את מספר המבחן ואת הציון במבחן ותעדכן את הציון במערך הציונים.
- **ממוצע תלמיד** – הפעולה תקבל מספר תלמיד (`studentNumber`) ותחזיר את ממוצע המבחנים של התלמיד הנתון.

בפעולה זו אנו נדרשים לעבור רק על שורה אחת מסוימת בטבלה. מעבר על שורה אחת דומה למעבר על מערך חד-ממדי (שבו כזכור יש רק שורה אחת). לכן אנו זקוקים כאן ללולאה אחת בלבד ולא לקינון לולאות. במהלך ביצוע הלולאה מספר השורה יישאר קבוע ואילו מספר העמודה ישתנה, בהתאם למשתנה הבקרה. כמו כן נזכור כי מספור העמודות והשורות ב-C# מתחיל ב-0.

1. אפס את משתנה סכום הציונים של התלמיד
2. עבור כל i שלם בין 0 למספר המבחנים פחות 1 בצע:
   2.1 הוסף לסכום הציונים את הערך הנמצא במערך הציונים בשורה studentNumber-1 ובעמודה i
3. חשב את ממוצע התלמיד כסכום הציונים חלקי מספר המבחנים

- **ממוצע מבחן** – הפעולה תקבל מספר מבחן (`testNumber`) ותחזיר את ממוצע כל התלמידים במבחן הנתון.

בדומה לפעולה הקודמת, כדי לעבור רק על עמודה אחת בטבלה (העמודה `testNumber-1`) נזדקק שוב רק ללולאה אחת, שמשתנה הבקרה שלה נע מ-0 עד למספר התלמידים פחות אחת. במהלך ביצוע הלולאה מספר השורה ישתנה בהתאם למשתנה הבקרה, ומספר העמודה יישאר קבוע:

1. אפס את משתנה סכום הציונים של המבחן
2. עבור כל i שלם בין 0 למספר התלמידים פחות 1 בצע:
   2.1 הוסף לסכוםהציונים את הערך הנמצא במערך הציונים בשורה i ובעמודה testNumber-1
3. חשב את ממוצע המבחן כסכום הציונים חלקי מספר התלמידים

- **ציון מקסימום** – הפעולה תחזיר את הציון הגבוה ביותר השמור במערך הציונים.

פעולה זו דורשת מעבר על **כל** הטבלה (בלולאה מקוננת), תוך מציאת המקסימום בערכי הטבלה (ראו ביישום האלגוריתם בהמשך).

מימוש המחלקה

```
/*
   מחלקת ציוני התלמידים
*/
public class StudentGrades
{
      // הצהרת מערך הציונים
      private int[,] grades;
      // פעולה בונה
```

```
    public StudentGrades (int students, int tests)
    {
        grades = new int[students,tests];
    }
    // עדכון ציון מבחן של תלמיד נתון
    public void SetGrade(int studentNumber, int testNumber,
                                                     int grade)
    {
        grades[studentNumber-1,testNumber-1] = grade;
    }
    // מציאת ממוצע תלמיד נתון
    public double StudentAverage(int studentNumber)
    {
        int sumStudent = 0;
        for (int i = 0; i < grades.GetLength(1); i++)
            sumStudent = sumStudent + grades[studentNumber-1,i];
        return (double)sumStudent / grades.GetLength(1);
    }
    // מציאת ממוצע מבחן נתון
    public double TestAverage(int testNumber)
    {
        int sumTest = 0;
        for (int i = 0; i < grades.GetLength(0); i++)
           sumTest = sumTest + grades[i,testNumber-1];
        return (double)sumTest / grades.GetLength(0);
    }
    // מציאת ציון מקסימלי
    public int MaxGrade()
    {
        int max=0;
        for(int i = 0; i < grades.GetLength(0); i++)
            for(int j = 0; j < grades.GetLength(1); j++)
                if (grades[i,j] > max)
                    max = grades[i,j];
        return max;
    }
} // class StudentGrades
```

שימו ♥ : הפעולה הבונה מקבלת כפרמטר את מספר התלמידים ואת מספר הציונים ומקצה מקום למערך דו-ממדי שיכיל את הציונים. ציוני התלמידים ייקלטו בפעולה הראשית ויתעדכנו באמצעות פעולת הגישה SetGrade. מיד נציג את מימוש הפעולה הראשית.

הגדרת הפעולה הראשית

פירוק הבעיה לתת-משימות:

משימות הפעולה הראשית הן:

1. יצירת עצם מסוג `StudentGrades`, קליטת נתוני הציונים ועדכונם.
2. קליטת מספר תלמיד `studentNumber` ומספר מבחן `testNumber`.
3. חישוב ממוצע **התלמיד** שמספרו `studentNumber`.
4. חישוב ממוצע **המבחן** שמספרו `testNumber`.
5. מציאת הציון המקסימלי.

תת-המשימה הראשונה כוללת יצירת עצם מסוג StudentGrades. לצורך כך נגדיר קבועים המייצגים את מספר התלמידים ואת מספר המבחנים ונעביר אותם לפעולה הבונה. לאחר מכן נקלוט את הציונים ונעדכן אותם בעצם ציוני התלמידים.

תת-המשימה השנייה היא קליטת שני ערכים לשני משתנים מטיפוס שלם. בתת-משימה השלישית עד החמישית נבצע את החישובים באמצעות פעולות המוגדרות בעצם StudentGrades.

מימוש הפעולה הראשית

```
/*
  המחלקה הראשית המשתמשת במחלקת ציוני התלמידים
*/
using System;
public class StudentTest
{
    public static void Main()
    {
      // הגדרת קבועים – ממדי המערך
      const int NUM_OF_STUDENTS = 25;
      const int NUM_OF_TESTS = 7;
      //    הגדרת משתנים
      double sAverage;
      double tAverage;
      int grade;
      int studentNum;
      int testNum;
      int max;
      // יצירת עצם מסוג מחלקת הציונים
      StudentGrades studentGrades =
                    new StudentGrades(NUM_OF_STUDENTS, NUM_OF_TESTS);
      // קליטת הציונים
      Console.WriteLine("Enter student grades: " );
      for(int i = 1; i <= NUM_OF_STUDENTS; i++)
      {
        for(int j = 1; j <= NUM_OF_TESTS; j++)
        {
           Console.Write(" Student number {0} Test number {1}: ",
                                                               i,j);
           grade = int.Parse(Console.ReadLine());
           studentGrades.SetGrade(i,j,grade);
        } // for j
      } // for i
      //קליטת מספר תלמיד ומספר מבחן
      Console.Write("Enter student number: ");
      studentNum = int.Parse(Console.ReadLine());
      Console.Write("Enter test number: ");
      testNum = int.Parse(Console.ReadLine());
      // חישוב הערכים המבוקשים באמצעות זימון הפעולות המתאימות
      sAverage = studentGrades.StudentAverage(studentNum);
      tAverage = studentGrades.TestAverage(testNum);
      max = studentGrades.MaxGrade();
```

```
      Console.WriteLine("The average of student {0} is {1}",
                                             studentNum, sAverage);
      Console.WriteLine("The average of test {0} is {1}",
                                                testNum, tAverage);
      Console.WriteLine("The highest grade is: {0}", max);
    }// Main
 }// class StudentTest
```

שימו ♥: מאחר שמספור איברים במערך מתחיל מ-0, ואילו מספור התלמידים והמבחנים מתחיל מ-1, הרי שהתלמיד הרביעי הוא התלמיד במקום 3 במערך. לכן בפעולה SetGrade אנו מפחיתים 1 ממספר התלמיד וממספר המבחן:

```
grades[studentNumber-1,testNumber-1] = grade;
```

סוף פתרון בעיה 1

שאלה 12.1

בנו מחלקה ובה מערך דו-ממדי המכיל מספרים שלמים. המחלקה תכיל את הפעולות הבאות:

א. פעולה בונה המקבלת את ממדי המערך הדו-ממדי ומקצה עבורו מקום.
ב. פעולת גישה לאתחול תא במערך הדו-ממדי. הפעולה מקבלת מספר שורה, מספר עמודה ומספר שלם ומעדכנת את התא המתאים.
ג. פעולה המקבלת מספר שורה ומחזירה את סכום איברי השורה.
ד. פעולה המציגה את ערכי האיברים הנמצאים בשורות **הזוגיות**.
ה. פעולה המקבלת מספר שורה ומחזירה "אמת" אם כל איברי השורה מתחלקים ב-3.

שאלה 12.2

א. פתחו אלגוריתם המאחסן בכל תא של מערך דו-ממדי מספר אשר ספרת העשרות שלו שווה למספר השורה של התא, וספרת האחדות שלו שווה למספר העמודה של התא. למשל, מערך דו-ממדי בגודל 3x4 יראה כך:

3	2	1	0
13	12	11	10
23	22	21	20

ב. הגדירו מחלקה הכוללת מערך דו-ממדי כתכונה ואת הפעולות הבאות:
- פעולה בונה המקבלת את ממדי המערך, מקצה עבורו מקום ומיישמת את האלגוריתם שפיתחתם בסעיף הקודם.
- פעולה להצגת תוכן המערך הדו-ממדי בצורת טבלה.

ג. הגדירו פעולה ראשית הקולטת מהמשתמש את ממדי המערך הרצוי, מייצרת עצם מהסוג שהוגדר בסעיף הקודם ומציגה את המערך שנוצר.

שאלה 12.3

בנו מחלקה ובה מערך דו-ממדי המכיל מספרים ממשיים. המחלקה תכיל את הפעולות הבאות:

א. פעולה בונה המקבלת את ממדי המערך הדו-ממדי ומקצה עבורו מקום.
ב. פעולת גישה לאתחול תא במערך הדו-ממדי. הפעולה מקבלת מספר שורה, מספר עמודה ומספר ממשי ומאתחלת את התא המתאים.
ג. פעולה המקבלת מספר שורה ומספר עמודה של איבר במערך ומחזירה את סכום הערכים שמסביב לאיבר (מעליו, מתחתיו, מימינו ומשמאלו).

ד. פעולה המקבלת מספר שורה ומספר עמודה של איבר במערך ומחזירה את סכום הערכים שמסביב לאיבר כולל הערכים הממוקמים אלכסונית לו.

שימו ♥: בשתי הפעולות האחרונות יש להיזהר מחריגה מגבולות המערך! כלומר להתייחס רק לשכנים שבגבולות המערך.

שאלה 12.4

פתחו וישמו אלגוריתם לניהול הזמנת מקומות במטוס. הקלט הוא סדרה של בקשות להזמנת מושבים במטוס. כל בקשה מורכבת ממספר שורה ומאות המייצגת את המושב ('a', 'b', 'c', ...). לאחר כל בקשה תוצג הודעה: "בקשתך התקבלה" או "המקום שביקשת תפוס". לאחר סיום העיבוד של כל בקשה יוצג מצב התפוסה של כל המושבים במטוס, כאשר A מסמן מושב פנוי ו-N מסמן מושב תפוס. הקלט יסתיים כאשר יוקלד המושב A 1 (והוא שייך כידוע לדיילת הראשית).

הדרכה: הגדירו מחלקה המייצגת מטוס. המחלקה תכלול מערך דו-ממדי בוליאני המייצג את המושבים ואת הפעולות הבאות:

א. פעולה בונה המקבלת את ממדי המטוס (מספר שורות ומספר מושבים בכל שורה). הפעולה תקצה את מערך המושבים ותאתחל את כל המושבים כך שיהיו פנויים.

ב. פעולת הזמנה המקבלת מספר שורה ואות המייצגת את המושב. הפעולה מסמנת את המקום כתפוס ומחזירה ערך בוליאני המציין את הצלחת הפעולה או את כישלונה.

זכרו: כדי להמיר את התווים למקומות במערך ('a' הוא 0, 'b' הוא 1 וכו') יש לחסר מהתו המבוקש את התו 'a'. כלומר אם התו מאוחסן במשתנה ch, נכתוב: 'ch-'a. חשבו מדוע? (ראו פרק 4).

ג. פעולה להצגת מצב מושבי המטוס.

הגדירו פעולה ראשית הכוללת את קליטת ממדי המטוס, את יצירת העצם מטוס, את קליטת הזמנות המשתמשים ואת ביצוע ההזמנות.

שאלה 12.5

חגית משחקת שש-בש עם לירון בכל יום בשבוע, בכל יום 10 משחקים. תוצאות המשחקים נשמרות במערך דו-ממדי בוליאני בגודל 7×10: כל שורה במערך מייצגת יום בשבוע, השורה תכיל את תוצאות עשרת המשחקים ששיחקו באותו היום. למשל אם חגית ניצחה במשחק השני ביום חמישי אז התא השני בשורה החמישית יכיל את הערך `true`, אם לירון ניצח באותו משחק, תא זה יכיל את הערך `false`. פתחו אלגוריתם שיחשב ויציג:

א. למי מהשניים כמות הניצחונות הגדולה ביותר.

ב. הודעה המציינת את יום המזל של חגית (**היום** שבו לחגית היו הכי הרבה ניצחונות). אם יש יותר מיום אחד כזה ההודעה תציין את היום הראשון שנמצא.

ג. הודעה המציינת את משחק המזל של לירון (מכיוון שביום מתקיימים עשרה משחקים נמספר אותם מ-1 עד 10. **משחק המזל** הוא זה שמספר הניצחונות בו במהלך השבוע הוא הגדול ביותר). אם יש יותר ממשחק אחד כזה ההודעה תציין את המשחק הראשון שנמצא.

הדרכה: הגדירו מחלקה המייצגת את תוצאות המשחקים. המחלקה תכלול מערך דו-ממדי בוליאני, ואת הפעולות הבאות:

- פעולה בונה ליצירת מערך התוצאות.
- פעולת גישה לעדכון תוצאה של משחק מסוים.
- פעולות הבודקות: למי יש יותר ניצחונות, מהו יום המזל של חגית, ומהו משחק המזל של לירון. הפעולות מחזירות ערכים בהתאם.

מערך דו-ממדי ריבועי

מערך דו-ממדי ריבועי הוא מערך שבו מספר השורות שווה למספר העמודות. כלומר בהינתן מערך דו-ממדי ששמו matrix, הערך של `matrix.GetLength(0)` שווה לערך של `matrix.GetLength(1)`.

במערך דו-ממדי ריבועי (או **מטריצה ריבועית**) אפשר להתייחס למושגים אלכסון **ראשי** ואלכסון **משני**:

אלכסון ראשי:

X		
	X	
		X

אלכסון משני:

		X
	X	
X		

חשבו: מה מאפיין את כל התאים באלכסון הראשי? ובאלכסון המשני? לצורך פתרון הבעיה הבאה נזדקק לאפיון זה.

בעיה 2

מטרת הבעיה ופתרונה: הצגת שימוש באלכסונים במטריצה ריבועית.

פתחו אלגוריתם שיקלוט ערכים ממשיים למטריצה ריבועית בגודל n×n. האלגוריתם יחשב ויציג את סכום הערכים באלכסון הראשי ואת סכום הערכים באלכסון המשני. ישמו את האלגוריתם בשפת C#.

הגדרת המחלקה *Diagonal*

הגדרת התכונות

- **matrix** – מערך דו-ממדי ריבועי, המכיל איברים מטיפוס שלם.

הגדרת הפעולות

בנוסף לפעולה הבונה ולפעולת גישה המאתחלת איבר במטריצה (בדומה לפעולות שהוצגו בבעיה מספר 1), נזדקק לשתי פעולות המחשבות את סכומי האלכסונים המבוקשים.

- **SumMain** – הפעולה תחזיר את סכום איברי האלכסון הראשי. לשם סכימה זו עלינו לבדוק **מה מאפיין איבר** באלכסון הראשי. איבר באלכסון הראשי הוא איבר שמספר השורה שלו ומספר העמודה שלו שווים. שימו לב לאיור הבא שבו מופיעים מצייני התאים:

0,0	0,1	0,2	0,3
1,0	**1,1**	1,2	1,3
2,0	2,1	**2,2**	2,3
3,0	3,1	3,2	**3,3**

- **SumSecond** – הפעולה תחזיר את סכום איברי האלכסון המשני. אם כך, מה מאפיין את איברי האלכסון **המשני**?

התבוננו בסכום המציינים של איברים אלה באיור - הוא קבוע וערכו הוא: N-1. N מייצג את מספר השורות והעמודות במטריצה הריבועית. כדי שהמציינים של איברי האלכסון המשני ישלימו את הסכום הזה , עלינו לפנות לאיבר במקום ה-matrix[i,N-1-i], חבְּרו את האינדקסים, מה קיבלתם?

במטריצה ריבועית בגודל n×n, המציינים של כל תא באיברי **האלכסון הראשי** שווים.
במטריצה ריבועית בגודל n×n, סכום המציינים של כל תא באיברי **האלכסון המשני** שווה ל-n-1.

מימוש המחלקה:

```
/*
  מחלקת Diagonal
*/
public class Diagonal
{
    private double[,] matrix;
    // פעולה בונה
    public Diagonal(int n)
    {
        matrix = new double[n,n];
    }
    // פעולת גישה לעדכון איבר במטריצה
    public void SetVal(int row, int col, double val)
    {
        matrix[row,col] = val;
    }
    // חישוב סכום איברי האלכסון הראשי
    public double SumMain()
    {
        double sumMain = 0;
        for (int i = 0; i < matrix.GetLength(0); i++)
           sumMain = sumMain + matrix[i,i];
        return sumMain;
    }
    // חישוב סכום איברי האלכסון המשני
    public double SumSecond()
    {
        double sumSecond = 0;
        for (int i = 0; i < matrix.GetLength(0); i++)
           sumSecond = sumSecond + matrix[i,matrix.GetLength(0)-1-i];
          return sumSecond;
    }
}//class Diagonal
```

מימוש הפעולה הראשית

```
/*
   המחלקה הראשית המשתמשת במחלקה Diagonal
*/
using System;
public class DiagonalTest
{
    public static void Main()
    {
      // הגדרת קבועים ומשתנים
      const int N = 5;
      double val;
      // יצירת עצם מסוג מחלקת האלכסונים
      Diagonal diagonalMat = new Diagonal(N);
```

```
      // קליטת האיברים
      Console.WriteLine("Enter matrix values: " );
      for(int i = 0; i < N; i++)
      {
         for(int j = 0; j < N; j++)
         {
            Console.Write(" row {0} col {1}: ", i, j);
            val = double.Parse(Console.ReadLine());
            diagonalMat.SetVal(i,j,val);
         } // for j
      } // for i
      //הצגת סכום הערכים באלכסון הראשי ובאלכסון המשני
      Console.WriteLine("Main diagonal sum {0}",
                                        diagonalMat.SumMain());
      Console.WriteLine("Secondary diagonal sum {0}",
                                        diagonalMat.SumSecond());
   }// Main
}// class DiagonalTest
```

סוף פתרון בעיה 2

שאלה 12.6

לפניכם כמה קטעי תוכניות בשפת C#, עבור כל אחד מהקטעים כתבו מה יוצג על המסך ובאיזו צורה.

הניחו ש-mat מוגדרת כמטריצה ריבועית בגודל 5×5 והיא מכילה את הערכים הבאים:

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

א.

```
for(i = 0; i < mat.GetLength(0); i++)
{
      Console.Write(m[2,i]);
}
```

ב.

```
for(i = 0; i < mat.GetLength(0); i++)
{
      Console.Write(m[i,2]);
}
```

ג.

```
for(i = 0; i < mat.GetLength(0); i++)
{
    Console.Write(m[5,5]);
}
```

ד.

```
for(i = 0; i < mat.GetLength(0); i++)
{
     Console.Write(m[i,i]);
}
```

שאלה 12.7

פתחו וישמו אלגוריתם הבודק לגבי מטריצה ריבועית אם היא:

א. "שוות שורות" – מטריצה "שוות שורות" היא מטריצה שסכום כל שורה בה שווה.

ב. "שוות עמודות" – מטריצה "שוות עמודות" היא מטריצה שסכום כל עמודה בה שווה.

ג. "ריבוע קסם" – "ריבוע קסם" הוא מטריצה "שוות שורות" ו"שוות עמודות" ובנוסף סכום כל שורה שווה לסכום כל עמודה, וסכום כל אלכסון שווה לסכום שורה ולסכום עמודה. הנה דוגמה לריבוע קסם:

8	1	6
3	5	7
4	9	2

הדרכה: הגדירו מחלקה המכילה מערך דו-ממדי של שלמים ואת הפעולות הבאות: פעולה בונה ליצירת המטריצה הריבועית, פעולת גישה לאתחול תא במטריצה ושלוש פעולות בוליאניות הבודקות אם המטריצה היא שוות שורות, שוות עמודות וריבוע קסם.

שימו ♥ : לצורך בדיקת ריבוע קסם, ניתן להיעזר בפעולות הקודמות שמימשתם.
הגדירו פעולה ראשית הקולטת מספרים עבור מטריצה בגודל 3×3 ובודקת אם המטריצה היא שוות שורות, אם היא שוות עמודות ואם היא ריבוע קסם.

שאלה 12.8

לחגית נמאס מהשש-בש. היא המציאה משחק חדש הנקרא: "תפוס את האלכסון". מהלך המשחק: מניחים לוח דמקה על השולחן (כלומר מטריצה בגודל 8×8). בכל תור חגית זורקת מטבע על הלוח. אם המטבע נפל מעל לאלכסון הראשי היא מקבלת נקודה, אם הוא נפל מתחת לאלכסון הראשי היא מפסידה נקודה, ואם המטבע נפל בדיוק על האלכסון הראשי היא מקבלת חמש נקודות! עליכם לפתח אלגוריתם שיקבל כקלט סדרה של זוגות המסתיימת ב-1-,1-. כל זוג מתאר את המשבצת שעליה נפל המטבע באותו תור (מספר שורה ומספר עמודה). פלט האלגוריתם יהיה סכום הנקודות שצברה חגית באותו משחק. ישמו את האלגוריתם בשפת C#.

הדרכה: עליכם לכתוב מחלקה עבור המשחק, המחלקה תכלול תכונה המייצגת את גודל הלוח, פעולה בונה המקבלת את גודל הלוח ומאתחלת אותו, ופעולה המבצעת מהלך במשחק (הפעולה תקבל את מיקום המטבע ותחזיר 1-, 1 או 5 לפי כללי המשחק). הפעולה הראשית תיצור עצם מסוג לוח משחק בגודל 8×8, תקלוט זוגות המציינים את מיקום המטבע, תפעיל את פעולת מהלך המשחק, תסכם את הניקוד ולבסוף תציג את התוצאה.

בעיה 3

מטרת הבעיה ופתרונה: הצגת התבנית "מעבר על זוגות סמוכים בסדרה", ושימוש בפעולה המזמנת פעולה.

> מטריצה "סדרתית" היא מטריצה אשר **כל אחת משורותיה** היא סדרה חשבונית.
> סדרה חשבונית היא סדרה שבה ההפרש בין כל שני איברים סמוכים הוא קבוע.
> למשל, שתי הסדרות הבאות הן סדרות חשבוניות: 13, 11, 9, 7, 5, 3 ו-25, 20, 15, 10, 5.
> פתחו וישמו אלגוריתם אשר יקלוט מספרים שלמים במטריצה ויציג הודעה אם המטריצה סדרתית או לא.

הגדרת המחלקה *Serial*

הגדרת התכונות

- **matrix** – מערך דו-ממדי, המכיל איברים מטיפוס ממשי.

הגדרת הפעולות

בנוסף לפעולה הבונה ולפעולת גישה המאתחלת איבר במטריצה, נזדקק לפעולות הבאות:

♦ **IsMatrixSeries** – פעולה בוליאנית הבודקת אם המטריצה היא "סדרתית". פעולה זו משתמשת בתבנית ***האם כל הערכים בסדרה מקיימים תנאי*** נבצע זאת כך:

1. עבור כל שורה במטריצה נבצע:

1.1 אם השורה <u>אינה</u> סדרה חשבונית החזר "שקר"

2. החזר "אמת"

מכיוון שהאלגוריתם דורש שנבדוק את כל שורות המטריצה עד שניתקל בשורה שאינה סדרה חשבונית, נגדיר פעולה נוספת הבודקת אם שורה נתונה היא סדרה חשבונית. כעת נוכל להפעיל פעולה זו שוב ושוב (בכל פעם על השורה הבאה) כל עוד השורות הן סדרות חשבוניות. כיוון שפעולה זו תהיה שימושית עבור המחלקה Serial בלבד, ולא עבור המחלקה הראשית, נוכל להגדיר אותה *כפעולה פרטית*. *פעולה פרטית* היא פעולה שהרשאת הגישה אליה היא **private**, וניתן לגשת אליה (להפעיל אותה) רק מתוך המחלקה שבה היא מוגדרת (כמו תכונות פרטיות). התבוננו בפעולה הבאה:

♦ **IsRowArithmetic** – פעולה בוליאנית המקבלת מספר שורה ובודקת אם היא סדרה חשבונית או לא. פעולה זו שימושית רק לצורך ביצוע הפעולה **IsMatrixSeries** ולכן היא תוגדר כפרטית. האלגוריתם למימוש פעולה זו ישתמש בתבנית המוכרת לנו ***מעבר על זוגות סמוכים בסדרה***.

מימוש המחלקה:

```
/*
   המחלקה Serial
*/
public class Serial
{
    private double[,] matrix;
    // פעולה בונה
    public Serial(int n, int m)
    {
        matrix = new double[n,m];
    }
    // פעולת גישה לעדכון איבר במטריצה
    public void SetVal(int row, int col, double val)
    {
        matrix[row,col] = val;
    }
    // פעולה פרטית הבודקת אם השורה הנתונה היא סדרה חשבונית
    private bool IsRowArithmetic(int row)
    {
        double distance = matrix[row,1] - matrix[row,0];
        for(int i = 2; i < matrix.GetLength(1); i++)
            if (matrix[row,i] - matrix[row,i-1] != distance)
                return false;
        return true;
    }
```

```
    // "פעולה הבודקת האם המטריצה היא "סדרתית
    public bool IsMatrixSeries ()
    {
        for(int i = 0; i < matrix.GetLength(0); i++)
            if (!IsRowArithmetic(i))
                return false;
        return true;
    }
}// class Serial
```

שימו ♥:

הפעולה `IsRowArithmetic` היא פעולה פרטית, כלומר ניתן לזמן אותה רק מתוך המחלקה `Serial`. במקרים שמחלקה כלשהי מגדירה פעולות עזר והן משמשות רק פעולות אחרות מאותה המחלקה, נגדיר את הרשאת הגישה לפעולות העזר כ-**`private`**.

תוכלו לממש בעצמכם את המחלקה הראשית לפי הפירוק הבא:

1. יצירת עצם מסוג `Serial` וקליטת ערכים למטריצה.
2. בדיקה אם המטריצה "סדרתית" באמצעות פעולה מתאימה.
3. הדפסת הודעה מתאימה בהתאם לערך שהוחזר.

סוף פתרון בעיה 3

שאלה 12.9

עלינו לפתח מערכת ממוחשבת לרכישת כרטיסי קולנוע. בבית הקולנוע יש שני אולמות: אולם שביט ואולם ארמון. באולם שביט מוצג בימים אלה הסרט מלחמת הכוכבים ובאולם ארמון מוצג בימים אלה הסרט שרק. בכל אולם יש 10 שורות, ו-20 מושבים בכל שורה.

המערכת מציגה למשתמש את הסרטים שמוצגים בבית הקולנוע בימים אלה ומבקשת ממנו לציין את שם הסרט שהוא רוצה לראות. לאחר מכן המערכת שואלת את המשתמש כמה כרטיסים הוא רוצה לרכוש. המערכת מציגה לפני המשתמש את המושבים הפנויים ומבקשת ממנו לבחור מושבים.

בסוף ההזמנה המערכת מדפיסה: נרכשו <מספר כרטיסים> כרטיסים לסרט <שם הסרט> המוצג באולם <שם האולם שהסרט מציג בו>

לפניכם הגדרת המחלקה Cinema. השלימו את המקומות החסרים, והוסיפו מחלקה ראשית שתטפל בפעולות הקולנוע:

```
public class Cinema
{
    private string name;            // שם אולם הקולנוע
    private string movie;           // שם הסרט המוצג
    private bool[,] freeSeats;      // רשימת מושבי האולם ומצבם
    private const int ROWS = 10;
    private const int COLS = 20;
    // הפעולה הבונה
    public Cinema (string aName, string aMovie)
    {
        name = aName;
        movie = aMovie;
        freeSeats = new bool[ROWS,COLS];
        // מסמנים את כל המושבים כפנויים
        for (int i = 0; i < ROWS; i++)
             for (int j = 0; j < COLS; j++)
```

```
                ___________________________;
    }
  // פעולת גישה: מחזירה את שם הסרט המוצג באולם
    public string GetMovie ()
    {
        return __________________;
    }
    // פעולת גישה: מחזירה את שם אולם הקולנוע
    public string GetName()
    {
        return __________________;
    }
    // פעולת גישה: מעדכנת את שם הסרט המוצג באולם
    public void SetMovie(string aMovie)
    {
        ________________________;
    }
    // פעולה בוליאנית הבודקת האם סרט נתון מוצג באולם הקולנוע
    public bool IsShowing(string aMovie)
    {
        return (_______________________)
    }
    //פעולה המחזירה מחרוזת שמתארת את רשימת המושבים הפנויים, במבנה
    // [מספר שורה,מספר כיסא פנוי][מספר שורה, מספר כיסא פנוי], ...
    public string GetAvailableSeats()
    {
        string availableSeats = "";
        for (int i = 0; i <freeSeats.GetLength(0); i++)
            for (int j = 0; j <freeSeats.GetLength(1); j++)
                if (_______________________)
                    availableSeats = availableSeats +
                                        _______________________________;
        return availableSeats;
    }
    // פעולה המקבלת מספר מושב פנוי ומעדכנת את מצבו כתפוס
    // ומחזירה "אמת" אם הפעולה הצליחה ו"שקר" אחרת
    public bool SetSeatAsTaken(int rowSeat, int colSeat)
    {
            if (__________________)
            {
                _____________________________ = false;
                return _______________;
            }
            else
                return _______________;
     }
} // class Cinema
```

שאלה 12.10

כתבו מחלקה המגדירה את "משחק החיים". משחק החיים הוא משחק סימולציה המתאר את מחזור החיים של יצורים חיים. משחקים במשחק במטריצה שכל תא בה מייצג אתר מִחְיָה: בכל אתר מתקיים אחד משני המצבים:

א."יש חיים" – אתר מחיה מלא – נסמן בתו '1'

ב."אין חיים" – אתר מחיה ריק – נסמן בתו '0'

לדוגמא, בהינתן המטריצה הבאה:

1	1	0	0
0	0	1	0
0	1	0	0
1	1	1	0

אפשר לראות שבאתר (2,1) יש חיים, ושלאתר זה יש 4 שכנים חיים והם (1,2), (3,0), (3,1) ו- (3,2). לעומת זאת באתר (3,3) אין חיים, ולאתר זה יש שכן חי אחד שהוא (3,2).

חוקי הגנטיקה של המשחק:

- **לידה** – בכל אתר שבו "אין חיים" ויש לו בדיוק 3 שכנים חיים תהיה לידה בדור הבא. אחרת האתר נשאר "ללא חיים".
- **מוות** – בכל אתר שבו "יש חיים" ויש לו לכל היותר שכן אחד שבו יש חיים יתרחש מוות בדור הבא כתוצאה מבדידות. בכל אתר שבו "יש חיים" ולו יש 4 שכנים חיים או יותר, יתרחש מוות בדור הבא כתוצאה מצפיפות.
- **קיום** – כל אתר שבו "יש חיים" ויש לו 2 או 3 שכנים חיים ימשיך להתקיים גם בדור הבא.

תהליכי הלידה, המוות והקיום מתרחשים בו זמנית בכל האתרים ויוצרים מצב חיים חדש הנקרא דור חדש.

כתבו מחלקה המגדירה את "מטריצת החיים". המחלקה צריכה להכיל את לוח המשחק; פעולה המבצעת מעבר דור אחד במשחק, לפי הכללים הנתונים ופעולה המציגה את מטריצת החיים באותו רגע. הפעילו את משחק החיים באמצעות הפעולה הראשית המקבלת כקלט מספר שלם X, יוצרת עצם מסוג מטריצת חיים ומקיימת עבורו X דורות. עליכם להציג את המטריצה לאחר כל דור.

הדרכה: את הדור החדש יש להכין במטריצה זמנית, ולהעתיק אותה למטריצת המשחק בתום הכנתה. הגדירו במחלקת מטריצת החיים פעולת עזר (פרטית) המקבלת מציינים של תא ומחזירה את מספר השכנים החיים שיש לתא זה.

שאלה 12.11

כתבו מחלקה המגדירה לוח של משחק "בינגו". המחלקה תכיל את לוח המשחק, מטריצה בגודל 4×4 ובה מספרים שלמים בתחום 1-100, ולוח בוליאני נוסף (באותו גודל) ותפקידו לסמן את התאים של המספרים שהוכרזו במהלך המשחק. במחלקה יוגדרו הפעולות הבאות:

- פעולה המבצעת מהלך במשחק: הפעולה מקבלת כפרמטר מספר שבחר מנהל המשחק. אם המספר נמצא על לוח המשחק, יש לסמן ב-`true` את המקום המתאים בלוח הבוליאני (אם המספר לא נמצא על הלוח לא יתבצע דבר).
- פעולה שתבדוק אם ניתן להכריז על "בינגו". ניתן להכריז על "בינגו" כאשר יש שורה או טור או אלכסון שכל מספריו נבחרו.

שאלות נוספות בנושא מערך דו-ממדי תוכלו למצוא בפרק 13 "פתרון בעיות".

12.2 חיפוש בינרי

בסעיף זה נלמד דרך נוספת לחפש ערך, מלבד החיפוש סדרתי – שכבר נתקלנו בו בעבר. הדרך לעשות זאת היא *חיפוש בינרי*.

בפרק 7 למדנו כיצד לבצע חיפוש סדרתי במעבר על איברי המערך בזה אחר זה ובהשוואתם לערך המבוקש. החיפוש נפסק כאשר נמצא הערך הדרוש או כאשר הגענו לקצה המערך. במקרה הגרוע (כאשר האיבר המבוקש לא נמצא במערך) נעשו N השוואות אם N הוא גודל המערך. כעת נלמד לבצע חיפוש יעיל יותר, כלומר כזה המבצע **פחות** השוואות, **ומסתמך על העובדה שהנתונים ממוינים.** את החיפוש הצגנו בפרק 8 אך כעת נדון בו בהרחבה.

נסו לחשוב כיצד מחפשים שם במדריך הטלפונים?

פותחים את הספר בערך באמצע ובוחנים את השמות הכתובים בעמוד זה. אם השם המבוקש אינו נמצא בעמוד זה, נמשיך את החיפוש רק בעמודים שאחריו או רק בעמודים שלפניו, זאת לפי תוצאת ההשוואה לשמות שבעמוד. באותו אופן ממשיכים בחיפוש באמצעות פתיחת הספר במחצית המתאימה עד שמוצאים את השם המבוקש או עד שמשתכנעים כי השם לא נמצא.

שימו ♥ לחשיבות ההנחה שהספר ממוין. הנחה זו מאפשרת לנו להמשיך את החיפוש רק במחצית אחת של הספר ולהתעלם מהמחצית השנייה לחלוטין.

האלגוריתם שתיארנו נקרא "חיפוש בינרי" כי בכל צעד אנחנו מחלקים את הנתונים לשני חלקים. (משמעות המילה בינרי = מורכב משני חלקים).

הנה אלגוריתם המשתמש בחיפוש בינרי כדי לחפש את הערך `key` במערך **ממוין**:

1. כל עוד יש איברים במערך בצע
   1.1 אם key שווה לאיבר האמצעי אז מצאנו את key במערך
   1.2 אחרת אם האיבר האמצעי גדול מ-key אז
      1.2.1 המשך את החיפוש במחצית התחתונה של המערך
   1.3 אחרת
      1.3.1 המשך את החיפוש במחצית העליונה של המערך

נתון המערך הממוין A להלן, נעקוב אחרי ביצוע האלגוריתם כאשר ערכו של key הוא 50 במערך.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]
20	35	37	42	49	50	52	60	75

A[middle]

אמצע המערך הוא האיבר A[4] וערכו 49. הערך 50 גבוה מ-49, ולכן אם הערך 50 נמצא במערך הרי הוא נמצא בקטע המערך שבין A[5] ל-A[8]:

A[5]	A[6]	A[7]	A[8]
50	52	60	75

A[middle]

נמשיך לחפש את הערך 50 בקטע המערך. אמצע המערך הוא האיבר A[6], וערכו 52. למעשה כאשר אורך המערך הוא זוגי, יש שני איברים שיכולים להיחשב כאמצע המערך – אין חשיבות

באיזה מהם נבחר. הערך 50 קטן מ-52, ולכן נמשיך לחפש בקטע המערך שבין A[5] ל-A[5] (מערך בן איבר אחד!):

A[5]
50

השוואת הערך של `key` ל-A[5] מסתיימת כמובן בהצלחה וניתן להסיק שהערך (50) `key` נמצא במערך A.

שאלה 12.12

נסו לעקוב אחרי ביצוע האלגוריתם כאשר ערכו של `key` הוא 35, וכאשר ערכו של `key` הוא 51.

כיצד נוכל להמשיך את החיפוש רק בחלק מהמערך? הרי המערך כולו שמור בזיכרון, ולא רק חלק ממנו! לשם בדיקה של מחצית המערך בלבד נשתמש בשני אינדקסים (מציינים) `low` ו-`high` שיציינו את גבולות המערך. בתחילת האלגוריתם יהיו האינדקסים שווים לגבולות האמיתיים של המערך (0 ו-N-1) ובמהלך האלגוריתם נצמצם את השטח שבו אנחנו מחפשים. נצמצם את השטח באמצעות משתנה עזר שייקרא `middle`.

חשבו: מה נשים במשתנה `middle` כדי שיציין את אמצע המערך בגבולותיו הנוכחיים?

אמצע המערך יתקבל בחישוב: `middle=(low+high)/2`. אם נרצה לטפל בקטע המערך העליון נתבונן בגבולות: `middle+1..high`, ואילו קטע המערך התחתון יהיה בגבולות: `low..middle-1`.

<u>מהו התנאי לעצירת החיפוש?</u> ברגע שמוצאים את הערך `key` במערך יש לעצור את החיפוש. במקרה שהערך אינו נמצא במערך, נעצור את החיפוש כאשר אי אפשר להקטין עוד את קטע המערך.

<u>כיצד נדע שאי אפשר להקטין עוד את קטע המערך?</u> קבענו כי גבולות קטע המערך הנוכחי הם מ-`low` ועד `high`. אם `low>high` אי-אפשר להקטין עוד את קטע המערך, כי אין איברים בין `A[low]` ל-`A[high]`!

לפניכם קטע תוכנית בשפת C# המממש את האלגוריתם של חיפוש בינרי לחיפוש הערך `key` במערך ממוין A. האלגוריתם מציג הודעה המציינת אם הערך `key` נמצא או לא נמצא במערך A:

```
int[] A;
int key;
…
int middle;
int low = 0;
int high = A.Length-1;
bool found = false;
while (low <= high && !found)
{
      middle = (low + high) / 2;
      if (A[middle] == key)   // האיבר המבוקש נמצא
            found = true;
      else
      {
            if (A[middle]> key)      // מחצית תחתונה
                  high = middle - 1;
            else                     // מחצית עליונה
```

```
                low = middle + 1;
        }
}
if (found)
        Console.WriteLine("{0} found in the array", key);
else
        Console.WriteLine("{0} was not found in the array", key);
```

יעילות

חיפוש בינרי יעיל יותר מחיפוש סדרתי (עבור מערכים ממוינים כמובן). בחיפוש בינרי, לאחר כל בדיקה שבה האיבר המבוקש לא נמצא, מספר האיברים הנותרים במערך קטן בחצי. נניח שלפנינו מערך בגודל 1000 איברים והערך שאנחנו מחפשים **לא** נמצא במערך. **בחיפוש סדרתי**, לאחר הבדיקה הראשונה יישארו לנו עוד 999 איברים לבדוק, לאחר מכן 998 וכך הלאה, בסך הכל נבצע את הלולאה, במקרה הגרוע ביותר, 1000 פעמים. **בחיפוש בינרי**, אחרי הבדיקה הראשונה נותרים לנו 500 איברים, אחרי הבדיקה השנייה 250 איברים, אחרי השלישית 125, וכך הלאה 62, 31, 15, 7, 3, 1. לולאת החיפוש תתבצע לכל היותר 10 פעמים. **שיפור עצום לעומת החיפוש הסדרתי!**

מספר הפעמים שהלולאה מתבצעת תלוי כמובן במיקומו של הערך key במערך. אם הערך key אינו נמצא אז הלולאה תתבצע כמספר הפעמים שאפשר לחלק את גודל המערך לשני חלקים. אפילו כשהמערך גדול מאוד. למשל במערך ובו 1,000,000 איברים תתבצע הלולאה לכל היותר 20 פעמים, יעילות-זמן מרשימה מאוד בהשוואה לחיפוש סדרתי. הטבלה שלהלן מציגה עבור ערכים שונים לגודל המערך, את מספר הפעמים המקסימלי שהלולאה תתבצע בחיפוש בינרי. (מספר הפעמים בפועל תלוי במיקומו של האיבר המבוקש במערך, כאן ההתייחסות היא למקרה הגרוע ביותר).

גודל המערך	מספר הפעמים שהלולאה תתבצע
8	4
10	4
16	5
50	6
100	7
1000	10
5000	13
1000000	20

נקטין את המספר N בחצי ונמשיך כך עד שלא נוכל להקטין אותו יותר (כלומר עד שנגיע ל-1). אם נספור את מספר ההקטנות שעשינו נקבל מספר שנקרא $\log_2 N$. למשל $\log_2 16 = 4$ כי צריך להקטין את 16 4 פעמים בחצי עד שמגיעים ל-1 (8, 4, 2 ו-1).

בעיה 4

מטרת הבעיה ופתרונה: הדגמת שימוש בחיפוש בינרי

בבואו לעבור טסט, מקבל כל נבחן מספר תלת ספרתי. בסוף היום מפורסמים כל המספרים של הנבחנים שעברו את הטסט והם ממוינים בסדר עולה. הגדירו מחלקה המכילה את כל הרשימה של מספרי הנבחנים שעברו טסט, ממוינת בסדר עולה. המחלקה תכיל את הפעולה "האם עברתי?" שתקבל את מספר הנבחן ותחזיר אם הוא נמצא ברשימה או לא.

הגדרת המחלקה *DrivingTest*

הגדרת התכונות

- **pass** – מערך ממוין המכיל את מספרי הנבחנים שעברו.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת מערך ממוין המכיל את מספרי הנבחנים שעברו את הטסט, ומעתיקה אותו למערך pass. בחרנו להעתיק את המערך כדי להגן עליו מפני שינוי מבחוץ.
- **DidIPass** – פעולה המקבלת מספר נבחן ומחזירה "אמת" אם מספר זה נמצא במערך העוברים ו"שקר" אחרת. בפעולה זו נוכל לבצע חיפוש בינרי מכיוון שהמערך ממוין.

מימוש המחלקה

```
/*
   DrivingTest המחלקה
*/
public class DrivingTest
{
      private int[] pass;
      //פעולה בונה
      public DrivingTest(int[] p)
      {
            pass = new int[p.Length];
            for (int i = 0; i < p.Length; i++)
                  pass [i] = p[i];
      }
      public bool DidIPass(int num)
      {
            int middle;
            int low = 0;
            int high = pass.Length - 1;
            while (low <= high)
            {
                  middle = (low + high) / 2;
                  if (pass[middle] == num)          // האיבר המבוקש נמצא
                        return true;
                  else
                  {
                        if (pass[middle] > num)         // מחצית תחתונה
                              high = middle - 1;
                        else                            // מחצית עליונה
                              low = middle + 1;
                  }
            }//while
            return false;
      }
}//class DrivingTest
```

סוף פתרון בעיה 4

שאלה 12.13

לפניכם קטע קוד שמבצע חיפוש בינרי במערך arr :

הניחו כי המשתנים high ו-low מאותחלים בגבולות המערך 0 ו-N-1.

```
bool flag = true;
while (low <= high)
{
      middle = (low + high) / 2;
      if (arr[middle] != num)
            flag = false;
      if (arr[middle] > num)
            high = middle - 1;
      else
            low = middle + 1;
}
if (flag)
      Console.WriteLine("The Value Is In The Array");
else
      Console.WriteLine("The Value Is Not In The Array");
```

הקטע שגוי. מצאו את השגיאה ותקנו את הקטע כך שיבצע את הנדרש.

שאלה 12.14

נתון מערך ממוין A. כתבו קטע תוכנית בשפת C# שיבדוק כמה פעמים מופיע הערך 25 במערך A. כתבו את הקטע יעיל ככל שתוכלו.

שאלה 12.15

נתון מערך A באורך 16 שאיבריו הם מספרים שלמים מ-1 עד 16, ממוינים בסדר עולה. עבור כל אחד מהערכים של key מ-1 ועד 16 כמה פעמים מתבצעת הלולאה בחיפוש בינרי?

שאלה 12.16

נתון מערך ממוין A בגודל N ובו מספרים שלמים חיוביים (בסדר עולה). כתבו קטע תוכנית שהפלט שלו הוא הודעה אם **רוב** איברי המערך גדולים מן הממוצע של האיבר הראשון והאחרון.

שאלה 12.17

נתון מערך ממוין A בגודל N ובו מספרים שלמים חיוביים עוקבים, מלבד שניים מן המספרים.

א. כתבו קטע תוכנית שהפלט שלו הוא הודעה אם זוג הערכים הלא רצופים נמצאים בחצי הראשון של המערך, בחצי השני של המערך או בדיוק במעבר בין החצי הראשון לחצי השני (N זוגי).

ב. כתבו קטע תוכנית שהפלט שלו הוא **המציינים** של שני איברי המערך שמצאתם בסעיף הקודם, כלומר מיקומם במערך.

שאלה 12.18

מטריצה "ממוינת למחצה" היא מערך דו-ממדי שכל שורה בה ממוינת בסדר עולה. הגדירו מחלקה ובה מטריצה "ממוינת למחצה". במחלקה הגדירו פעולה שתקבל כפרמטר ערך X ותדפיס את מיקומו של X במטריצה, כלומר את מספר השורה והעמודה שהערך X נמצא בהן. נתון שהערך X מופיע בדיוק פעם אחת במטריצה.

הדרכה : השתמשו בפעולת עזר פרטית שתחפש בכל שורה בנפרד.

שאלה 12.19

"סדרת הפרשים עולה" היא סדרת ערכים שבה ההפרשים בין כל זוג ערכים סמוכים עולים-ממש. כלומר ההפרש הגדול ביותר הוא בין האיבר האחרון בסדרה וזה שלפניו, וההפרש הקטן ביותר הוא ההפרש בין האיבר הראשון לשני. דוגמה ל"סדרת הפרשים עולה": 97, 66, 40, 29, 20, 15.

עליכם לכתוב אלגוריתם שיקלוט "סדרת הפרשים עולה" במערך ומספר נוסף k. האלגוריתם יבדוק אם קיים זוג ערכים סמוכים במערך שהפרשם הוא k. ממשו את האלגוריתם בשפת #C. עליכם לכתוב את האלגוריתם יעיל ככל שתוכלו מבלי להשתמש במערך עזר.

12.3 מיונים

הצורך במיון נתונים מופיע שוב ושוב: מדריך טלפונים ממוין לפי שמות, טבלת הליגה בכדורגל ממוינת לפי מספר הנקודות ועוד... בעיית המיון היא בעיה מרתקת במדעי המחשב, מספר רב של אלגוריתמים פותחו לשם כך, ולכל אלגוריתם יתרונות וחסרונות משלו. בסעיף זה נציג שלושה אלגוריתמים שונים למיון.

מיון בחירה

חשבו: מונחת לפניכם ערמה של מטבעות. איך אפשר למיין אותם לפי ערכן?

השיטה הטבעית היא לאסוף את כל המטבעות הקטנות ביותר של 5 אגורות, אחר כך את המטבעות של 10 אגורות, וכך הלאה עד שנותרת רק ערימת מטבעות של 10 שקלים. שיטה טבעית זו היא הבסיס לאלגוריתם המיון שנקרא "מיון בחירה".

הנה אלגוריתם המשתמש במיון בחירה כדי למיין סדרת מספרים:

```
עבור כל i מ-1 עד אורך הסדרה בצע
    השם במיקום ה-i את הערך הקטן ביותר מבין הערכים שבהם לא טיפלנו
```

נניח שמצאנו את הערך הקטן ביותר בסדרה. האלגוריתם דורש לשים אותו במקום הראשון. אבל מקום זה תפוס בידי ערך אחר, ואם נשים במקומו את הערך הקטן ביותר נאבד את הערך שנמצא במקום הראשון. **חשבו**: היכן ניתן לשמור את הערך שנמצא במקום הראשון? נשמור אותו במקום שהתפנה מהאיבר הקטן ביותר. אם האיבר הקטן ביותר נמצא במקום j, נחליף בין האיבר הנמצא במקום הראשון לאיבר הנמצא במקום ה-j.

נתבונן לדוגמה במערך A הבא:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
2	8	20	4	7	0	15	18	1	11

הערך הקטן ביותר 0, נמצא ב-A[5], אם ברצוננו להעביר את 0 ל-A[0] אז נחליף את A[5] ב-A[0]:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
0	8	20	4	7	2	15	1	18	11

דוגמה זו מראה את הצעד הראשון באלגוריתם למיון. הצעד השני יהיה מציאת הערך הקטן ביותר בשארית המערך A. כלומר מבין הערכים שנמצאים בתאים A[1] עד A[9]. הערך העונה על הדרישה הוא 1 השמור ב-A[7], ולכן נחליף אותו עם הערך 8 הנמצא ב-A[1]:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]	A[7]	A[8]	A[9]
0	1	20	4	7	2	15	8	18	11

שאלה 12.20

שרטטו תרשים של המערך לאחר שתי ההחלפות הבאות.

עתה אפשר להשלים את האלגוריתם למיון המערך A :

```
עבור כל i מ-0 עד A.Length-2 בצע:
    מצא את הערך הקטן ביותר בקטע המערך A[i]..A[A.Length-1]
        ושמור את המציין שלו ב-iMin
    החלף את האיבר A[i] עם האיבר A[iMin]
```

שיטת המיון שתיארנו כאן נקראת "מיון בחירה" (Selection Sort). בכל שלב באלגוריתם בוחרים את האיבר הקטן ביותר בשארית המערך ומכניסים אותו למקום המתאים.

בעיה 5

מטרת הבעיה ופתרונה: הצגת מיון בחירה בשפת #C.

> מנהלת בית הספר החליטה כי בספריית בית הספר, הספרים יהיו מסודרים לפי עוביים. עזרו לספרן המבולבל לסדר את הספרים. הגדירו מחלקה ובה מערך ובכל תא בו יש מספר עמודים של ספר אחד. במחלקה תהיה הפעולה **SelectionSort** שבעת הצורך תמיין את המערך מהערך הקטן לגדול.

הגדרת המחלקה *Library*

הגדרת התכונות

- **bookSizes** – מערך חד-ממדי לשמירת עוביו של כל ספר.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת מערך המכיל את עובי הספרים, ותעדכן את המערך bookSizes.
- **SelectionSort** – פעולה הממיינת את המערך שבמחלקה לפי אלגוריתם "מיון בחירה".
- **GetBookSizes** – פעולת גישה המחזירה את המערך הממוין.

מימוש המחלקה

```
/*
      Library המחלקה
*/
public class Library
{
    private int[] bookSizes;
    // פעולה בונה
    public Library(int[] books)
```

```
    {
        bookSizes = books;
    }
    public void SelectionSort()
    {
        int iMin, temp;
        for(int i = 0; i < bookSizes.Length - 1; i++)
        {
            iMin=i;
            // מציאת מינימום
            for(int j = i + 1; j < bookSizes.Length; j++)
                if(bookSizes[j] < bookSizes[iMin])
                    iMin = j;
             // החלפת איברים
             temp = bookSizes[iMin];
             bookSizes[iMin] = bookSizes[i];
             bookSizes[i] = temp;
        }
     }
     public int[] GetBookSizes()
     {
        return bookSizes;
     }
} // Library
```

סוף פתרון בעיה 5

שאלה 12.21

שנו את פעולת המיון בבעיה 5 כך שהיא תזמן שתי פעולות עזר במקום לטפל בכל תתי-המשימות בעצמה. פעולות העזר הן: פעולה המחזירה את המציין של הערך המינימלי בקטע נתון של המערך ופעולה המחליפה בין שני ערכים במערך. לשתי הפעולות יתקבלו אינדקסים כפרמטרים. את הפעולות יש לממש כפעולות פרטיות במחלקה Library.

יעילות

חשבו: אם N הוא מספר איברי המערך, כמה פעמים מתבצע גוף הלולאה במיון בחירה?

עבור i = 0, הלולאה הפנימית מתבצעת N-1 פעמים

עבור i = 1, הלולאה הפנימית מתבצעת N-2 פעמים

עבור i= 2, הלולאה הפנימית מתבצעת N-3 פעמים

. . . . .

עבור N-1 = i הלולאה הפנימית מתבצעת N-(N-1) = 1 פעמים

לפי הנוסחה לסכום סדרה חשבונית נקבל שמספר הפעמים שהלולאה מתבצעת הוא:

$$1 + 2 + . . . + (N – 1) = \frac{(N-1)\cdot N}{2}$$

מיון בחירה של N מספרים דורש בערך N^2 ביצועים של גוף הלולאה. מספר זה גדל בקצב גבוה ומגיע לחצי מיליון עבור מערך בן 1000 איברים.

שאלה 12.22

האם מספר הפעמים שהלולאה מתבצעת במיון בחירה תלוי בתוכן המערך?

שאלה 12.23

כמה פעולות החלפה מתבצעות במיון בחירה?

שאלה 12.24

התבוננו בפעולה SelectionSort (בבעיה 5). כמה פעמים מתבצע המשפט `iMin = j` בפעולה למיון בחירה?

שאלה 12.25

כתבו אלגוריתם המקבל כקלט סדרת מספרים בסדר כלשהו. האלגוריתם יציג את מספר הערכים שמיקומם המקורי הוא גם מיקומם בסדרה **המסודרת** של הערכים הנתונים.

הדרכה: קלטו את האיברים במערך, מיינו במיון בחירה ומנו תוך כדי כך כמה איברים נמצאו במיקומם.

שאלה 12.26

הגדירו מחלקה ובה מערך חד-ממדי בגודל N. הגדירו במחלקה פעולה שתקבל כפרמטר מספר שלם חיובי K קטן מ-N, וערך S. הפעולה תחזיר "אמת" אם קיימים K מספרים במערך שסכומם קטן מ-S ו"שקר" אחרת.

מיון הכנסה

בתת-סעיף זה נציג מיון נוסף הנקרא מיון הכנסה. זמן הביצוע של מיון הכנסה אמנם דומה לזה של מיון בחירה, אך על סוגי קלט רבים מיון הכנסה יעיל יותר.

מיון הכנסה מבוסס על הדרך שבה שחקן קלפים מסדר יד קלפים. השחקן מרים את הקלפים אחד-אחד ושומר על הקלפים שבידו באופן ממוין. כשהוא מקבל קלף חדש הוא מפנה לו מקום, ומכניס אותו למקום הנכון ביד.

יד עם קלף אחד:

10

אחרי קבלת הקלף 4:

4	10

אחרי קבלת הקלף 8:

4	8	10

אחרי קבלת הקלף 2:

2	4	8	10

ברור שלאחר ביצוע צעדי ההכנסה עבור כל הקלפים שקיבלנו יהיו הקלפים ממוינים. הפעולה שבבסיס מיון הכנסה היא הצעד "הכנס ערך חדש למערך ממוין". כעת נתחיל בדיון לפיתוח אלגוריתם לביצוע פעולה זו.

הכנסת איבר במערך ממוין

הכנסת איבר חדש למערך תגרום להזזת מספר איברים.

חשבו: מי הם האיברים במערך שצריכים לזוז? האיברים שצריכים לזוז הם האיברים שערכיהם גדולים מהאיבר שנרצה להכניס. התרשימים שלהלן מראים את המערך A ובו 5 ערכים, ואת

הכנסת הערך החדש 85. האיברים שצריכים לזוז "ימינה" הם A[4] ו- A[3]. לאחר ההזזה, אפשר להכניס את הערך החדש ל-A[3] שיהיה המקום המקורי של הערך 92:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
59	63	75	92	98		
59	63	75	92	92	98	
59	63	75	85	92	98	

התבוננו באלגוריתם הבא להכנסת איבר element למערך ממוין:

1. השם ב-i את מספר האיברים הממוינים במערך פחות אחת
2. כל עוד האיבר במקום ה-i גדול מ-element בצע
   2.1. הזז את האיבר מקום אחד ימינה
   2.2. הקטן את i ב-1
3. הכנס את element במקום שהתפנה

שאלה 12.27

הכניסו את הערך 6 למערך הנתון, לפי האלגוריתם שתואר למעלה:

5	8	20	22	25		

חשבו: מה יקרה כאשר האיבר להכנסה יהיה קטן יותר מכל איברי המערך? **האלגוריתם לא יעצור ותהיה גלישה מהמערך!** כדי למנוע מצב כזה נשנה את תנאי הכניסה ללולאה כך שיכיל גם את התנאי "אם הגענו לקצה המערך". התנאי החדש בשורה השנייה של האלגוריתם יהיה:

2. כל עוד i >= 0 וגם האיבר במקום ה-i גדול מ-element בצע

בעיה 6

מטרת הבעיה ופתרונה: הצגת מחלקה ובה הכנסת איבר למערך ממוין.

המורה להיסטוריה צופיה, החליטה לשמור את רשימת תלמידיה בצורה ממוינת לפי ציונם בהיסטוריה. כך אם היא תידרש לאתר את התלמידים המצטיינים בעבור האולימפיאדה בהיסטוריה, תוכל לאתרם בקלות. עליכם לבנות מערכת שתעזור למורה צופיה לנהל את ציוני תלמידיה. לשם כך עליכם להגדיר מחלקת "תלמיד" המכילה את התכונות: שם תלמיד וציונו בהיסטוריה, ומחלקת "מורה" המגדירה מערך של תלמידים הממוין לפי ציוניהם, ופעולה המאפשרת להכניס תלמיד למערך. מערך התלמידים יישאר ממוין גם לאחר הכנסה של תלמידים נוספים. המערכת תאפשר לצופיה להוסיף תלמידים, ולשלוף מידע לפי שם תלמיד.

הגדרת המחלקה *Student*

הגדרת התכונות

- **name** – שם התלמיד, מטיפוס מחרוזת.
- **historyGrade** – ציונו של התלמיד בהיסטוריה, מטיפוס ממשי.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת שם וציון, ומעדכנת את תכונות המחלקה בהתאם.
- **GetName** – פעולה המאחזרת את שם התלמיד.
- **GetHistoryGrade** – פעולה המחזירה את ציונו של התלמיד.

מימוש המחלקה

```
/*
        Student המחלקה
*/
public class Student
{
    private double historyGrade;
    private string name;
    // פעולה בונה
    public Student(string name, double historyGrade)
    {
        this.historyGrade = historyGrade;
        this.name = name;
    }
    // פעולות גישה //
    public double GetHistoryGrade()
    {
        return historyGrade;
    }
    public string GetName()
    {
        return name;
    }
}//class Student
```

הגדרת המחלקה *Teacher*

הגדרת התכונות

- **students** – מערך לשמירת התלמידים, יישאר ממוין לאחר כל הכנסה.
- **numOfStudents** – מספר התלמידים השמורים אצל המורה עד עכשיו (במערך students).

הגדרת הפעולות

- **פעולה בונה** – הפעולה תקצה זיכרון עבור מערך התלמידים, ותאתחל את מספר התלמידים ב-0 (בהתחלה אין תלמידים במערך).
- **InsertStudent** – פעולה המקבלת "תלמיד" ומכניסה אותו למקום המתאים במערך. הפעולה תעדכן את מספר התלמידים. פעולה זו תמומש לפי האלגוריתם שהוצג לעיל להכנסת איבר למערך ממוין. הנחות הכרחיות: יש מקום במערך. המערך ממוין.
- **GetStudents** – פעולה המחזירה את מערך התלמידים.
- **GetStudentByName** – פעולה המקבלת שם תלמיד ומחזירה את התלמיד בשם המבוקש.

מימוש המחלקה

```
/*
        Teacher המחלקה
*/
public class Teacher
{
    const int NUM_OF_STUDENTS = 40;
    private Student[]students;
    private int numOfStudents;
    // פעולה בונה
    public Teacher()
    {
        students= new Student [NUM_OF_STUDENTS];
        numOfStudents=0;
    }
    public Student[] GetStudents()
    {
        Student[] studs = new Student[numOfStudents];
        for(int i = 0; i < numOfStudents; i++)
            studs[i] = new Student(students[i].GetName(),
                                    students[i].GetHistoryGrade());
        return studs;
    }
    public Student GetStudentByName(string name)
    {
        for(int i = 0; i < numOfStudents; i++)
            if (students[i].GetName() == name)
                return new Student(students[i].GetName(),
                                    students[i].GetHistoryGrade());
        return null; // לא נמצא תלמיד בשם זה
    }
    public void InsertStudent(Student stud)
    {
        int i = numOfStudents - 1;
        while (i >= 0 &&
              students[i].GetHistoryGrade() > stud.GetHistoryGrade())
        {
                students[i+1] = students[i];
                i--;
        }//while
        students[i+1] = new Student(stud.GetName(),
                                    stud.GetHistoryGrade());
        numOfStudents++;
    }
}//class Teacher
```

סוף פתרון בעיה 6

שאלה 12.28

א. כמה דוגמאות מייצגות שונות של המשתנים `numOfStudents`, `stud` ו-`students` כדאי לבדוק כדי להשתכנע בנכונות הפעולה `InsertStudent`? הראו דוגמאות כאלה.

ב. כמה פעמים תתבצע הלולאה עבור כל דוגמה מייצגת שבחרתם?

ג. בחרו שתי דוגמאות מייצגות מסעיף א והראו באמצעות איור, את המערך students לאחר כל ביצוע-חוזר של הלולאה, ובסיום ביצוע הלולאה.

ד. כמה פעמים תתבצע הלולאה עבור כל דוגמה מייצגת שבחרתם?

שאלה 12.29

כתבו פעולה ראשית לבעיה 6. הפעולה הראשית תיצור עצם מסוג Teacher שאליו יוכנסו תלמידים באמצעות הפעולה InsertStudent. לאחר מכן קלטו שם של תלמיד וחפשו אותו באמצעות הפעולה GetStudentByName והציגו את ציון התלמיד. לבסוף זמנו את הפעולה GetStudents לקבלת המערך הממוין והציגו את רשימת התלמידים שציוניהם מעל ל-80.

מיון הכנסה במערך שלם

אנו חוזרים לאלגוריתם למיון הכנסה של מערך A. ניתן להתייחס לאיבר הראשון של המערך שנמצא ב-A[0] כאל מערך ממוין באורך 1. האלגוריתם מתעלם משאר איברי המערך ומכניס למקום הנכון את הערך שנמצא ב-A[1]. כתוצאה מכך מתקבל מערך ממוין באורך 2 המורכב מ-A[0] ומ-A[1] . הצעד הבא: הכנסת הערך שנמצא ב-A[2] למקום הנכון וכן הלאה. להלן מערך A לפני תחילת המיון. סימנו באפור את קטע המערך שכבר מוין:

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
40	35	15	38	42	17	39

התרשימים שלהלן מראים את המערך לאחר הכנסת האיברים שבמקומות A[1], A[2], A[3], A[4], לפי הסדר. שימו ♥ שהאיבר A[4] נמצא במקומו כבר לאחר הכנסת A[3], ואינו זז כתוצאה מפעולה ההכנסה.

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
35	**40**	15	38	42	17	39

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
15	**35**	**40**	38	42	17	39

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
15	**35**	**38**	**40**	42	17	39

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	A[6]
15	**35**	**38**	**40**	**42**	17	39

נסכם ונאמר שהאלגוריתם של מיון הכנסה מניח שקטע המערך בין A[0] ל-A[k] ממוין, ומכניס למקום הנכון את האיבר שנמצא ב-A[k+1]. כתוצאה מכך מתקבל מערך ממוין בין A[0] ל-A[k+1]. הכנסת האיבר ה-A[k+1] למערך הממוין תיעשה באמצעות האלגוריתם "הכנס איבר למערך ממוין" שכבר פיתחנו בבעיה 6. בכל הכנסה למערך, אנו נעזרים במשתנה המציין את מספר האיברים הממוינים במערך. ערכו של משתנה זה גדל באחת לאחר כל הכנסה.

אם נרצה לכתוב פעולה הממיינת את המערך כולו באמצעות פעולת ההכנסה, נעבור על המערך מהאיבר השני ואילך, כל איבר יישלח בתורו לפעולת ההכנסה, והיא תכניס אותו למקומו בחלק הממוין של המערך. כך נעשה בפתרון הבעיה הבאה.

בעיה 7

מטרת הבעיה ופתרונה: הצגת אלגוריתם למיון הכנסה.

הגדירו מחלקה ובה מערך של מספרים שלמים. הגדירו במחלקה פעולה בשם **InsertionSort** הממיינת את המערך באמצעות האלגוריתם למיון הכנסה. המחלקה תשתמש בפעולה פרטית "הכנס איבר למערך ממוין".

הגדרת המחלקה *SortedArray*

הגדרת התכונות

- **numbers** – מערך של מספרים שלמים.
- **sorted** – מספר האיברים הממוינים במערך עד כה.

הגדרת הפעולות

- **פעולה בונה** – הפעולה מקבלת מערך חד-ממדי, ומעדכנת את המערך numbers בהתאם ומאתחלת ב-1 את מספר האיברים הממוינים (מכיוון שהאיבר הראשון ממוין ביחס לעצמו בלבד).
- **InsertElement** – פעולה פרטית המקבלת איבר להכנסה, ומכניסה את האיבר בחלק הממוין של המערך, כך שהמערך יישאר ממוין. הפעולה מקדמת ב-1 את מספר האיברים הממוינים במערך.
- **InsertionSort** – פעולה הממיינת את המערך numbers לפי מיון הכנסה. (בשימוש ב-**InsertElement**)
- **GetNumbers** – פעולת גישה המחזירה את המערך הממוין.

מימוש המחלקה

```
/*
     SortedArray המחלקה
*/
public class SortedArray
{
    private int[] numbers;
    private int sorted;
    // פעולה בונה
    public SortedArray(int[] a)
    {
        numbers = a;
        sorted = 1;
    }
    // הכנס למערך ממוין, פעולה פרטית
    private void InsertElement(int element)
    {
        int i = sorted -1 ;
        while (i >= 0 && numbers[i] > element)
        {
```

```
                numbers[i+1] = numbers[i];
                i--;
            }//while
            numbers[i+1] = element;
            sorted++;
        }
        // מיון הכנסה
        public void InsertionSort()
        {
            for(int i = 1; i < numbers.Length; i++)
                InsertElement(numbers[i]);
        }
        public int[] GetNumbers()
        {
           return numbers;
       }
} // class SortedArray
```

יעילות

שימו ♥: אם N הוא מספר איברי המערך, כמה פעמים מתבצע גוף הלולאה הפנימית במיון הכנסה (הלולאה בפעולה `InsertElement`)? החישוב דומה לחישוב שעשינו עבור מיון בחירה. גם כאן יש סדרה חשבונית עולה: בזימון הראשון ל- `InsertElement` ייתכן שיש להזיז איבר אחד במערך, בזימון השני שני איברים וכן הלאה עד שבזימון האחרון i=N-1 וייתכן שיש להזיז את כל N-1 האיברים. במקרה הגרוע מיון הכנסה דורש בערך N^2 ביצועים של גוף הלולאה, וראינו שמספר זה גדל במהירות ככל ש-N גדל.

נשווה מיון הכנסה למיון בחירה לפי סוגי קלט שונים. למיון בחירה יש יתרון כאשר האיברים הראשונים במערך גדולים, כי בכל מעבר בלולאה יש בדיוק החלפה אחת של איברים. לעומת זאת במיון הכנסה האיברים הגדולים "זוחלים" לאט לאט למקומם. היתרון של מיון הכנסה בולט כאשר מנסים למיין מערך שכבר ממוין או שכמעט ממוין. במקרה זה הלולאה בפעולה `InsertElement` תתבצע מספר מועט של פעמים, כי הערך `element` יהיה כמעט תמיד גדול מהאיברים בחלק הממוין של המערך ולכן לא ניכנס ללולאה המזיזה איברים ומפנה לו מקום.

סוף פתרון בעיה 7

שאלה 12.30

תארו מה קורה במיון הכנסה כאשר מנסים למיין מערך ממוין.

שאלה 12.31

אפיינו את המערך הגורם לביצועים הגרועים ביותר במיון הכנסה.

שאלה 12.32

מה צריך לשנות במחלקה SortedArray כדי לקבל מיון הכנסה בסדר יורד?

שאלה 12.33

נתון מערך לא ממוין בגודל N, עליכם לבדוק כמה ערכים שונים יש במערך.

הדרכה: הגדירו מחלקה המכילה מערך לא ממויין כתכונה. כתבו פעולה הממיינת את המערך וסופרת תוך כדי המיון את מספר הערכים השונים במערך. הפעולה תחזיר את המספר שנמצא.

שאלה 12.34

נתון מערך לא ממוין בגודל N המכיל מספרים החוזרים על עצמם. עליכם למיין את המערך ולגרום לכך שכל מספר יופיע בו פעם אחת בלבד.

הדרכה: הגדירו מחלקה המכילה מערך לא ממוין כתכונה. כתבו פעולה הממיינת את המערך במיון הכנסה, ללא הכנסת מספרים חוזרים. הוסיפו פעולה המחזירה את המערך הממוין.

מיון בועות

מיון בועות הוא מיון המתבסס על השוואת כל זוג ערכים צמודים, ומחליף ביניהם אם הם לא מסודרים בסדר המבוקש. כלומר אם נרצה למיין בסדר עולה, נשווה תחילה את האיבר הראשון לשני ונחליף ביניהם אם הראשון גדול מהשני. לאחר מכן, נשווה את האיבר השני לשלישי ונחליף ביניהם אם צריך. כך נמשיך עד לזוג האחרון במערך. כתוצאה ממעבר כזה על כל הזוגות הצמודים, האיבר בעל הערך הגבוה ביותר יגיע למקומו – לסוף המערך.

מעבר נוסף על כל הזוגות הצמודים יגרום לאיבר השני בגודלו להגיע למקומו – מקום אחד לפני האיבר האחרון. בכל מעבר, איבר אחד יגיע למקומו. חשבו כמה מעברים צריכים לבצע כדי שכל האיברים יגיעו למקומם?

הנה אלגוריתם הממיין מערך של מספרים שלמים לפי הרעיון הנתון:

עבור כל i מ-1 עד A.Length-1 בצע:
  עבור כל j מ-0 עד A.Length-1-i בצע:
    אם a[j] > a[j+1]
      החלף את האיבר A[j] ואת האיבר A[j+1]

מיון בועות נקרא כך כיוון שבכל שלב "מבעבעים" את האיבר הגדול למקומו.

שאלה 12.35

לפניכם קטע תוכנית הממיין בשיטת "מיון בועות" את המערך A ובו מספרים שלמים. השלימו את החלקים החסרים במיון: (שימו ♥: המערך צריך להיות ממוין בסדר עולה).

```
int temp;
for (int i = 1 ; i < ___________; i++)
{
      for (int j = 0 ; j < ____________; j++)
      {
            if (A[____] > A[____])
            {
                  temp = ___________;
                  A[____] = ___________;
                  A[____] = _________;
            }
      }
}
```

12.4 מיזוג

בסעיף זה נלמד אלגוריתם לבעיית המיזוג.

מיזוג הוא שילוב של קלט משני מקורות לפלט יחיד.

הקלטים ממוינים והפלט חייב להיות ממוין גם הוא.

נניח שברשותנו שני מערכים ממוינים A ו-B המכילים ערכים מסוג מסוים (תווים, מספרים שלמים, ממשיים או כל טיפוס סדור אחר). נרצה למזג את שני המערכים במערך C. האלגוריתם יסרוק במקביל את המערך A באמצעות המציין iA ואת המערך B באמצעות המציין iB, ויבחר את האיבר המתאים (מבין האיבר ב-A והאיבר ב-B) ויכניס אותו למערך C. המשתנה iC יציין את המיקום שאליו צריך להכניס את האיבר הבא במערך C.

חשבו: מה צריך להיות גודל המערך C אם גודלם של המערכים A ו-B הוא A.Length ו-B.Length?

כך נראים המערכים A, B ו-C לפני תהליך המיזוג (בדוגמה זו המערכים מכילים איברים מטיפוס תווי):

	A[0]	A[1]	A[2]	A[3]
A:	F	G	K	R
	iA			

	B[0]	B[1]	B[2]	B[3]	B[4]
B:	J	L	M	T	X
	iB				

	C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
C:									
	iC								

כדי שהמערך C יהיה ממוין בסוף תהליך המיזוג נבחר בכל שלב להכניס את האיבר הקטן מבין האיברים שב-A וב-B. להלן תרשים של המערכים לאחר הצעד הראשון המעתיק את האיבר הראשון של A לאיבר הראשון של C. שימו לב לקידום המציינים.

A[0]	A[1]	A[2]	A[3]
F	G	K	R
	iA		

B[0]	B[1]	B[2]	B[3]	B[4]
J	L	M	T	X
iB				

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
F								
	iC							

המערכים לאחר הצעד השני:

A[0]	A[1]	A[2]	A[3]
F	G	K	R
		iA	

B[0]	B[1]	B[2]	B[3]	B[4]
J	L	M	T	X
iB				

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
F	G							
		iC						

ולאחר הצעד השלישי:

A[0]	A[1]	A[2]	A[3]
F	G	K	R
		iA	

B[0]	B[1]	B[2]	B[3]	B[4]
J	L	M	T	X
	iB			

C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
F	G	J						
			iC					

שאלה 12.36

ציירו את המערך C ואת המציין iC אחרי הצעד הרביעי באלגוריתם ואחרי הצעד השישי.

הביצוע החוזר יסתיים כאשר באחד משני המערכים לא יישארו איברים. ייתכן כי במערך השני נשאר "זנב": איברים שלא נכנסו למערך C, ויש להעתיקם למערך C.

נכתוב את האלגוריתם למיזוג מערך:

```
1.  כל עוד נשארו איברים ב-A וגם ב-B בצע
    1.1  אם האיבר ב-A קטן מהאיבר ב-B אז
         1.1.1  העתק את האיבר מ-A ל-C
         1.1.2  הגדל את המציין של A ב-1
    1.2  אחרת
         1.2.1  העתק את האיבר מ-B ל-C
         1.2.2  הגדל את המציין של B ב-1
    1.3  הגדל את המציין של C ב-1
2.  כל עוד נשארו איברים ב-A בצע        //"זנב" של A – אם יש
    2.1  העתק את האיבר מ-A ל-C
    2.2  הגדל את המציין של A ב-1
    2.3  הגדל את המציין של C ב-1
3.  כל עוד נשארו איברים ב-B בצע        //"זנב" של B – אם יש
```

3.1 העתק את האיבר מ-B ל-C 3.2 הגדל את המציין של B ב-1 3.3 הגדל את המציין של C ב-1

שימו ♥: רק אחת מבין שתי הלולאות שבשורות 2 ו-3 תתבצע. לעולם לא שתיהן. חשבו מדוע.

שאלה 12.37

עקבו אחר ביצוע האלגוריתם שתיארנו עבור המערכים A ו-B הבאים:

A[0]	A[1]	A[2]	A[3]
12	27	70	93

B[0]	B[1]	B[2]	B[3]	B[4]
12	27	70	93	97

שאלה 12.38

מהו ה"זנב" המתקבל ממיזוג המערכים בשאלה הקודמת?

שאלה 12.39

לפניכם קטע תוכנית בשפת C#, למיזוג המערכים A ו-B במערך C, השלימו את הקטע במקומות החסרים:

```
int iA = ______;
int iB = ______;
int iC = ______;
while (iA < A.Length && iB < B.Length)
{
    if(A[iA] ___ B[iB])
    {
        C[____] = A[____];
        ____++;
    }
    else
    {
        C[____] = B[____];
        ____++;
    }
    ____++;
}
while (________________)
{
    C[iC] = A[iA];
    iA++;
    iC++;
}
while (_________________)
{
    C[iC] = B[iB];
    iB++;
    iC++;
}
```

שאלה 12.40

תנו דוגמאות של ערכים התחלתיים במערכים A ו-B שעבורן:

א. ערכו של iA נשאר 0 לאורך ביצוע לולאת ה-while הראשונה.

ב. ערכו של iB נשאר 0 לאורך ביצוע לולאת ה-while הראשונה.

ג. ערכי iA ו-iB עולים לסירוגין: ערכו של iA עולה ב-1, אחר כך ערכו של iB עולה ב-1, אחר כך ערכו של iA עולה ב-1, וכן הלאה עד ל"זנב".

שאלה 12.41

כמה פעמים מתבצעות הלולאות במיזוג? חשבו את המספר הכולל של הלולאות, לרבות ה"זנבות".

שאלה 12.42

נתונים שני מערכים ממוינים sortedA ו-soredtB. כתבו קטע תוכנית בשפת C# שיציג את ערכו של האיבר הקטן ביותר המופיע בשני המערכים.

למשל עבור המערכים:

sortedA = 1 ,5 ,6 ,8 ,10

sortedB = 2 ,4 ,6 ,10 ,20

יודפס הערך: 6

שאלה 12.43

"רשימות זרות" הן שתי רשימות ערכים שאין להן אף ערך משותף. כתבו קטע תוכנית הבודק אם שני המערכים הממוינים A ו-B הם רשימות זרות.

שאלה 12.44

נתונים שני מערכים ממוינים, אחד בסדר עולה והשני בסדר יורד. כתבו קטע תוכנית הממזג את שני המערכים כך שהמערך הממוזג יהיה ממוין בסדר עולה.

סיכום

בסעיף 12.1 הוצגו בעיות שפתרונן מצריך שמירה של ערכים מאותו טיפוס בצורה של טבלה. ניתן לשמור ערכים אלה כמערך דו-ממדי.

- **גישה לאיבר במערך דו-ממדי** מתבצעת באמצעות: שם מערך ושני מציינים (אינדקסים), הראשון מציין את מספר השורה והשני מציין את מספר האיבר בשורה (מספר העמודה).
- **סריקה מלאה** של כל איברי המערך מתבצעת באמצעות לולאה מקוננת, כלומר לולאה בתוך לולאה. הלולאה החיצונית עוברת על שורות המערך והפנימית על כל האיברים בשורה.
- **מטריצה ריבועית** היא מערך דו-ממדי שמספר השורות שלו שווה למספר העמודות שלו.
- **אלכסון ראשי** במטריצה ריבועית הוא אוסף כל האיברים שמספר השורה שלהם שווה למספר העמודה שלהם.
- **אלכסון משני** במטריצה ריבועית הוא אוסף כל האיברים שסכום ערכי השורה והעמודה שלהם שווה למספר השורות פחות 1.

בסעיפים 12.2, 12.3 ו-12.4 הוצגו אלגוריתם לחיפוש בינרי ושלושה אלגוריתמים למיון: מיון הכנסה, מיון בחירה ומיון בועות והוצג אלגוריתם למיזוג.

♦ **חיפוש בינרי** מתאים רק למערך ממוין, והוא יעיל בהרבה מהאלגוריתם הסדרתי לחיפוש מכיוון שהוא מצמצם את טווח החיפוש בחצי בכל ביצוע של הלולאה.

♦ **במיון בחירה** מחפשים את הערך הקטן ביותר בקטע המערך שטרם מוין ומחליפים בינו ובין האיבר שנמצא במיקום הראשון בקטע. עבור מערך בגודל N מתבצעת הלולאה N^2 פעמים ללא תלות בערכים ההתחלתיים של איברי המערך.

♦ האלגוריתם ל**מיון הכנסה** פועל בצורה דומה להכנסת קלף בצורה ממוינת ליד האוחזת בקלפים. גם כאן במקרה הגרוע מיון הכנסה דורש בערך N^2 ביצועים של גוף הלולאה.

♦ **מיון בועות** פועל על עיקרון של החלפת זוגות. בכל שלב "מבעבעים" את האיבר הגדול (או הקטן) למיקומו המתאים במערך.

♦ **מיזוג** הוא פעולה למיזוג של שני מערכים ממוינים במערך אחד ממוין. בכל שלב מכניסים למערך הממוזג את האיבר הקטן ביותר מבין האיברים שבמערכי הקלט. בסוף התהליך יכול להישאר "זנב" באחד ממערכי הקלט, יש להעתיק אותו למערך הממוזג.

סיכום מרכיבי שפת C# שנלמדו בפרק 12

♦ **הצהרה על מערך** דו-ממדי בשפת C# נכתבת כך:

```
טיפוס [,] שם המערך;
```

ההצהרה מורכבת משם הטיפוס, אחריו **זוג** של סוגריים מרובעים המופרד בפסיק (המציין הצהרה על מערך דו-ממדי) ולבסוף שם המערך.

♦ לפני שניתן להשתמש במערך יש לבצע עבורו **הקצאת זיכרון**. הקצאת הזיכרון מתבצעת באמצעות ההוראה **new**:

```
שם המערך = new טיפוס [מספר שורות,מספר עמודות];
```

ניתן לשלב את ההצהרה ואת ההקצאה בהוראה אחת:

```
טיפוס [,]  שם המערך = new טיפוס [מספר שורות,מספר עמודות];
```

♦ **מספור האיברים במערך** דו-ממדי מתחיל בשורה 0 ובעמודה 0, לכן `mat[0,1]` מפנה לאיבר **השני** בשורה **הראשונה** במטריצה ששמה mat.

♦ לכל מערך דו-ממדי מוגדרת **הפעולה** **`GetLength(i)`**, המחזירה את מספר האיברים במימד המבוקש. למשל, `mat.GetLength(0)` מציין את מספר השורות במטריצה `mat`. ואילו `mat.GetLength(1)`, מציינת את מספר האיברים בשורה (את מספר העמודות).

♦ יש להקפיד על פנייה לאיבר באמצעות מציין שערכו איננו חורג מתחום הערכים המותר, כלומר, מציין של שורה נע בין 0 לבין מספר השורות פחות 1 ומציין של איבר בשורה נע בין 0 לבין מספר האיברים בשורה פחות 1. **חריגה מגבולות המערך** תגרום לשגיאה בזמן ריצה.

שאלות נוספות

שאלות נוספות לסעיף 12.2

1. נתון מערך A בגודל N, אשר ערכיו הראשונים הם 0 והערכים שאחריהם הם 1. כתבו קטע תוכנית המוצא את מציין האיבר האחרון שערכו הוא 0 (ניתן להניח שיש לפחות 0 אחד).

2. כתבו תוכנית המממשת את המשחק "חם קר". שחקן המשחק נגד התוכנית בוחר מספר שלם חיובי בין 0 ל-N נתון. על התוכנית לגלות את המספר הנבחר במינימום ניחושים. ניחוש הינו מספר שלם בין 0 ל-N ותשובת השחקן לניחוש היא: "חם יותר" אם המרחק (**בערך מוחלט**) בין הניחוש הנוכחי ל-K גדול יותר מן המרחק בין הניחוש הקודם ל-K, "קר יותר" אם ההיפך, ו-"אין שינוי" אם המרחק לא השתנה.

3. נתון מערך ממוין A בגודל N ובו ערכים חוזרים. כתבו קטע תוכנית שהפלט שלו הוא הודעה אם יש ערך המופיע יותר מ-N/2 פעמים.

4. נתון מערך ממוין ובו מספרים שלמים בתחום 100..200 בלבד, כל ערך יכול להופיע יותר מפעם אחת. עליכם לכתוב קטע תוכנית בשפת C# שיציג על המסך אילו מבין הערכים בין 100 ל-200 **לא** נמצאים במערך.

5. נתון מערך A בגודל N והוא "ממוין מעגלית"; כלומר מאיבר מסוים בו שאינו בהכרח האיבר הראשון, הוא ממוין בסדר עולה כאשר מתבוננים בו בצורה מעגלית (למשל המערך A שאיבריו הם: 5 3 2 1 9 7). כתבו קטע תוכנית המוצא את מציין האיבר הראשון שה"מיון המעגלי" מתחיל ממנו.

6. נתון מערך ממוין A בגודל N והוא מורכב משתי סדרות עולות, כך שהשנייה המתחילה עם סיום הראשונה היא קצרה יותר, ומהווה רישה של הראשונה (כלומר היא זהה לתחילת הראשונה). כתבו קטע תוכנית המוצא את מציין האיבר הראשון בסדרה השנייה.

שאלות נוספות לסעיף 12.3

1. נתון מערך (לא ממוין) A בגודל N ובו ערכים שונים זה מזה. כתבו קטע תוכנית המסדר את המערך בצורת "זיג-זג", כלומר איברי המערך A יסודרו כך שעבור כל ערך שנמצא במקום זוגי במערך – הערך שלפניו קטן ממנו והערך שאחריו גם קטן ממנו.

   הדרכה: בצעו מיון **בחירה** שבו תבחרו לסירוגין את המספר הקטן ביותר ולאחר מכן את המספר הגדול ביותר.

2. נתון מערך לא ממוין A בגודל N זוגי. כתבו קטע תוכנית שהפלט שלו הוא חלוקה של N המספרים לזוגות כך שהסכום המרבי מבין סכומי הזוגות הוא מינימלי.

3. נתון מערך (לא ממוין) ובו N ערכים. כתבו קטע תוכנית שהפלט שלו הוא הודעה אם סדרת ההפרשים של ערכי הרשימה הנתונה היא סדרה יורדת. (סדרת הפרשים היא הסדרה הנוצרת מההפרש בין כל זוג ערכים סמוכים בסדרה המקורית).

4. מיון נקרא "יציב" אם הסדר היחסי בין ערכים **זהים** במערך המקורי הוא גם הסדר ביניהם במערך הממוין. לדוגמה, אם במערך המקורי הערך 4 מופיע פעמיים - (למשל במקום השני ובמקום השמיני) ולאחר המיון, ה-4 הראשון מופיע לפני ה-4 השני המיון ייקרא מיון "יציב". לעומת זאת אם במערך הממוין, הסדר ביניהם התהפך, כלומר ה-4 השני מופיע לפני ה-4 הראשון אז המיון "אינו יציב".

ציינו והסבירו: האם מיון בחירה הוא יציב? האם מיון הכנסה הוא יציב? האם מיון בועות הוא יציב?

5. נתון מערך באורך N, וערך כל אחד מאיבריו הוא 0 או 1. כתבו קטע תוכנית שהפלט שלו הוא מספר ההחלפות המינימלי הדרוש כך שכל ה-1 יהיו מימין לכל ה-0. ההחלפה אינה חייבת להיעשות בין ערכים סמוכים.

שאלות נוספות לסעיף 12.4

1. הניחו שאין ערכים זהים בתוך מערך A ושאין ערכים זהים בתוך מערך B, אך ייתכן שיש ערכים הקיימים גם ב-A וגם ב-B. שנו את פעולת **המיזוג** כך שערכים זהים יופיעו פעם אחת בלבד במערך הממוזג.

2. נתונים שני מערכים ממוינים ובהם מספרים שונים. כתבו קטע תוכנית שיבדוק אם בכל עשיריית מספרים עוקבים במערך **הממוזג** קיים לפחות נציג אחד מכל אחד ממערכי המקור. קטע התוכנית יציג הודעה מתאימה.

3. נתונים שני מערכים ממוינים. כתבו קטע תוכנית המייצר את המערך **הממוזג** הארוך ביותר האפשרי המתחיל במספר הקטן ביותר (מבין שני המערכים), וכל מספר נוסף במערך הממוזג יהיה גדול לפחות ב-10 מקודמו.

4. נתונים שני מערכים ממוינים ובהם יש ערכים חוזרים. כתבו קטע תוכנית המוצא ומדפיס את הערך המופיע מספר מרבי של פעמים בשתי הרשימות יחדיו (אם יש יותר מאחד כזה, אזי נדפיס את אחד מן הערכים האלה).

5. נתונות שתי רשימות ממוינות. כתבו קטע תוכנית שהפלט שלו הוא הרשימה הממוזגת **הארוכה ביותר** האפשרית המתחילה במספר הקטן ביותר מבין שתי הרשימות ומקיימת את החוק הבא: כל מספר נוסף ברשימה הוא מעשיריית המספרים העוקבת לעשיריית המספרים של קודמו. למשל, אם המספר הראשון הוא 17 אזי המספר הבא אחריו יכול להיות בין 20 ל-29.

6. נתונים שלושה מערכים ממוינים. בתוך כל מערך יש ערכים השונים זה מזה. כתבו קטע תוכנית שהפלט שלו הוא רשימה ממוינת המתקבלת ממיזוג המערכים הנתונים, ללא חזרה של ערכים.

פרק 13 – פתרון בעיות

בפרק 11 התעמקנו בנושא מחלקות ועצמים ובפרק 12 התמקדנו במבנה נתונים מסוג מערך דו-ממדי ובפיתוח אלגוריתמים בסיסיים לחיפוש, למיון ולמיזוג. במהלך פתרון הבעיות התנסינו הן בהגדרת מחלקות, תכונות ופעולות והן בפיתוח ויישום של אלגוריתמים. בפרק זה נעסוק בבעיות המשלבות פיתוח אלגוריתמי יחד עם הגדרת מחלקות מתאימות לפתרון הבעיה. במהלך פתרון הבעיות נתמקד בתכונות ובפעולות הנדרשות ונתאים לכל בעיה את האלגוריתמים המתאימים לפתרונה.

בפרק זה תמצאו בעיות הלקוחות מפרק ג של בחינת הבגרות, ובעיות חדשות ברוח הפרק. בפרק זה בבגרות על הנבחן לנתח בעיה אלגוריתמית ולממש אותה באמצעות מחלקה מתאימה.

בעיה 1 מבגרות 2004 תשס"ד, שאלה 10

מטרת הבעיה ופתרונה: הצגת חיפוש במערך דו-ממדי.

נתון מערך דו-ממדי בגודל NxN המכיל מספרים שלמים מ-1 עד MAX_NUM (כולל).
נגדיר כי במערך קיימת "רביעייה k", אם המספר k מופיע ב-4 תאים של תת-מערך בגודל 2x2.
לדוגמה, לפניכם מערך דו-ממדי בגודל 5x5. במערך יש "רביעייה 9":

1	3	2	2	8
2	9	9	6	1
12	9	9	1	4
17	6	8	5	2
1	3	5	7	1

הגדירו מחלקה המכילה מערך דו-ממדי כתכונה, וכללו בה את הפעולות הבאות:

1. פעולה למציאת "רביעייה k". הפעולה תקבל את המספר k ותחזיר `true` אם קיימת "רביעייה k" במערך ו-`false` אחרת.
2. פעולה להחזרת המספר k הגדול ביותר שעבורו קיימת "רביעייה k" במערך. אם לא נמצאה רביעייה k במערך, תחזיר הפעולה 1-. למימוש פעולה זו יש להשתמש בפעולה שכתבתם בסעיף הקודם.

בבעיה זו נדרשנו להגדיר מחלקה המכילה מערך דו-ממדי כתכונה ושתי פעולות הפועלות על המערך. בשלב ראשון, נגדיר בצורה מדויקת כיצד נייצג את התכונות וכיצד נממש כל פעולה. לאחר מכן, נממש את המחלקה בשפת #C.

הגדרת המחלקה *רביעייהK*

הגדרת התכונות

- **matrix** – מערך דו-ממדי של מספרים מטיפוס שלם.
- **MAX_NUM** – שלם קבוע, המייצג את הערך המקסימלי שיכול להופיע במערך.

הגדרת הפעולות

נפרט את כל פעולות המחלקה ונסביר את הפעולות הדורשות התייחסות מיוחדת:

- ♦ **הפעולה הבונה** – פעולה זו תקבל מערך דו-ממדי המשמש לאתחול matrix.
- ♦ **מצא רביעייה k** – פעולה זו תקבל מספר k ותחזיר `true` אם קיימת עבורו רביעייה במערך ו-`false` אחרת. כדי לממש פעולה זו עלינו לבדוק עבור כל תא במערך, אם הוא ושלושת שכניו (השכן מימינו, השכן מתחתיו וכן השכן האלכסוני בכיוון למטה וימינה) זהים למספר k. לשם כך, נשתמש בלולאה מקוננת, ועבור כל תא במערך שערכו שווה ל-k נבדוק אם שכניו גם שווים ל-k (פרט לתאים בשורה האחרונה ובעמודה האחרונה):

```
if(matrix[i,j] == k)
{
    if ((matrix[i+1,j] == k) && (matrix[i,j+1] == k) &&
        (matrix[i+1,j+1] == k))
     {
     // k מצאנו רביעייה
     }
}
```

שימו ♥: אנו לא בודקים את התאים בשורה האחרונה ובעמודה האחרונה, כדי להימנע מגלישה מגבולות המערך.

- ♦ **מצא k מקסימלי** – פעולה זו תחזיר את המספר המקסימלי שקיימת עבורו "רביעייה k". יישום פעולה זו יתבצע באופן הבא: כיוון שאנו יודעים כי איברי המערך הינם מספרים שלמים בין 1 ל-MAX_NUM. אנו יכולים לבדוק את כל המספרים באמצעות שימוש בפעולה **מצא רביעייה k**, החל מ-MAX_NUM בסדר יורד עד 1. ברגע שנמצא מספר שקיימת עבורו "רביעייה k" נוכל לסיים את החיפוש כיוון שבידינו המספר הגדול ביותר העונה על הדרישה.

שימו ♥: האלגוריתם שהוצג לעיל מתאים למקרים ש-MAX_NUM הוא מספר קטן יחסית, מדוע? חשבו על אלגוריתם חלופי למקרה ש-MAX_NUM הוא מספר גדול.

מימוש המחלקה

```
/*
     k המחלקה רביעייה
*/
public class QuadrupletK
{
      // הגדרת התכונות
      private int[,] matrix;  // המערך
      private const int MAX_NUM = 30;
      // פעולה בונה
      public QuadrupletK(int[,] mat)
      {
            matrix = mat;
      } // QuadrupletK
      // k בדיקה אם קיימת במערך רביעייה
      public bool Find4K(int k)
      {
            int i,j;
            for (i = 0; i <  matrix.GetLength(0) - 1; i++)
                  for (j = 0; j < matrix.GetLength(1) - 1; j++)
```

```
                    if (matrix[i,j] == k)
                         if ( (matrix[i+1,j] == k) &&
                              (matrix[i,j+1] == k) &&
                              (matrix[i+1,j+1] == k) )
                              return true;
          return false;
     } // Find4K
     //  מציאת הרביעייה הגדולה ביותר במטריצה
     public int FindBiggestK()
     {
          int num = MAX_NUM;
          while (num > 0)
          {
               if (Find4K(num))
                    return num;
               else
                    num--;
          } // while
          return -1;
     }//FindBiggestK
}// class QuadrupletK
```

סוף פתרון בעיה 1

שימו ♥: בפעולה הבונה של המחלקה QuadrupletK אתחלנו את התכונה matrix באמצעות ההשמה matrix = mat; ולא באמצעות יצירת מערך חדש ובהעתקת האיברים אחד אחד. כתוצאה מהשמה זו התכונה matrix והמשתנה שהועבר כפרמטר מפנים לאותו מערך דו-ממדי. כעת, כל שינוי שיתבצע במערך דרך המשתנה שנמצא מחוץ לפעולה ישתקף גם במערך matrix. הדבר אינו רצוי במקרים שחשוב להגן על נתוני התכונה (על המערך matrix) מפני שינוי לא מבוקר מבחוץ. במקרה זה העדפנו לוותר על ההגנה ולבצע השמה פשוטה החוסכת הן מקום בזיכרון והן זמן בפעולת ההעתקה.

שאלה 13.1 בהשראת בגרות 2003 תשס״ג, שאלה 9

בהינתן מערך דו-ממדי המכיל מספרים שלמים שונים זה מזה ובהינתן מספר b, נגדיר ״תת-מערך b״ כתת-המערך המתקבל מהתחום שמימין ומתחת למספר b במערך כולל המספר b.

<u>לדוגמה</u>: עבור המערך הנתון ועבור המספר b=4:

2	7	12	3	17
27	22	4	0	1
9	-2	8	13	-9
-1	5	-20	20	10

ה״תת-מערך 4״ הוא:

4	0	1
8	13	-9
-20	20	10

א. הגדירו מחלקה בשם "תת-מערך מספרי" המכילה מערך דו-ממדי כתכונה (ובו מספרים שונים זה מזה), ופעולה בונה המקבלת מערך דו-ממדי המשמש לאתחול התכונה.

ב. בהינתן מערך דו-ממדי כפי שתואר לעיל, פתחו אלגוריתם אשר ימצא את ה"תת-מערך k" עבור k נתון. אם האלגוריתם מצא את ה"תת-מערך k" המבוקש הוא יציג את איבריו, אחרת תוצג הודעה ש"תת-מערך k" לא נמצא.

ג. כתבו פעולה במחלקה "תת-מערך מספרי" המממשת את האלגוריתם מהסעיף הקודם. הפעולה מקבלת כפרמטר מספר כלשהו k ומציגה את ה"תת-מערך k" שנמצא או הודעה ש"תת-מערך k" לא נמצא.
הדרכה: לצורך מימוש הפעולה, היעזרו בפעולת עזר פרטית המקבלת מציינים של איבר במערך הדו-ממדי ומציגה את איברי התת-מערך שנמצאים מימין ומתחת לאיבר המצוין, כולל האיבר עצמו.

ד. הגדירו במחלקה פעולה נוספת המציגה את ה"תת-מערך k" עבור k שהוא המספר המינימלי במערך הדו-ממדי. הפעולה לא מקבלת פרמטרים, היא מוצאת את האיבר המינימלי במערך הדו-ממדי ולאחר מכן נעזרת בפעולה הפרטית מהסעיף הקודם לצורך הצגת איברי התת-מערך.

שאלה 13.2 בהשראת בגרות 2005 תשס"ה, שאלה 9

נגדיר "פרח" במערך באופן הבא: הפרח מורכב מ-5 איברים בתת-מערך בגודל 3X3. האיבר המרכזי בתת-מערך הוא "לב הפרח". ארבעת האיברים הצמודים לו בפינותיו הם "עלי הכותרת" של ה"פרח". הערך של "לב הפרח" שווה לסכום ערכי "עלי הכותרת" של ה"פרח" ובכל "פרח" חייבים להיות בדיוק 4 "עלי כותרת".

<u>דוגמה</u>: במערך בגודל 5x4 שלפניכם יש "פרח" אחד:

0	0	3	3
2	0	2	1
0	0	2	3
1	4	8	11
0	-2	9	7

"עלה כותרת"
"לב הפרח"

א. כתבו מחלקת "שדה פרחים" המכילה מערך דו-ממדי המייצג שדה. הפעולה הבונה מקבלת מערך דו-ממדי, ומאתחלת בו את השדה.

ב. כתבו פעולה המקבלת שני מספרים שלמים המציינים מיקום של איבר במערך (אינדקסים), המספר הראשון מציין שורה, והמספר השני מציין עמודה. הפעולה תחזיר **true** אם איבר זה הוא "לב הפרח" של "פרח" במערך; אחרת – הפעולה תחזיר **false**.

ג. מערך הוא "פרחוני" אם יש בו לפחות 5 "פרחים".
כתבו פעולה שתבדוק אם המערך הוא "פרחוני", ותחזיר **true** או **false** בהתאם. השתמשו בפעולה שכתבתם בסעיף ב. שימו לב, הפרחים יכולים להיות חופפים זה לזה, כלומר מותר לפרח אחד להכיל לב או עלי כותרת השייכים לפרח אחר.

בעיה 2 בהשראת בגרות 2004 תשס"ד, שאלה 9

מטרת הבעיה ופתרונה: הצגת בעיה מורכבת שלפתרונה יש להשתמש במערך של עצמים, במערך כתכונה של עצם ובתבנית מיון.

> בסוכנות הנסיעות "שלום שלום" מארגנים טיולים. הסוכנות מציעה 100 טיולים, ולכל אחד מהם מספר בין 1 ל-100. לכל טיול יכולים להירשם עד 50 נוסעים. נוסע יכול להירשם לטיול רק אם יש מקום בטיול.
>
> פתחו וממשו אלגוריתם לניהול סוכנות הנסיעות. האלגוריתם ירשום אנשים על פי שמם לטיולים שאליהם הם מבקשים להירשם, עד אשר יתקבל השם End לסיום הקלט. האלגוריתם יציג את רשימת הטיולים ממוינת לפי מספר האנשים בכל טיול, ויוסיף בצד כל טיול את רשימת שמות האנשים המשתתפים בו.

בבעיה זו אנו נדרשים לרשום אנשים לטיול על בסיס מקום פנוי, ולהציג את כל הטיולים ממוינים לפי מספר הנרשמים אליהם. אם כך, בעיה זו עוסקת באוסף טיולים ומתאים לייצג אותו כמערך. מידע על טיול כולל נתונים שונים (כגון מספר המשתתפים ורשימת המשתתפים). על כל טיול ניתן לבצע מספר פעולות שונות (כגון רישום משתתף). לכן מתאים להגדיר מחלקה בשם טיול.

הגדרת המחלקה *טיול*

הגדרת התכונות

נבחן מהן הדרישות עבור עצם מהמחלקה טיול: יש לשמור את מספרו הסידורי של הטיול, את מספר המשתתפים בו ואת רשימת שמות האנשים הרשומים אליו. אם כך, תכונות של עצם מסוג "טיול" הן:

- **tripNum** – מספר שלם המייצג את מספרו הסידורי של הטיול.
- **numOfPassengers** – מספר שלם המייצג את מספר הנרשמים לטיול.
- **passengers** – מערך של מחרוזות המייצג את רשימת שמות הנרשמים לטיול.
- **MAX_PASS_PER_TRIP** – קבוע שלם שערכו 50 והוא מייצג את מספר הנוסעים המקסימלי בטיול.

הגדרת הפעולות

כעת נבחן מהן הפעולות שניתן לבצע על טיול: הוספת משתתף (אם יש מקום פנוי), והצגת רשימת שמות המשתתפים. אם כך פעולות של עצם מסוג "טיול" הן:

- **הפעולה הבונה** – פעולה זו תקבל כפרמטר את מספרו הסידורי של הטיול (ערך שלם), ותאתחל את התכונה tripNum. בנוסף, הפעולה הבונה תקצה את המערך של שמות המשתתפים ותאפס את התכונה של מספר המשתתפים.
- **הוספת מטייל** – פעולה זו תקבל את שם המטייל כפרמטר. אם יש מקום פנוי בטיול, הוא יצורף לרשימת המשתתפים, מספר המשתתפים יקודם ב-1 ויוחזר הערך `true`. אם אין מקום פנוי יוחזר הערך `false`.
- **החזרת רשימת המטיילים** – פעולה זו תחזיר את רשימת המטיילים לצורך תצוגה.

- ♦ **החזרת מספרו הסידורי של הטיול** – פעולה זו תחזיר את מספרו הסידורי של הטיול לצורך תצוגה.
- ♦ **החזרת מספר המשתתפים בטיול** – פעולה זו תחזיר את מספר המשתתפים בטיול לצורך מיון הטיולים על פי מספר המשתתפים.

מימוש המחלקה

```
/*
        מחלקת טיול
*/
public class Trip
{
      // הגדרת התכונות
      private const int MAX_PASS_PER_TRIP = 50; // קבוע: מספר נוסעים
      private int tripNum;           // מספר טיול
      private int numOfPassengers;  // מספר נרשמים
      private string[] passengers;  // שמות הנרשמים לטיול
      // פעולה בונה
      public Trip(int tripNum)
      {
            passengers = new string[MAX_PASS_PER_TRIP];
            this.tripNum = tripNum;
            numOfPassengers = 0;
      }
      // פעולות גישה
      public int GetTripNum()
      {
            return tripNum;
      }
      public int GetNumOfPassengers()
      {
            return numOfPassengers;
      }
      // הפעולה מחזירה את שמות המשתתפים בטיול
      public string[] GetPassengers()
      {
            string[] pass = new string[numOfPassengers];
            for(int i = 0; i < numOfPassengers; i++)
                  pass[i] = passengers[i];
            return pass;
      }
      // הפעולה מוסיפה נוסע לטיול
      // ומחזירה ערך אמת אם יש מקום בטיול ושקר אחרת
      public bool AddPassenger(string passenger)
      {
            if (numOfPassengers < MAX_PASS_PER_TRIP)
            {
                  passengers[numOfPassengers] = passenger;
                  numOfPassengers++;
                  return true;
            } // if
            return false;
```

```
	} // AddPassenger
}// class Trip
```

שימו ♥ הפעולה `GetPassengers` יוצרת מערך חדש בשם `pass` וגודלו כמספר המשתתפים בטיול. הפעולה מעתיקה את רשימת המשתתפים בטיול לתוך המערך ומחזירה את המערך החדש. כלומר הפעולה מחזירה **עותק** של המערך ולא את המערך עצמו. חשבו, מה היה יכול לקרות אילו החזרנו פשוט את המערך `passangers` באופן הבא:

```
return passangers;
```

הגדרת הפעולה הראשית

פירוק לתת-משימות

משימות הפעולה הראשית הן:

4. יצירת מערך של 100 טיולים ואתחול הטיולים במערך.
5. קליטת שמות אנשים, ועבור כל אחד קליטת מספר הטיול שהוא מעוניין להירשם אליו ורישום לטיול זה.
6. מיון הטיולים לפי מספר המשתתפים. המיון שנשתמש בו הוא "מיון בועות".
7. הצגת רשימת הטיולים הממוינת.

בחירת משתנים

בפעולה הראשית יוגדר מערך של 100 טיולים, כלומר של עצמים מהמחלקה טיול.

מימוש הפעולה הראשית

```
/*
              המחלקה הראשית
*/
using System;
public class TravelAgency
{
	public static void Main()
	{
		//  הגדרת והקצאת משתנים
		const int TRIPS_NUM = 100;   // קבוע: מספר טיולים
		Trip[] trips = new Trip[TRIPS_NUM]; // הקצאת מערך הטיולים
		string passengerName;       // שם נוסע
		int tripNum;                // מספר טיול מבוקש
		//  הקצאת הטיולים
		for (int i = 0; i < TRIPS_NUM; i++)
			trips[i] = new Trip(i + 1);
		Console.Write("Enter passenger name, type 'End' to " +
		                                        "finish: ");
		passengerName = Console.ReadLine();
		while (passengerName!="End")
		{
			Console.Write("Enter the trip number: ");
			tripNum = int.Parse(Console.ReadLine());
			if (trips[tripNum-1].AddPassenger(passengerName))
				Console.WriteLine("You were added " +
				                               "successfully");
			else
```

```
                    Console.WriteLine("This trip is full");
                Console.Write("Enter passenger name, type 'End' " +
                                                "to finish: ");
                passengerName = Console.ReadLine();
            } // while
            // מיון הטיולים לפי כמות הנרשמים
            Trip temp;
            for (int i = 1; i < TRIPS_NUM; i++)
            {
                for (int j = 0; j < TRIPS_NUM - i; j++)
                {
                    if (trips[j].GetNumOfPassengers() >
                                trips[j+1].GetNumOfPassengers())
                    {
                        // החלפה
                        temp = trips[j];
                        trips[j] = trips[j+1];
                        trips[j+1] = temp;
                    } // if
                } // for j
            } // for i
            string[] passengers;
            //  הצגת הטיולים והמשתתפים בהם, לפי מספר המשתתפים
            for (int i = 0; i < TRIPS_NUM ; i++)
            {
                Console.WriteLine("Trip {0} has {1} passengers",
                                trips[i].GetTripNum(),
                                trips[i].GetNumOfPassengers());
                passengers = trips[i].GetPassengers();
                for (int j = 0; j < passengers.Length; j++)
                {
                    Console.WriteLine(passengers[j]);
                }
            } // for
        } // main
    } // class TravelAgency
```

סוף פתרון בעיה 2

שאלה 13.3

באולימפיאדת הארנבים מתקיימת תחרות בשלוש קטגוריות שונות: ריצה לגזר, ריצה לכרוב וריצה לברוקולי. באולימפיאדה מחולקות שתי מדליות זהב. את הראשונה מקבל הארנב שממוצע התוצאות שלו בשלוש התחרויות הוא הטוב ביותר (הנמוך ביותר). את השנייה מקבל הארנב שזמן הריצה שלו הוא הנמוך ביותר מבין כל זמני הריצה של כל הארנבים בכל הקטגוריות כולן.

פתחו וממשו אלגוריתם העוזר למארגני התחרות לבחור את המנצחים. האלגוריתם יקלוט תחילה את מספר הארנבים המתחרים, לאחר מכן יקלוט רשימה של שמות הארנבים ואת זמני הריצה לכל ארנב בכל קטגוריה. פלט האלגוריתם יהיה שמות שני הזוכים במדליות הזהב.

הדרכה: בדומה לבעיה 2 גם בבעיה זו יש לשמור אוסף, כלומר הפעולה הראשית תכיל מערך של עצמים. בבעיה זו כל עצם הוא ממחלקת "ארנב". תכונות ה"ארנב" הן: שם הארנב ומערך המכיל את שלוש התוצאות שלו. הפעולות על עצם מסוג ארנב יאפשרו לקבל את ממוצע התוצאות של

הארנב וכן את תוצאת הריצה הטובה ביותר שלו. הפעולה הראשית תקלוט את נתוני הארנבים ותשמור אותם במערך ארנבים. היא תחפש את הארנב שיש לו התוצאה הממוצעת הנמוכה ביותר ואת הארנב שיש לו התוצאה הנמוכה ביותר, ותציג את שמם.

בעיה 3

מטרת הבעיה ופתרונה: הצגת שילוב תבניות

נצחיה המורה להיסטוריה מעוניינת לדעת כמה תלמידים קיבלו את הציון הגבוה ביותר (שאינו בהכרח 100) במבחן השכבתי. צחי הכין עבור נצחיה את קטע התוכנית הבא:

```
int maxCount=0;
int max=0;
int numOfStudents;
int grade;
Console.Write("Enter number of students: ");
numOfStudents = int.Parse(Console.ReadLine());
for (int i = 0; i < numOfStudents; i++)
{
      grade = int.Parse(Console.ReadLine());
      if (grade > max)
          max = grade;
      if (grade == max)
          maxCount++;
}
Console.WriteLine("{0} students received the maximum grade",
                                                    maxCount);
```

א. האם קטע התוכנית משיג את מטרתו?

ב. אם קטע התוכנית שגוי הציעו תיקון כך שהוא ישיג את מטרתו עבור כל קלט אפשרי.

בדיקת קטע התוכנית

בבעיה זו אנו נדרשים לנתח קטע תוכנית ולבחון אם הוא משיג את מטרתו. לשם כך, עלינו לבחון דוגמאות קלט שונות ולבדוק אם פלט התוכנית תקין. לשם בדיקת התוכנית ניתן להסתפק בדוגמאות שבהן 10 תלמידים בלבד.

שאלה 13.4

הריצו את התוכנית ובדקו עבור דוגמאות הקלט הבאות אם הפלט תקין (משמאל לימין המספר הראשון בקלט מייצג את מספר התלמידים):

א. 10 90 55 84 90 74 89 80 84 90 56

ב. 10 84 71 84 90 53 84 76 90 85 67

בבדיקה זו עבור דוגמת קלט א, נקבל פלט תקין המודיע כי שלושה תלמידים קיבלו את הציון המקסימלי. לעומת זאת, עבור דוגמת קלט ב נקבל פלט שגוי המודיע כי ארבעה תלמידים קיבלו את הציון המקסימלי ולא שניים כפי שניתן לראות מהקלט.

מציאת השגיאה ותיקונה

כעת כדי לתקן את השגיאה עלינו להבין מדוע שגיאה זו התרחשה. כפי שניתן לראות בדוגמת הקלט הראשונה, הציון המקסימלי מתקבל ראשון. המשתנה max שתפקידו לשמור את הציון

הגבוה ביותר מקבל את הערך 90. המשתנה maxCount שתפקידו למנות כמה פעמים מתקבל הציון הגבוה ביותר מקבל את הערך 1, כיוון שעד כה הציון 90 התקבל פעם אחת. מרגע זה ואילך, כל הציונים שמתקבלים, קטנים או שווים ל-max. במקרה שציון קטן מ-max, לא מתבצע דבר. במקרה שציון שווה ל-max, המשתנה maxCount מתקדם ב-1. כך התוכנית ממשיכה עד לסיום הקטע בהצלחה.

לעומת זאת, בדוגמת הקלט השנייה הציון המקסימלי לא מתקבל הראשון. תחילה מתקבל הציון 84 אשר נמנה פעמיים כציון הגבוה ביותר, ורק לאחר מכן מתקבל הציון 90. ברגע שמתקבל הציון 90 המשתנה max מתעדכן ל-90 (הגבוה מ-84), והמשתנה maxCount מתקדם כי התקבל ציון מקסימלי אחד נוסף. כאן נמצא שורש הבעיה, אף על פי שהמשתנה max עודכן למקסימום החדש, המשתנה maxCount לא אופס, וכך נמשיך למנות באופן שגוי בכל פעם החל מהערך שנמצא ב-maxCount.

המסקנה המתבקשת היא, שעלינו לאפס את המונה maxCount ברגע שמעדכנים את המשתנה max. נעשה זאת ב-**if** הראשון, שבודקים בו אם התקבל ציון גבוה יותר מהציון שנשמר עד כה. אם אכן התקבל ציון גבוה יותר, יש לאפס את המונה שסופר כמה פעמים התקבל ציון זה. אם כך, קטע התוכנית ייראה כך לאחר התיקון:

```
int maxCount=0;
int max=0;
int numOfStudents;
int grade;
Console.Write("Enter number of students: ");
numOfStudents = int.Parse(Console.ReadLine());
for (int i = 0; i < numOfStudents; i++)
{
      grade = int.Parse(Console.ReadLine());
      if (grade > max)
      {
            max = grade;
            maxCount = 0;  //  איפוס המונה maxCount
      }
      if (grade == max)
            maxCount++;
}
Console.WriteLine("{0} students received the maximum grade",
                                                      maxCount);
```

סוף פתרון בעיה 3

בבעיה 3 נעזרנו בשתי תבניות מוכרות: תבנית מקסימום ותבנית מנייה. התבניות שזורות זו בזו כך שברגע שמשתנה המקסימום מוחלף, יש לאתחל מחדש את המונה. לפעמים משולבות יחדיו תבניות שונות בקטע קוד אחד. במקרים כאלה יש להקפיד על שילוב נכון של חלקי התבניות.

שימו ♥: כאשר משלבים כמה תבניות בפתרון בעיה, עלולים להשמיט חלק מההוראות הנדרשות או לכתוב אותן שלא במקומן. יש להקפיד ולבדוק היטב שהרכבת התבניות נעשתה בהצלחה, ולא נגרמו שגיאות לקוד עקב הרכבה לא מוצלחת.

שאלה 13.5

נצחיה, המורה להיסטוריה, מעוניינת לדעת את ממוצע כל אחת מ-6 הכיתות שלה במבחן המשווה, וכן את ממוצע השכבה. צחי שוב ניסה את מזלו בהכנת קטע קוד לפתרון הבעיה:

```
const int NUM_OF_CLASSES = 6;
int classSize;
int classSum = 0;
int totalSum = 0;
int totalNumOfStudents = 0;
int grade;
for (int i = 0; i < NUM_OF_CLASSES; i++)
{
      Console.Write("Enter number of students in class {0} ", (i+1));
      classSize = int.Parse(Console.ReadLine());
      totalNumOfStudents = totalNumOfStudents + classSize;
      for (int j = 0; j < classSize; j++)
      {
            grade = int.Parse(Console.ReadLine());
            classSum = classSum + grade;
      }
      Console.WriteLine( (double)classSum / classSize );
      totalSum = totalSum + classSum;
}
Console.WriteLine( (double)totalSum / totalNumOfStudents );
```

א. האם קטע התוכנית משיג את מטרתו?

ב. אם קטע התוכנית שגוי, הציעו תיקון כך שקטע התוכנית ישיג את מטרתו עבור כל קלט אפשרי.

ג. אילו תבניות שולבו בקטע התוכנית?

שאלה 13.6

במפעל מועסקים 100 עובדים. בעל המפעל מעוניין לדעת נתונים על המשכורות שעליו לשלם בסופו של חודש. כל עובד נמצא במפעל 25 ימים בחודש ועובד בכל יום מספר מסוים של שעות. התעריף לעובדים אינו קבוע אלא אישי ומשתנה מעובד לעובד.

פתחו אלגוריתם הקולט עבור כל אחד מ-100 העובדים את שמו, את התעריף שלו לשעה, וכמה שעות עבד בכל אחד מ-25 ימי העבודה באותו חודש. האלגוריתם יציג כפלט את סכום המשכורות שעל בעל המפעל לשלם לעובדיו. ממשו את האלגוריתם **בשתי הדרכים** הבאות:

א. ממשו את האלגוריתם כולו בפעולה הראשית, כך שלולאה חיצונית תקלוט נתונים לכל אחד מ-100 העובדים. בכל סיבוב של הלולאה ייקלט שם עובד ותעריף לשעה ולאחר מכן בלולאה פנימית, ייקלטו השעות שהעובד עבד בכל אחד מ-25 ימי העבודה בחודש. עליכם לצבור עבור כל עובד את מספר השעות שעבד, ולהכפילן בתעריף שלו. בנוסף עליכם לצבור את המשכורות של כל העובדים.

שימו ♥: זכרו לאפס את צובר השעות של העובד במקום המתאים.

ב. הגדירו מחלקת עובד. תכונות עובד כוללות: שם, תעריף לשעה וסכום השעות שהעובד עבד במשך החודש. למחלקה זו הגדירו פעולה בונה המקבלת את שם העובד ואת התעריף לשעה. בנוסף כתבו פעולה המקבלת את השעות שהעובד עבד באחד מהימים ומעדכנת את סכום השעות החודשי של העובד, ופעולה המחזירה את המשכורת החודשית שיש לשלם לעובד. בפעולה הראשית, ממשו לולאה אשר תייצר עצם מסוג עובד לכל אחד מ-100 העובדים,

ובלולאה פנימית תקלוט ותעדכן את מספר השעות שהעובד עבד בכל אחד מ-25 ימי העבודה. סכמו את המשכורות של כל 100 העובדים.

שימו ♥: היכן איפסתם את צובר השעות? האם בגרסה השנייה של פתרון הבעיה קיים החשש כי נשכח לאפס את הצובר?

בעיה 4

מטרת הבעיה ופתרונה: הצגת מקסימום כתכונה, הצגת מונה כתכונה והצגת מערך מונים כתכונה.

טורניר השש-בש מתחיל ועדיין לא נמצאו הקוביות! כתבו מחלקה המגדירה זוג קוביות שש-בש תקניות. בנוסף לערכי הקוביות, עצם מסוג זוג קוביות יכלול מידע סטטיסטי על זוג הקוביות. כדי לדמות את הטלת הקוביות ואת החזרת ערכי הקוביות, ממשו במחלקה את הפעולות:

```
public void ThrowDice()
public int GetDie1()
public int GetDie2()
```

בסיום המשחק, נוכל לקבל מעצם מסוג "זוג קוביות" את המידע הבא:

א. מה סכום ההטלה הגבוה ביותר שהתקבל בזוג הקוביות במשחק?

ב. כמה פעמים הוגרל דאבל בזוג הקוביות?

ג. מהי תוצאת הדאבל שהוגרלה הכי הרבה פעמים? אם הוגרלה יותר מאחת כזו תוחזר התוצאה שערכה גדול יותר.

הגדרת המחלקה *זוג קוביות*

בבעיה זו אנו נדרשים לדמות הטלת שתי קוביות ובמקביל לאסוף מידע סטטיסטי על ההטלות. לשם כך נגדיר את המחלקה "זוג קוביות". עצם מסוג זוג קוביות יאפשר לנו להטיל את הקוביות ולשמור מידע סטטיסטי הקשור להטלתן.

הגדרת התכונות

נבחן מהם הנתונים שעלינו לשמור כדי שנוכל לענות על כל אחת מהשאלות המבוקשות:

א. מה סכום ההטלה הגבוה ביותר שהתקבל בזוג הקוביות במשחק?

כדי לענות על שאלה זו נשתמש בתבנית *מקסימום* לחישוב הערך הגבוה ביותר שהתקבל עד רגע ההטלה. נגדיר משתנה מקסימום כתכונה של העצם, ולאחר כל הטלה הוא יעודכן לפי הצורך.

ב. כמה פעמים הוגרל דאבל בזוג הקוביות?

כדי לענות על שאלה זו נשתמש בתבנית *מנייה*. משתנה המנייה יישמר כתכונה, ולאחר כל הטלה יעודכן משתנה זה לפי הצורך.

ג. מהי תוצאת הדאבל שהוגרלה הכי הרבה פעמים?

לשאלה זו שש תוצאות אפשריות שונות. (דאבל-1 הוגרל הכי הרבה פעמים, דאבל-2 הוגרל הכי הרבה פעמים, ..., דאבל-6 הוגרל הכי הרבה פעמים). כדי לענות על שאלה זו עלינו למנות עבור כל אחת מ-6 התוצאות האפשריות כמה פעמים היא התקבלה, ולבדוק איזו תוצאה התקבלה הכי הרבה פעמים. לצורך כך נשתמש בתבנית *מערך-מונים*. מערך המונים יישמר כתכונה, ולאחר כל הטלה הוא יעודכן לפי הצורך.

אם כך, התכונות של עצם מסוג "זוג קוביות" הן:

- **die1** – מספר שלם המייצג את תוצאת ההטלה האחרונה של הקובייה הראשונה.
- **die2** – מספר שלם המייצג את תוצאת ההטלה האחרונה של הקובייה השנייה.
- **maxDiceSum** – מספר שלם המייצג את הסכום המקסימלי שהוגרל בהטלת הקוביות עד כה.
- **doubleCount** – מספר שלם המייצג את מספר הפעמים שהתקבל דאבל עד כה.
- **doubleCountArr** – מערך בגודל 6 מטיפוס שלם המייצג כמה פעמים הוגרל כל אחד מהדאבלים האפשריים.

הגדרת הפעולות

כעת נבחן מהן הפעולות שניתן לבצע על עצם מסוג "זוג קוביות".

- **הפעולה הבונה** – הפעולה תאפס את התכונות, תקצה מקום למערך מוני הדאבל ותאפס אותו.
- **הטל קוביות** – הפעולה תגריל שני מספרים אקראיים בין 1 ל-6 ותציב אותם בתכונות המתאימות.
- **החזרת תוצאת ההטלה של הקובייה הראשונה** – הפעולה תחזיר את תוצאת ההטלה של הקובייה הראשונה.
- **החזרת תוצאת ההטלה של הקובייה השנייה** – הפעולה תחזיר את תוצאת ההטלה של הקובייה השנייה.
- **החזרת סכום ההטלה הגבוה ביותר** – הפעולה תחזיר את הסכום המקסימלי שהתקבל בהטלת הקוביות.
- **החזרת מספר דאבלים** – הפעולה תחזיר את מספר הפעמים שהתקבל דאבל.
- **החזרת מספר דאבלים שהתקבל הכי הרבה פעמים** – הפעולה תסרוק את מערך המונים ותחזיר את המספר שהוגרל הכי הרבה פעמים כדאבל. אם הוגרלו כמה תוצאות כאלה, תוחזר הגדולה מביניהן. אם לא התקבלו דאבלים כלל יוחזר הערך 0.

מימוש המחלקה

```
/*  מחלקת זוג הקוביות  */
using System;
public class Dice
{
      // הגדרת התכונות
      private int die1; // תוצאת ההטלה של הקובייה הראשונה
      private int die2; // תוצאת ההטלה של הקובייה השנייה
      private int maxDiceSum;
      private int doubleCount;
      private int[] doubleCountArr; // מערך המונים עבור הדאבלים
      // פעולה בונה
      public Dice()
      {
            die1 = 0;
            die2 = 0;
            maxDiceSum = 0;
            doubleCount = 0;
```

```
        doubleCountArr = new int[6];
        for (int i = 0; i < 6; i++)
            doubleCountArr[i] = 0;
    }
    //פעולות גישה
    public int GetDie1()
    {
        return die1;
    }
    public int GetDie2()
    {
        return die2;
    }
    //  הפעולה מטילה את שתי הקוביות ומעדכנת את התכונות
    public void ThrowDice()
    {
        Random rnd = new Random();
        die1 = rnd.Next(1,7);
        die2 = rnd.Next(1,7);
        // עדכון הסכום המקסימלי שהתקבל מהטלת הקוביות
        if (die1 + die2 > maxDiceSum)
            maxDiceSum = die1 + die2;
        //  עדכון כמות הפעמים שהתקבל דאבל, ומערך המונים
        if (die1 == die2)
        {
            doubleCount++;
            doubleCountArr[die1-1]++;
        }
    }
    // החזרת סכום הטלת זוג הקוביות הגבוה ביותר שהיה במשחק
    public int GetMaxDiceSum()
    {
        return maxDiceSum;
    }
    // החזרת מספר הפעמים שהתוצאה היתה דאבל
    public int GetDoubleCount()
    {
        return doubleCount;
    }
    // הפעולה מחזירה את תוצאת הדאבל שהתקבלה הכי הרבה פעמים
    public int GetMaxTimesDouble()
    {
        int maxIndex = 0;
        if (doubleCount == 0)  // לא היה אף דאבל
            return 0;
        for (int i = 1; i < doubleCountArr.Length; i++)
            if (doubleCountArr[i] > doubleCountArr[maxIndex])
                maxIndex = i;
        return maxIndex + 1;
    }
}// class Dice
```

סוף פתרון בעיה 4

שאלה 13.7

משרד החינוך החליט לבדוק את הרגלי הקריאה של הנוער. הוחלט לבצע סקר, כך שכל משתתף בו ירשום את שמו, ויקליד 1 עבור כל ספר שקרא מרשימת 50 הספרים שהוצגו לפניו או 0 אם לא קרא את הספר. כתבו תוכנית שתקלוט את תשובותיהם של משתתפי הסקר. הקליטה תסתיים עם קבלת "@@@" כשם הקורא). על התוכנית להציג את התשובות לשאלות:

א. מספר המשתתפים שלא קראו אף אחד מהספרים שברשימה.

ב. מספר המשתתפים שקראו יותר ממחצית הספרים שברשימה.

ג. עבור כל אדם שקרא את כל הספרים יש להדפיס את שמו בצירוף הודעה "נוער למופת".

ד. מספר הספר הנפוץ ביותר (מספר הספר שקראו מספר הקוראים הגדול ביותר).

הדרכה: כתבו מחלקה המגדירה "סקר". הגדירו לעצם מסוג סקר את התכונות הבאות המייצגות את תוצאות הסקר: מספר המשתתפים שלא קראו אף ספר מהרשימה, מספר המשתתפים שקראו יותר ממחצית הספרים שברשימה ומערך מונים המונה עבור כל ספר את מספר המשתתפים שקראו אותו.

הגדירו במחלקה "סקר" את הפעולות הבאות:

פעולה בוליאנית המקבלת מערך בגודל 50 המייצג תשובות של קורא יחיד לסקר. הפעולה מעדכנת את התכונות המייצגות את תוצאות הסקר ומחזירה **true** אם התלמיד קרא את כל הספרים אחרת יוחזר **false**.

הוסיפו פעולות המחזירות את מספר המשתתפים שלא קראו אף ספר מהרשימה, את מספר המשתתפים שקראו יותר ממחצית הספרים שברשימה ואת מספר הספר הנפוץ ביותר.

שאלה 13.8

מנהלי קבוצת הכדורסל "ג'ירפות בע"מ" מעוניינים לקבל מידע על ביצועי עשרת השחקנים בקבוצה. בעת משחק, ברגע ששחקן קולע לסל, מקבלים כקלט את מספר השחקן (מספר בין 1 ל-10) ואת מספר הנקודות שקלע (מספר בין 1 ל-3). בסיום משחק מתקבל כקלט המספר 1- כמספר שחקן כדי לסיים את הקליטה. בסיום המשחק מנהלי הקבוצה מעוניינים לדעת: מה מספר השחקן שקלע את מספר הסלים המרבי? מה מספר השחקן שצבר את מספר הנקודות המרבי? מה מספר השחקן שקלע את מספר הסלים הנמוך ביותר? כתוב תוכנית בשפת C# אשר תספק למנהלי הקבוצה את המידע הדרוש. עליכם לבנות מחלקה המגדירה משחק כדורסל. עצם מסוג זה ירכז את נתוני השחקנים במשחק ויספק את המידע הדרוש. אם קיימת יותר מתשובה אחת (למשל, שני שחקנים צברו את מספר הנקודות המרבי), יוחזר מספר השחקן הוותיק יותר (שמספרו נמוך יותר).

הדרכה: הגדירו לעצם מסוג "משחק כדורסל" פעולה המקבלת נתונים של קליעה אחת: מספר שחקן הקולע ומספר הנקודות שקיבל. פעולה זו תעדכן את התכונות המתאימות אשר ישמרו את המידע הדרוש למנהלי הקבוצה.

שאלה 13.9 בהשראת בגרות 2006 תשס"ו, שאלה 10

בית ספר מזמין מחנות ספרים ספרי קריאה עבור 620 תלמידיו.

הספרים בחנות מסומנים בקודים. קוד יכול להיות מספר בין 1 ל-315. אם יש בחנות כמה עותקים מאותו ספר, הם מסומנים באותו קוד. כל תלמיד מזמין לפחות ספר אחד, ומחיר כל ספר לתלמיד הוא 28 שקל. בית הספר גובה מהתלמידים את התשלום בעבור הספרים שהזמינו, ומעביר את התשלום הכולל לחנות הספרים.

א. פתחו אלגוריתם שיקלוט את הזמנות התלמידים ויטפל בתשלומי התלמידים ובתשלום בית הספר, ממשו אותו בשפת C#. עליכם לפתח את האלגוריתם לפי השלבים שלפניכם:

i. הגדירו את מחלקת "הזמנות ספרים". עצם מסוג זה יכיל את סך כל ההזמנות שביצעו התלמידים עבור כל ספר וספר.

ii. הגדירו במחלקת "הזמנות ספרים" פעולה "הזמנה מתלמיד". פעולה זו תקבל מערך המכיל את קודי הספרים שהזמין תלמיד ותעדכן את ההזמנות. הפעולה תחזיר את הסכום שעל התלמיד לשלם עבור הזמנה זו.

iii. הגדירו במחלקת "הזמנות ספרים" פעולה "סכם הזמנות". פעולה זו תכין מחרוזת המפרטת עבור כל קוד של ספר את מספר העותקים שהוזמנו ממנו בסך הכול. כמו כן, יחושב ויפורט התשלום הכולל שעל בית הספר להעביר לחנות בעבור כל הספרים שהוזמנו.

ב. כתבו בשפת C# את הפעולה הראשית אשר תנהל את ההזמנות. בפעולה הראשית ייקלטו מספר הספרים שכל תלמיד מזמין; ואחריו ייקלטו הקודים של הספרים שהתלמיד מזמין. בסיום הזמנה של תלמיד יוצג כפלט הסכום שעליו לשלם בעבור הספרים שהזמין. בסיום קליטת הנתונים של כל התלמידים יוצג סיכום ההזמנות.

הערה: אין צורך לבדוק את תקינות הקלט.

בעיה 5

מטרת הבעיה ופתרונה: יחסים בין מערכים.

> נתונה סדרה של מספרים, עליכם לפתח אלגוריתם המפצל את הסדרה לשתי סדרות, האחת מכילה את המספרים הזוגיים והשנייה את האי-זוגיים. הגדירו מחלקה בשם EvenOdd, במחלקה יהיו שלושה מערכים: המערך השלם המכיל את כל המספרים ושני מערכים נוספים: אחד עבור המספרים הזוגיים והאחר עבור המספרים האי-זוגיים. בנוסף תהיה פעולה המפצלת את המערך השלם לשני מערכים – אחד של מספרים זוגיים והאחר של מספרים אי-זוגיים.

הגדרת המחלקה *EvenOdd*

בבעיה זו אנו נדרשים לפתח אלגוריתם המפצל מערך אחד לשני מערכים שונים.

הגדרת התכונות

התכונות של עצם מסוג EvenOdd כוללות שלושה מערכים: המערך השלם, מערך הזוגיים ומערך האי-זוגיים.

הגדרת הפעולות

פעולות של עצם מסוג EvenOdd הן: הפעולה הבונה המקבלת את המערך השלם, פעולה המפצלת את המערך השלם לשני מערכים ושתי פעולות גישה – אחת מחזירה את מערך הזוגיים והאחרת מחזירה את מערך האי-זוגיים.

נבחן את האלגוריתם הדרוש לביצוע פעולת הפיצול:

כדי לדעת את הגודל הדרוש עבור כל אחד מהמערכים החדשים, עלינו תחילה לסרוק את המערך השלם, למנות את מספר הזוגיים ואת מספר האי-זוגיים, ורק לאחר מכן נוכל להקצות את המערכים החדשים בהתאם לגודל הנדרש.

לשם הפיצול עלינו לסרוק את המערך השלם ובו-בזמן למלא את המערכים החדשים בהתאם.

שימו ♥: בתהליך הפיצול עלינו להשתמש ב-3 אינדקסים שונים. אחד עבור המערך השלם, והוא יתקדם לתא הבא בכל סבב של הלולאה. בנוסף, לכל אחד משני המערכים החדשים יהיה אינדקס אשר יקודם ב-1 רק במקרה הצורך.

מימוש המחלקה

```
/*
    מחלקה לפיצול מערך שלם לזוגיים ולאי-זוגיים
*/
public class EvenOdd
{
      // הגדרת התכונות
      private int[] arr; // המערך השלם
      private int[] oddArr; // מערך המספרים הזוגיים
      private int[] evenArr; // מערך המספרים האי-זוגיים
      // פעולה בונה
      public EvenOdd(int[] a)
      {
            arr = a;
      }
      // פיצול
      public void Split()
      {
            // סופרים כמה זוגיים יש במערך
            int evenCount = 0;
            for (int i = 0; i < arr.Length; i++)
                  if (arr[i] % 2 == 0)
                        evenCount++;
            // מקצים מקום עבור מערך הזוגיים
            evenArr = new int[evenCount];
            // כעת נוכל להסיק כמה אי זוגיים יש במערך
            oddArr = new int[arr.Length - evenCount];
            int iEven = 0, iOdd = 0;
            for (int i = 0; i < arr.Length; i++)
            {
                  if (arr[i] % 2 == 0)
                  {
                        evenArr[iEven] = arr[i];
                        iEven++;
                  }
                  else
                  {
                        oddArr[iOdd] = arr[i];
                        iOdd++;
                  }
            }
      } // Split
      public int[] GetOddArr()
      {
            return oddArr;
      }
      public int[] GetEvenArr()
      {
```

```
            return evenArr;
        }
} // EvenOdd
```

סוף פתרון בעיה 5

שימו ♥: בפעולה הבונה של מחלקה זו, כמו בפעולה הבונה של בעיה 1 בפרק זה, אתחלנו את התכונה `arr` באמצעות השמה פשוטה ולא באמצעות העתקה. גם כאן ויתרנו על ההגנה על המערך `arr` מפני שינויים לא מבוקרים מבחוץ, בתמורה לחיסכון במקום ובזמן.

שאלה 13.10

א. נתונות שתי סדרות של מספרים. עליכם לפתח אלגוריתם היוצר סדרה שלישית, המכילה את המספרים הקיימים בשתי הסדרות. הגדירו מחלקה בשם MatchNumbers, במחלקה יוגדרו שני מערכים כתכונות, ותוגדר פעולה שמחפשת את המספרים הקיימים בשני המערכים ומחזירה מערך חדש המכיל מספרים אלו.

ב. כעת נתון כי שני המערכים ממוינים בסדר עולה. האם האלגוריתם ישתנה בעקבות מידע זה? כתבו מחדש את הפעולה לפי האלגוריתם החדש.

בעיה 6

מטרת הבעיה ופתרונה: יחסים בין מערכים ושימוש בפעולת עזר פרטית.

פתחו אלגוריתם שיגדיר מערך דו-ממדי של מספרים שלמים בגודל NxM (N שורות ו-M עמודות). האלגוריתם ייצור מערך חד-ממדי בן N איברים כך שבמיקום ה-k במערך החד-ממדי יהיה סכום איברי השורה ה-k במערך הדו-ממדי.

למשל, אם המערך הדו-ממדי הוא:

1	3	2
3	1	5
5	6	2
3	7	2
7	6	3

אז המערך החד-ממדי יהיה:

6	9	13	12	16

במקום השלישי מוצב הערך 13, כי סכום האיברים בשורה השלישית במערך הדו-ממדי הוא 13.
כתבו מחלקה ובה מוגדר מערך דו-ממדי כתכונה ומוגדרת פעולה המחזירה את מערך הסכומים.

הגדרת המחלקה *סכום-שורות*

בבעיה זו אנו נדרשים לפתח אלגוריתם הסוכם שורות של מערך דו-ממדי ומשים את התוצאות במערך חד-ממדי.

הגדרת התכונות

התכונה של עצם מסוג סכום-שורות היא המערך הדו-ממדי.

הגדרת הפעולות

הפעולות של עצם מסוג סכום-שורות הן: פעולה בונה המקבלת את המערך הדו-ממדי, ופעולה לסכימת השורות המחזירה את המערך החד-ממדי הנדרש.

נבחן את האלגוריתם שיסכום את השורות:

עלינו לעבור על המערך הדו-ממדי, שורה אחר שורה, לסכום את איברי השורה ולהציב את הסכום במיקום המתאים במערך החד-ממדי. לשם כך, נשתמש בפעולת עזר **פרטית** אשר תקבל מספר שורה ותחזיר את סכומה.

מימוש המחלקה

```
/*
          מחלקה לסכימת שורות מטריצה
*/
public class LinesSum
{
      // הגדרת התכונות
      private int[,] mat;
      // פעולה בונה
      public LinesSum(int[,] a)
      {
            mat = a;
      }
      // פעולה פרטית לסכימת שורה במטריצה
      private int SumOneLine(int lineNum)
      {
            int sum = 0;
            for (int i = 0; i < mat.GetLength(1); i++)
                  sum = sum + mat[lineNum,i];
            return sum;
      }
      // סכימת השורות והחזרת הסכומים במערך
      public int[] SumLines()
      {
            // מקצים מקום עבור מערך הסכומים
            int[] sumArr = new int[mat.GetLength(0)];
            for (int i = 0; i < mat.GetLength(0); i++)
                  // קריאה לפעולה פרטית לקבלת הסכום
                  sumArr[i] = SumOneLine(i);
            return sumArr;
      } // SumLines
} // LinesSum
```

סוף פתרון בעיה 6

שאלה 13.11

כתבו את מחלקת "סכומי ספרות". הגדירו במחלקה תכונה יחידה: nums – מערך חד-ממדי של מספרים שלמים, והגדירו שתי פעולות:

א. פעולה המחזירה מערך אשר כל אחד מאיבריו הוא סכום הספרות של האיבר המקביל לו ב-nums. לדוגמה עבור המערך nums הבא:

123	45	532	12	75

יוחזר המערך:

6	9	10	3	12

ב. "ספרת הסכום" היא הספרה אשר מתקבלת כאשר סוכמים ספרות מספר שוב ושוב עד אשר מתקבלת ספרה יחידה. הגדירו פעולה המחזירה מערך אשר כל אחד מאיבריו הוא "ספרת הסכום" של האיבר המקביל לו ב-nums. לדוגמה, עבור המערך nums מסעיף א יוחזר המערך:

6	9	1	3	3

הדרכה: הגדירו במחלקה זו פעולת עזר פרטית המקבלת מספר וסוכמת את ספרותיו.

שאלה 13.12

אופיר קיבלה מערכת שעות לשנת הלימודים החדשה, והחליטה לכתוב את המחלקה "מערכת שעות" אשר תעזור לה בניהול סדר היום שלה. המחלקה מכילה מערכת שעות כתכונה (הניחו כי אופיר לומדת 5 ימים בשבוע, 8 שעות בכל יום). בנוסף לפעולה הבונה המקבלת את מערכת השעות ומאתחלת אותו, במחלקה יוגדרו שלוש הפעולות הבאות: הפעולה הראשונה מקבלת מספר יום (1 עבור יום ראשון, 2 עבור יום שני וכך הלאה) ומחזירה מערך של מחרוזות המייצג את מערכת השיעורים עבור יום זה. הפעולה השנייה מקבלת יום ושעה (לדוגמה 1 1 ייצג את השיעור הראשון ביום ראשון) ותחזיר מחרוזת המייצגת את השיעור המתקיים במועד המבוקש. פעולה שלישית תקבל שם מקצוע (כמחרוזת) ותחזיר כמה פעמים בשבוע אופיר לומדת מקצוע זה. ממשו מחלקה זו בשפת #C.

בעיה 7 מבגרות 2007 תשס"ז, שאלה 9

מטרת הבעיה ופתרונה: הצגת בעיה שלפתרונה יש להשתמש בצובר ובחיפוש ערך במערך.

בעל חניון למכוניות החליט למחשב את ניהול החניון.

בחניון יש 318 מקומות חניה, הממוספרים מ-1 עד 318. החניון פתוח בכל יום מהשעה 6:00 עד השעה 23:00. בחניון משלמים 14 שקל על כל שעת חניה. כלי רכב יכולים להיכנס או לצאת מהחניון רק בשעות שלמות. אפשר להיכנס לחניון עד השעה 22:00 (כולל). בסוף כל יום לא נשארות מכוניות בחניון.

א. פתחו אלגוריתם שיטפל בניהול החניון ביום מסוים. עליך לפתח את האלגוריתם לפי השלבים שלפניך:

i בחרו במשתנים עיקריים, הגדירו את טיפוסיהם ותארו את תפקידיהם.

ii פרקו את הבעיה לתת-משימות. באלגוריתם יש לכלול את התת-משימות האלה:

* פתיחת החניון בתחילת היום – איפוס הקופה וסימון כל מקומות החניה כפנויים.
* כניסת רכב לחניון – קליטת שעת הכניסה של הרכב (מספר שלם בין 6 ל-22 (כולל)), מציאת מקום פנוי לרכב, הדפסת המספר של המקום הפנוי, סימון מקום החניה כתפוס. אם אין מקום פנוי, תודפס הודעה מתאימה.
* יציאת רכב מהחניון – קליטת המספר של מקום החניה של הרכב, קליטת שעת היציאה שלו (מספר שלם בין 7 ל-23 (כולל)), חישוב והדפסה של התשלום, עדכון הקופה, סימון מקום החניה כפנוי.
* סגירת החניון בסוף היום – הדפסת סך כל הכסף שנגבה במשך היום בעבור חניית מכוניות בחניון.

הגדירו לכל אחת מהתת-משימות את מטרתה (טענת כניסה וטענת יציאה), וישמו כל אחת מהתת-משימות באמצעות פעולה בשפת C#.

ב. כתבו בשפת C# תוכנית לניהול החניון, שתיישם את האלגוריתם שפיתחתם בסעיף א.
לאחר פתיחת החניון התוכנית תקלוט בעבור כל רכב: את הקוד 1 אם הרכב נכנס לחניון, ואת הקוד 2 אם הרכב יוצא מהחניון, ותבצע את התת-משימות בהתאם. הקליטה תסתיים כאשר ייקלט הקוד -1. לאחר סיום הקליטה התוכנית תסגור את החניון. עליכם להשתמש בפעולות שיישמתם בסעיף א.

הניחו: שעת היציאה היא תמיד מספר גדול יותר משעת הכניסה.

הערה: אין צורך לבדוק את תקינות הקלט.

הגדרת המחלקה *חניון*

הגדרת התכונות

התכונות של עצם מסוג חניון כוללות מערך בגודל 318, אשר ייצג את מקומות החניה בחניון. תא שערכו 0 יחשב כמקום חניה פנוי. בתא תפוס תישמר שעת כניסת הרכב לחניון. בנוסף נשמור במשתנה מטיפוס שלם את קופת החניון אשר תצבור את תשלומי הרכבים בעת יציאתם מן החניון. אם כך, תכונות של עצם מסוג "חניון" הן:

- **spaces** – מערך של שלמים המייצגים את המקומות בחניון. גודל המערך 318.
- **cash** – מספר שלם הצובר את סכום הכסף בקופה.
- **NUM_OF_SPACES** – קבוע שלם שערכו 318 – מייצג את מספר המקומות בחניון.
- **HOURLY_RATE** – קבוע שלם שערכו 14 – מייצג את המחיר לשעת חניה.

הגדרת הפעולות

להלן פעולות העצם כפי שהן מפורטות בהגדרת השאלה:

- **הפעולה הבונה** – הפעולה תקצה מקום למערך spaces ותאתחל את התאים ב-0. בנוסף הפעולה תאפס את הסכום המצטבר בקופה.
- **כניסת רכב לחניון** – הפעולה תקבל כפרמטר את שעת הכניסה לחניון – מספר שלם בין 6 ל-22. הפעולה תבדוק במערך spaces אם יש מקום פנוי בחניון. אם יש מקום, שעת הכניסה תישמר בתא המתאים במערך והפעולה תחזיר את מספרו של המקום הפנוי שנמצא, אחרת יוחזר 0.
- **יציאת רכב מהחניון** – הפעולה תקבל כפרמטר את מספר מקום החניה – מספר שלם בין 1 ל-318, ואת שעת היציאה – מספר שלם בין 7 ל-23. הפעולה תסמן את המקום כפנוי, תחשב את הסכום לתשלום, תעדכן את הסכום המצטבר בקופה ותחזיר את הסכום לתשלום.
- **סגירת החניון** – הפעולה תחזיר את הסכום שהצטבר בקופה.

שימו ♥: בחרנו לא לקלוט מידע או להציג אותו בתוך הפעולות השונות. במקום זאת, הגדרנו לכל פעולה פרמטרים שבאמצעותם היא מקבלת את פרטי המידע הנדרש לביצועה. כמו כן, כל פעולה מחזירה ערך המייצג את תוצאת ביצועה. נקלוט את הנתונים ונציג את התוצאות בפעולה הראשית.

מימוש המחלקה

```
/*
        מחלקת חניון
*/
public class ParkingLot
{
    private int[] spaces;
    private int cash;
    private const int HOURLY_RATE = 14;
    private const int NUM_OF_SPACES = 318;
    // פעולה בונה, מאפסת את הקופה ואת המערך  //
    public ParkingLot()
    {
        cash = 0;
        spaces = new int[NUM_OF_SPACES];
        for (int i = 0; i < spaces.Length ; i++)
            spaces[i] = 0;
    }
    // פעולה המקבלת שעת כניסה ומחזירה את מספרו של המקום הפנוי הראשון, //
    // אם אין מקום פנוי יוחזר 0 //
    public int CarIn(int hour)
    {
        int spaceNumber = 0;
        int i = 0;
        while (i < spaces.Length && spaceNumber == 0)
        {
            if(spaces[i] == 0)
            {
                spaceNumber = i + 1;
                spaces[i] = hour;
            }
            i++;
        }
        return spaceNumber;
    }
    //פעולה המקבלת שעת כניסה ומקום בחניון, ומחזירה את הסכום שיש לשלם//
    public int CarOut(int spaceNumber, int hour)
    {
        int sumToPay;
        sumToPay  = (hour - spaces[spaceNumber-1]) * HOURLY_RATE;
        spaces[spaceNumber-1] = 0;
        cash = cash + sumToPay;
        return sumToPay;
    }
    // פעולה המחזירה את הפדיון היומי //
    public int EndOfDay()
    {
        return cash;
    }
} // class ParkingLot
```

הגדרת הפעולה הראשית

פירוק לתת-משימות

משימות הפעולה הראשית הן:

1. פתיחת החניון.
2. קליטת קודים: 1 לכניסת רכב, 2 ליציאת רכב, 1- לסיום יום פעילות.
3. סגירת החניון

בחירת משתנים

בפעולה הראשית יוגדר עצם מטיפוס חניון ומשתנים עבור הפעולה המבוקשת, שעת הכניסה או היציאה, מקום החניה והסכום לתשלום.

מימוש הפעולה הראשית

```
/*
            המחלקה הראשית
*/
using System;
public class ParkingManager
{
    public static void Main()
    {
        // הגדרת משתנים והקצאתם
        ParkingLot parkingLot = new ParkingLot(); // עצם מסוג חניון
        int action;                 // הפעולה המבוקשת
        int hour;                   // שעת כניסה או יציאה
        int numOfSpace;             // מספר מקום החניה
        int sumToPay;               // סכום לתשלום
        // קליטת הפעולה המבוקשת
        Console.Write("Enter 1 for entry, 2 for exit, -1 to end: ");
        action = int.Parse(Console.ReadLine());
        while (action != -1)
        {
            if (action == 1)  // כניסה לחניון
            {
                Console.Write("Enter the time - " +
                                          "a number between 6-22: ");
                hour = int.Parse(Console.ReadLine());
                numOfSpace = parkingLot.CarIn(hour);
                if (numOfSpace == 0)
                    Console.WriteLine("No free spaces");
                else
                    Console.WriteLine("Space number {0}",numOfSpace);
            }
            if (action == 2)  // יציאה מהחניון
            {
                Console.Write("Enter the time - " +
                                          "a number between 7-23: ");
                hour = int.Parse(Console.ReadLine());
                Console.Write("Enter your space number: ");
                numOfSpace = int.Parse(Console.ReadLine());
```

```
                sumToPay = parkingLot.CarOut(numOfSpace, hour);
                Console.WriteLine("You need to pay {0}", sumToPay);
            }
            Console.Write("Enter 1 for entry, 2 for exit, -1 to"
                                                    + " end: ");
            action = int.Parse(Console.ReadLine());
        } // while
        // סגירת החניון
        Console.WriteLine("Total sum at the end of the day: {0}",
                                            parkingLot.EndOfDay());
    } // Main
} // class ParkingManager
```

סוף פתרון בעיה 7

בעיה 8 מבגרות 2007 תשס"ז, שאלה 10

מטרת הבעיה ופתרונה: הצגת בעיה שלפתרונה יש להשתמש בשילוב של תבניות אלגוריתמיות: מנייה, האם ערך בסדרה מקיים תנאי והזזה של תת-סדרה שמאלה.

א. איברים ברצף במערך חד-ממדי הם איברים שהאינדקסים שלהם הם מספרים עוקבים. כתבו בשפת C# פעולה בשם Seven שתקבל מערך חד-ממדי בגודל 105 המכיל מספרים שלמים. הפעולה תבדוק אם יש במערך 7 (או יותר) איברים ברצף שהערך של כל אחד מהם הוא אפס. אם כן הפעולה תחזיר "אמת" אחרת תחזיר "שקר".

ב. כתבו בשפת C# פעולה בשם Shift שתקבל מערך חד-ממדי a בגודל 105 המכיל מספרים שלמים, ומספר שלם k בין 1 ל-4 (כולל). הפעולה תבצע הזזה של כל אחד מאיברי המערך k מקומות שמאלה מהמקום שבו הוא נמצא. לאחר ההזזה האיברים השמאליים לא יופיעו יותר במערך. ב-k המקומות הימניים במערך יוצב 0.

דוגמה עבור k=2 ומערך a בגודל 5:

3	7	2	0	1

ג. לאחר הפעלת הפעולה Shift המערך שיתקבל ייראה כך:

2	0	1	0	0

ד. נתון מערך חד-ממדי בגודל 105 המכיל מספרים שלמים. כתבו בשפת C# תוכנית שתשתמש בפעולה Seven ותבדוק אם יש במערך 7 (או יותר) איברים ברצף שהערך של כל אחד מהם הוא 0. אם אין – התוכנית תקלוט מספר שלם k בין 1 ל-4 (כולל) ותבצע את הפעולה Shift עם המספר k שנקלט. התוכנית תמשיך לקלוט מספרים ולבצע עם כל אחד מהם את הפעולה Shift, עד שיהיו במערך 7 (או יותר) איברים ברצף שכל אחד מהם הוא 0. בסיום הביצוע התוכנית תדפיס את המערך.

הערה: אין צורך לבדוק את תקינות הקלט.

בסעיפים א ו-ב נתבקשנו לכתוב פעולות המקבלות מערך כפרמטר. הפעולה הראשונה בודקת קיום תנאי כלשהו במערך והפעולה השנייה מבצעת הזזה לאיברי המערך. בסעיף ג עלינו לכתוב תוכנית המשתמשת בפעולות אלה.

מכיוון שנתבקשנו לכתוב פעולות ולא להגדיר עצם כלשהו נוכל לכתוב אותן כפעולות סטטיות במחלקת שירות (כפי שהוצג בבעיה 7 בפרק 11).

הגדרת המחלקה *שירותי-מערך*

כזכור, מחלקת שירות לא מגדירה תכונות כי תכונות שייכות לעצם ואנו לא יוצרים שום עצם. הפעולות הסטטיות פועלות על הפרמטרים שהן מקבלות ולא על תכונות. הפעלת הפעולות הסטטיות מתבצעת בדרך כלל ישירות דרך שם המחלקה ולא באמצעות עצם של המחלקה. כמו כן, לא נזדקק לפעולה בונה שתפקידה ליצור עצם ולאתחל את תכונותיו.

הגדרת הפעולות

להלן פעולות השירות כפי שהן מפורטות בהגדרת השאלה:

- **Seven** – פעולה סטטית המקבלת מערך של שלמים כפרמטר ובודקת אם קיים בו רצף של שבעה אפסים או יותר. אם רצף כזה קיים הפעולה תחזיר **true** אחרת יוחזר **false**.
- **Shift** – פעולה סטטית המקבלת מערך של שלמים ומספר שלם k כפרמטר. הפעולה מזיזה את איברי המערך k מקומות שמאלה ומציבה אפס ב-k האיברים האחרונים של המערך. פעולה זו אפשר לבצע כך: נעבור על כל איבר במערך פרט ל-k הראשונים ונעביר אותו k מקומות שמאלה. לאחר ההזזה נשתמש בלולאה העוברת על k האיברים האחרונים של המערך ומציבה בהם אפס.

שימו ♥: מכיוון שהגישה למערך ב-C# נעשית דרך הפניה (כתובת בזיכרון), כל שינוי שהפעולה תבצע במערך שהתקבל כפרמטר יהיה למעשה שינוי במערך המקורי שהועבר לפעולה.

מימוש המחלקה

```
/*
 מחלקת שירותי מערך
*/
public class ArrayServices
{
     // הפעולה מקבלת כפרמטר מערך חד-ממדי
     // הפעולה מחזירה "אמת" אם קיים רצף של שבעה אפסים או יותר,
     // אחרת יוחזר "שקר"
     public static bool Seven(int[] a)
     {
          int counter = 0;
          for(int i = 0; i < a.Length; i++)
          {
              if (a[i] == 0)
                  counter++;
              else
              {
                  if (counter >= 7)
                      return true;
                 else
                       counter = 0;
              }
          }
          if (counter >= 7)
             return true;
          else
             return false;
     }
```

```
    // הזזת כל האיברים החל מהאיבר ה-k, k מקומות שמאלה
    public static void Shift(int[] a, int k)
    {
        for (int i = k; i < a.Length; i++)
            a[i-k] = a[i];
        for (int i = a.Length - k; i < a.Length; i++)
            a[i] = 0;
    }
}//ArrayServices
```

את פעולת ה-Shift אפשר לבצע בכמה דרכים. דרך נוספת היא לבצע k פעמים הזזה אחת שמאלה של כל איברי המערך:

```
    public static void Shift(int[] a, int k)
    {
            for (int i = 0; i < k; i++)
                for (int j = 0; j < a.Length - 1; j++)
                    a[j] = a[j+1];
            for (int i = a.Length - k; i < a.Length; i++)
                a[i] = 0;
    }
```

מימוש הפעולה הראשית

הפעולה הראשית תקלוט מערך של 105 שלמים ותבדוק באמצעות הפעולה Seven אם יש במערך לפחות 7 אפסים ברצף. אם אין – התוכנית תקלוט מספר שלם k בין 1 ל-4 ותבצע את הפעולה Shift עם המספר k שנקלט. התוכנית תמשיך לקלוט מספרים ותבצע עם כל אחד מהם את הפעולה Shift, עד שהפעולה Seven תחזיר **true**. בסיום הביצוע התוכנית תדפיס את המערך.

עד כה יצרנו מחלקה נפרדת עם הפעולה הראשית `Main`, והפעולות הסטטיות הופעלו בציון שם המחלקה, למשל כך: `ArrayServices.Shift(a,3)`. קיימת גם אפשרות נוספת והיא לשים את הפעולה הראשית `Main` בתוך מחלקת השירות, יחד עם הפעולות Seven ו-Shift. במקרה זה נוכל להפעיל את הפעולות הסטטיות בצורה ישירה ללא קידומת שם המחלקה. התבוננו בפתרון הבא:

```
/*
      מחלקת שירותי מערך והפעולה הראשית
*/
using System;
public class ArrayServices
{
      public static bool Seven(int[] a)
      {
       ... // גוף הפעולה מופיע כאן
      }
     public static void Shift(int[] a, int k)
      {
       ... // גוף הפעולה מופיע כאן
      }
     // שימו לב לכך שהפעולה הראשית מזמנת את הפעולות הסטטיות
     // בצורה ישירה מבלי לציין את שם המחלקה כקידומת
     public static void Main()
      {
          int k;
```

```
        int[] a = new int[105];
        // קלט למערך
         Console.WriteLine("Enter 105 array values: ");
         for (int i = 0; i < a.Length; i++)
             a[i] = int.Parse(Console.ReadLine());
         while (Seven(a) ==  false)
         {
             Console.Write("Enter a number between 1-4: ");
             k = int.Parse(Console.ReadLine());
             Shift(a,k);
         }
         // הדפסת המערך
         Console.WriteLine("The array values are: ");
         for (int i = 0; i < a.Length; i++)
              Console.Write("{0} ", a[i]);
   }//Main
}//ArrayServices
```

סוף פתרון בעיה 8

שאלות נוספות

1. לפניכם מחלקת שירות להצפנה, לפענוח ולבדיקת סיסמאות. אלגוריתם ההצפנה פועל כך: כל אות אנגלית הופכת להיות האות **הבאה** וכל ספרה הופכת להיות הספרה **הקודמת**. ההצפנה פועלת באופן מעגלי כך שהאות הבאה אחרי האות z היא האות a והספרה הקודמת לספרה 0 היא הספרה 9. לדוגמה הסיסמה: abz9130 תוצפן כך: bca8029.

המחלקה כוללת את הפעולות הבאות:

- **Encrypt** – פעולת הצפנה המקבלת סיסמה ומחזירה סיסמה מוצפנת לפי האלגוריתם שתואר לעיל.
- **Decrypt** – פעולת פיענוח המקבלת סיסמה מוצפנת ומחזירה סיסמה גלויה.
- **IsLegal** – פעולת לבדיקת חוקיות סיסמה. הפעולה מקבלת סיסמה (גלויה או מוצפנת) ובודקת את חוקיותה. סיסמה תיחשב כחוקית אם היא עונה על הדרישות הבאות: אורכה 6-8 תווים, היא מורכבת מאותיות אנגליות ומספרות בלבד וכוללת לפחות אות אחת וספרה אחת.

השלימו את פעולות המחלקה שלהלן:

```
/*
          מחלקת שירות לסיסמאות
 */
public class Password
{
    // הפעולה מקבלת סיסמה ומחזירה סיסמה מוצפנת
    public static string Encrypt(string pass)
    {
      ...
    }
```

```
    // הפעולה מקבלת סיסמה מוצפנת ומחזירה סיסמה גלויה
    public static string Decrypt(string pass)
    {
      ...
    }
    // הפעולה מקבלת סיסמה ומחזירה אם הסיסמה חוקית
    public static bool IsLegal(string pass)
    {
      ...
    }
}
```

2. לפניכם מחלקה המגדירה ארנק. בארנק יש מטבעות של 10 ₪, 5 ₪, ו-1 ₪. כדי שהארנק לא יהיה כבד, נגרום למספר המטבעות בו להיות מינימלי. למשל, אם בארנק יש 21 שקלים, הארנק יכיל שתי מטבעות של עשרה שקלים, אפס מטבעות של 5 שקלים ומטבע אחד של שקל אחד.

המחלקה מכילה את הפעולות הבאות:

- **הפעולה הבונה** – פעולה המאתחלת את הארנק להיות ריק.
- **GetAmount** – פעולה המחזירה מספר שלם המייצג את סכום הכסף שבארנק בשקלים.
- **AddMoney** – פעולה המקבלת סכום כסף שיש להוסיף לארנק ומעדכנת את מספר המטבעות בו מכל סוג.
- **Pay** – פעולה המקבלת סכום כסף לתשלום. הפעולה מפחיתה מהארנק את הסכום המבוקש ומשאירה את הארנק במספר מינימלי של מטבעות כפי שתואר לעיל. הפעולה מחזירה ערך בוליאני המציין אם היה די כסף בארנק.

השלימו את תכונות ופעולות המחלקה שלהלן:

```
/*
        מחלקת ארנק
*/
public class Wallet
{
    private int ________;
    private int ________;
    private int ________;
    public Wallet()
    {
        ________ = ___;
        ________ = ___;
        ________ = ___;
    }
    // הפעולה מחזירה את סכום הכסף שבארנק בשקלים
    public int GetAmount()
    {
        return _________________________________;
    }
```

```
    // הפעולה מקבלת סכום כסף שיש להוסיף לארנק
    public void AddMoney(int givenSum)
    {
        int sum = GetAmount() + givenSum;
        _________ = _________ + sum / 10;
        sum = sum % 10;
        _________ = _________ + ______________;
        sum = _____________;
        _________ = _________ + ______________;
    }
    // הפעולה מקבלת סכום כסף לתשלום,
    // הפעולה מחזירה ערך בוליאני המציין אם היה די כסף בארנק
    public bool Pay(int sumToPay)
    {
        if (______________)
            return false;
        AddMoney(_____________________);
        return ___________;
    }
}  // class Wallet
```

3. מפעל הפיס מתכנן הגרלה חדשה. בהגרלה זו יונפקו מספר קבוע של כרטיסים, וסכום הפרסים הכולל יהיה 1,000,000 ₪. לכל כרטיס יהיה מספר מזל בן 8 ספרות (ייתכנו מספר כרטיסים שלהם אותם מספרי מזל). כרטיס זוכה יהיה כרטיס אשר סכום ארבע הספרות הראשונות של מספר המזל בו יהיה שווה לסכום ארבע הספרות האחרונות. בסיום הנפקת הכרטיסים ייקבע גודל הזכייה לכרטיס יחיד, כיוון שרק אז ידעו את מספר הכרטיסים הזוכים. אם כלל לא הונפקו כרטיסים זוכים, סכום הזכייה לכרטיס יהיה 0. לפניכם מחלקה המגדירה את מכונת הנפקת הכרטיסים. השלימו את פעולות המחלקה שלפניכם:

```
    /*  מחלקה להנפקת כרטיסי הגרלה  */
    public class TicketFactory
    {
        // הגדרת התכונות
        private const int TOTAL_PRIZE = __________;
        private int ticketCounter;
        private int winTicketCounter;
        // פעולה בונה //
        public TicketFactory(int totalNumOfTickets)
        {
            ticketCounter = totalNumOfTickets;
            winTicketCounter = 0; // מספר הכרטיסים הזוכים
        }
        // פעולה המגרילה מספר מזל חדש //
        public int GetNextLuckyNum()
        {
            if (ticketCounter <= 0)
                return 0;
            ticketCounter--;
            Random rnd = new Random();
            int num = __________________;// הגרלת מספר מזל בן 8 ספרות
```

```
        if (_____________________)
              winTicketCounter++;
        return num;
    }
    // פעולה המחזירה את סכום הזכייה של כרטיס זוכה
    public double GetWinningPrize()
    {
        if (winTicketCounter == 0)
              return _____________________________;
        else
              return _____________________________;
    }
    // פעולה פרטית הבודקת אם מספר מזל הוא זוכה
    private bool IsWinnerNum(int num)
    {
        return GetDigitsSum(__________) ==
                                      GetDigitsSum(___________);
    }
    // פעולה פרטית המחזירה את סכום ספרות המספר
    private int GetDigitsSum(int num)
    {
        int sum = ______;
        while (____________)
        {
              _______________;
              _______________;
        }
        return _______;
    }
}
```

4. בחנות התקליטים הוחלט לבדוק איזה סוג מוזיקה הוא הנמכר ביותר. כל אריזות התקליטים בחנות מקודדות בקוד בן 2 ספרות. הספרה השמאלית מייצגת את המחלקה והספרה הימנית היא מספר התקליטים באריזה.

המחלקות הקיימות בחנות:

1 – קלאסי	3 – ישראלי	5 – היפ הופ
2 – רוק	4 – ג'אז	6 – רגאי

לדוגמה, אריזה בקוד 62, מכילה תקליט כפול ממחלקת רגאי.

פתחו וממשו אלגוריתם הקולט את קודי אריזות התקליטים שנמכרו במשך היום, ומציג כפלט באיזו מחלקה נמכר מספר התקליטים הגדול ביותר. הקלט מסתיים בקוד 0.

שימו לב, תקליט כפול נחשב כשני תקליטים, תקליט משולש נחשב כשלושה וכן הלאה.

הדרכה: כתבו מחלקה המגדירה חנות תקליטים. הגדירו תכונות לייצוג המידע הנדרש ופעולה המעדכנת את הנתונים לאחר כל מכירה.

5. בהשראת בגרות 2006 תשס"ו, שאלה 9

בהינתן מערך דו-ממדי שאיבריו הם המספרים 0 ו-1. נגדיר "שרשרת" במערך כרצף של איברים בשורה מסוימת או רצף של איברים בעמודה מסוימת המכילים את המספר 1. אורך "שרשרת" הוא מספר האיברים ב"שרשרת".

אם בשורה כלשהי או בעמודה כלשהי אין "שרשרת", אורך ה"שרשרת" בה יהיה 0. נתון כי בכל שורה או עמודה יכולה להיות לכל היותר "שרשרת" אחת.

איבר במערך ייקרא "מוקף", אם הוא מכיל את המספר 1 וגם אורך ה"שרשרת" בשורה שבה הוא נמצא שווה לאורך ה"שרשרת" בעמודה שבה הוא נמצא.

<u>דוגמה</u>: במערך בגודל 4x5 שלפניכם יש שני איברים "מוקפים".

0	0	0	1	0
0	1	1	1	0
0	0	0	1	0
0	0	0	0	1

"איבר מוקף" (השורה השנייה, העמודה הרביעית)

"איבר מוקף" (השורה הרביעית, העמודה החמישית)

א. כתבו את המחלקה "שרשרת במערך" המכילה מערך דו-ממדי כתכונה, ובה פעולה בונה המקבלת מערך דו-ממדי ומאתחלת את התכונה. ניתן להניח שהמערך המתקבל כפרמטר תקין, כלומר מכיל רק את הערכים 0 ו-1, ושבכל שורה או עמודה יש לכל היותר שרשרת אחת.

ב. כתבו פעולה המקבלת שני מספרים שלמים המציינים מיקום של איבר במערך (אינדקסים), המספר הראשון מציין שורה והמספר השני מציין עמודה. הפעולה תחזיר **true** אם איבר זה הוא איבר "מוקף" אחרת – תחזיר הפעולה **false**.

ג. כתבו פעולה שתמנה את מספר האיברים ה"מוקפים" שיש במערך, ותחזיר ערך בהתאם. השתמשו בפעולה שכתבתם בסעיף ב.

6. בהשראת בגרות 2005 תשס"ה, שאלה 10

במפעל לתכשיטים מרכיבים שרשרות מְחֻרוּזִים בשלושה צבעים: אדום, צהוב וירוק. בכל שרשרת יש לפחות חרוז אחד מכל צבע. שרשרת "אחידה" היא שרשרת שבה יש מספר שווה של חרוזים מכל אחד מהצבעים. שרשרת "פגומה" היא שרשרת שאינה "אחידה".

<u>לדוגמה</u>: השרשרת הבאה היא "פגומה"

אדום — ירוק — צהוב — אדום

א. כתבו את המחלקה "מפעל תכשיטים". הגדירו פעולה המקבלת שרשרת ובודקת אם היא אחידה או פגומה. הגדירו פעולות המחזירות את המידע הבא: כמה שרשרות נוצרו באותו היום וכמה מתוכן היו פגומות.

ב. כתבו פעולה ראשית שקולטת את השרשרות שמיוצרות במפעל ביום מסוים. בעבור כל שרשרת יש לקלוט מחרוזת המורכבת מהאותיות R, G, B ו-Y המייצגות את הצבעים: אדום, ירוק, כחול וצהוב בהתאמה. הקלט עבור השרשרת שבדוגמה לעיל יהיה המחרוזת "RYGR". הקלט יסתיים במחרוזת ריקה "".

לאחר סיום הקלט יש להציג את מספר השרשרות שיוצרו במפעל באותו היום, ואת מספר השרשרות הפגומות. השתמשו במחלקה שכתבתם בסעיף א.

הערה: אין צורך בבדיקת תקינות הקלט.

7. פתחו אלגוריתם המקבל כקלט מספר שלם חיובי N, המבטא את מספר הבנים ואת מספר הבנות בכיתה (N בנים ו-N בנות); ואחריו רשימה של 2N מספרים. N המספרים הראשונים מייצגים עבור כל בן באיזו בת הוא בוחר, ו-N המספרים הבאים מייצגים עבור כל בת באיזה בן היא בוחרת. האלגוריתם נותן כפלט את מספר הזוגות התואמים, כלומר מספר הזוגות של בן ובת שבחרו זה בזה.
לדוגמה, הקלט הבא: 1 2 2 1 1 3 3 3 מציין כי בכיתה יש 3 בנים ו-3 בנות (המספר הראשון בקלט הוא 3). שלושת המספרים הבאים מפרטים את בחירת הבנים. בן מספר 1 בחר בבת מספר 3. בן מספר 2 בחר בבת מספר 1 וגם בן מספר 3 בחר בבת מספר 1. שלושת המספרים האחרונים מפרטים את בחירת הבנות: בת מספר 1 בחרה בבן מספר 2. בת מספר 2 בחרה בבן מספר 2 ובת מספר 3 בחרה בבן מספר 1.
אם כך, יש שני זוגות אשר בחרו זה בזה: בן מספר 1 בחר בבת מספר 3, וגם היא בחרה בו, וכך גם בן מספר 2 ובת מספר 1 בחרו זה בזה.
ממשו את האלגוריתם בשפת C#.

הדרכה: כתבו מחלקת "צמדים" אשר מגדירה שני מערכים: אחד עבור בחירת הבנים והאחר עבור בחירת הבנות. הגדירו במחלקה פעולה הסורקת את שני המערכים ומחזירה את מספר הזוגות התואמים.

8. בהשראת בגרות 2003 תשס"ג, שאלה 10
לקראת תחרות ארצית במדעי המחשב, נערכה בחינת מיון ל-1750 תלמידים. לתחרות הארצית יתקבלו תלמידים שציונם בבחינת המיון גבוה מהציון הממוצע של כל הנבחנים בבחינה זו.
א. כתבו מחלקת "תלמיד" המכילה את התכונות הבאות: שם, כתובת, מספר תעודת זהות, שפת התכנות המועדפת על התלמיד (C# או Java) וציון התלמיד בבחינת המיון.
ב. כתבו פעולה ראשית הקולטת את נתוני כל המועמדים, ומציגה כפלט שתי רשימות המכילות את שמות התלמידים שיתקבלו לתחרות הארצית, את כתובותיהם ואת מספרי תעודת הזהות שלהם. הרשימה הראשונה תכלול את פרטי התלמידים ששפת התכנות המועדפת עליהם היא C#, והרשימה השנייה תכלול את אלה שמעדיפים את Java.
השתמשו במחלקה שכתבתם בסעיף א.

הדרכה: הפעולה הראשית צריכה לשמור עצמים מסוג תלמיד לצורך הצגת הפלט המבוקש.
חשבו כיצד כדאי לשמור את הנתונים. להלן כמה אפשרויות:
1) לשמור את התלמידים במערך משותף לפי סדר קליטתם.
2) לשמור את התלמידים בשני מערכים לפי שפת התכנות המועדפת עליהם. האם ניתן לדעת מראש כמה מקום להקצות לכל מערך?
3) לשמור את התלמידים במערך משותף בהפרדה בין שתי הקבוצות (אלה המעדיפים את C# ואלה המעדיפים את Java) קבוצה אחת תמלא את המערך מההתחלה לכיוון הסוף והקבוצה השנייה תמלא את המערך מהסוף להתחלה.

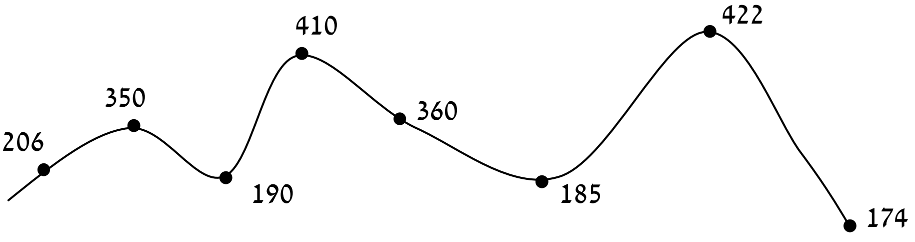
9. חנות הדיסקים "דיסקו" שכרה יועץ כדי לארגן מחדש את החנות. היועץ המליץ לארגן את קטלוג הדיסקים בחנות באופן הבא: היוצרים יהיו מסודרים בסדר אלפביתי ולכל יוצר תהיה רשימה ממוינת של שמות הדיסקים שלו.
כתבו תוכנית הקולטת שם של יוצר ושם של דיסק ובודקת אם הוא נמצא בחנות.

הדרכה: כתבו את המחלקה "יוצר" המכילה שם של יוצר ומערך של שמות הדיסקים שלו. הפעולה הראשית תכיל מערך של יוצרים ותמיין אותם לפי שמותיהם.
אין צורך לקלוט את שמות היוצרים ואת שמות הדיסקים שלהם. ניתן להניח שמערך היוצרים מאותחל בעצמים מסוג יוצר.

10. (הרחבת שאלה 13.6)
במפעל מועסקים 100 עובדים. בעל המפעל מעוניין לדעת נתונים על המשכורות שעליו לשלם בסופו של חודש מסוים. כל עובד נמצא במפעל 25 ימים בחודש ועובד מספר מסוים של שעות ביום. התעריף לעובדים אינו קבוע אלא אישי ומשתנה מעובד לעובד.
פתחו וממשו אלגוריתם הקולט עבור כל אחד מ-100 העובדים את שמו, את התעריף שלו לשעה, וכמה שעות עבד בכל אחד מ-25 ימי העבודה באותו חודש. האלגוריתם יציג כפלט את הערכים הבאים:
- סכום המשכורות שעל בעל המפעל לשלם לעובדיו.
- המשכורת הגבוהה ביותר המשולמת באותו חודש ושמו של העובד שקיבל משכורת זו.
- מספר המשכורות מעל לממוצע.
- ממוצע שעות העבודה ליום לכל עובדי המפעל.
- שמות העובדים שעבדו פחות ממספר השעות הממוצע ביום במשך יותר מ-5 ימים במהלך החודש.

11. במגמת גיאוגרפיה נלמדו מספר הגדרות:
פסגה – נקודת גובה אשר מימינה ומשמאלה יש נקודות נמוכות יותר.
מדרון – נקודת גובה אשר מצד אחד – נקודה גבוהה ממנה, ומצד שני – נמוכה ממנה.
מישור – נקודת גובה אשר לפחות מצד אחד קיימת נקודה אשר שווה לה.
עמק – נקודת גובה אשר מימינה ומשמאלה נקודות גבוהות יותר.
לאחר מכן יצאו לטיול לימודי שבו המדריך הכריז מידי פעם על גובה השטח. התלמידים נדרשו לרשום עבור כל נקודה (פרט לראשונה ולאחרונה) את הגדרתה הגיאוגרפית, ולספור את מספר הפסגות שעברו.

לדוגמה: עבור הנקודות בתרשים הבא המסודרות משמאל לימין יש לרשום: פסגה עמק פסגה מדרון עמק פסגה (לא נתייחס לנקודה הראשונה ולנקודה האחרונה) ונספרו בסך הכול 3 פסגות.



אורי החליט לכתוב תוכנית אשר מקבלת כקלט מספר המציין כמה נקודות גובה הוכרזו בטיול, ואחריו רשימה של נקודות הגובה. פלט התוכנית יהיה שעורי הבית שהוא נדרש להגיש.

הדרכה: הגדירו מחלקה בשם "מסלול טיול" המכילה את רשימת נקודות הגובה כתכונה. הגדירו פעולות המחזירות את המידע המבוקש.

סיכום

בפרק זה הצגנו בעיות המשלבות פיתוח אלגוריתמי יחד עם הגדרת מחלקות מתאימות לפתרון הבעיות. במהלך פתרון בעיה זיהינו את התכונות הנדרשות לייצוג עצם מהמחלקה ואת הפעולות הנדרשות למימוש האלגוריתמים המתאימים לפתרונה. ראינו מקרים רבים שפעולה כלשהי נעזרה בפעולה נוספת (פרטית) לצורך מימוש האלגוריתם. בכל הבעיות והשאלות שהוצגו בפרק נעשה שימוש בתבניות ובאלגוריתמים שנלמדו בפרקים הקודמים. בחלק מהבעיות שילבנו יותר מתבנית אחת והדגשנו את המורכבות הנוספת הקיימת בשילוב זה.

בנוסף, הצגנו מגוון רחב של בעיות אשר הופיעו בפרק ג של בחינת הבגרות, משנת 2003 ועד שנת 2007, וכן בעיות נוספות ברוח פרק זה. לכל בעיה צירפנו הנחיות מתאימות לפתרונה, כדי להקנות לכם את הניסיון הנדרש בניתוח ובפתרון בעיות הכוללות פיתוח אלגוריתמי והגדרת מחלקות.

פרק זה מסכם את לימוד יסודות מדעי המחשב ומכין אתכם בהצלחה לקראת השלב הבא – עיצוב תוכנה.

יסודות מדעי המחשב

בשפת #C – נספח

תמר בניה וד"ר מיכל ארמוני – ראשי צוות הכתיבה

יעל בילצ'יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי התבניות)

הילה קדמן (נספח)

ייעוץ: ד"ר דוד גינת

עריכה: לירון ברגר

תשע"א 2010

אוניברסיטת תל-אביב החוג להוראת המדעים

מטה מל"מ המרכז הישראלי להוראת המדעים ע"ש עמוס דה-שליט

משרד החינוך האגף לתכנון ולפיתוח תכניות לימודים

יסודות מדעי המחשב בשפת #C – נספח

תמר בניה וד״ר מיכל ארמוני – ראשי צוות הכתיבה

יעל בילצ׳יק

נעה גרדוביץ

עדי גרין

אתי מנשה (סעיפי תבניות)

הילה קדמן (נספח)

ייעוץ: ד״ר דוד גינת

עריכה: לירון ברגר

כל הזכויות שמורות © 2010

השראה הוצאה לאור, ת״ד 19022, חיפה 31190

טל׳: 04-8254752, פקס: 1534-8254752

E-Mail: books@hashraa.co.il

www.hashraa.co.il

השראה הוצאה לאור

עיצוב העטיפה: טל גרין

אין לשכפל, להעתיק, לצלם, לתרגם, להקליט, לאחסן במאגר מידע כלשהו, לשדר או לקלוט בכל דרך או בכל אמצעי אלקטרוני, אופטי או מכני (לרבות צילום, הקלטה, אינטרנט, מחשב ודואר אלקטרוני), כל חלק שהוא מהחומר שבספר זה. שימוש מסחרי מכל סוג בחומר הכלול בספר זה

אסור בהחלט, אלא ברשות מפורשת בכתב מהמו״ל ומהגורמים המפורטים להלן.

©

כל הזכויות שמורות

משרד החינוך

פעולות סטטיות ומחלקות שירות

רוב תכניות המחשב, המבצעות משימה משמעותית, מורכבות ממגוון של תת-משימות. הדרך היעילה להתמודד עם תכנון התכנית, עם בנייתה ועם תחזוקתה, היא חלוקתה ליחידות קטנות יחסית שכל אחת מהן מבצעת תת-משימה מוגדרת.

היתרונות בחלוקת המשימה הכללית לתת משימות:

- **שיפור הקריאות** – התכנית הראשית עוסקת **במה** ולא **באיך**. במקום הרבה שורות קוד שקשה לעקוב אחריהן ולהבין מה הן מבצעות, תכלול התכנית סדרה של הוראות הפעלה לתת-משימות מוגדרות.
- **תחזוקה** – קל לאתר שגיאות בתכנית ולתקנן. מכיוון שכל פעולה עוסקת במשימה מוגדרת, ניתן להגיע ישירות לקטע הקוד הבעייתי, ולטפל בו.
- **חיסכון בכתיבה** – כאשר יש משימה מסוימת הצריכה להתבצע יותר מפעם אחת במהלך התכנית, ניתן לכתוב פעולה המבצעת את המשימה ולזמן אותה מספר פעמים לפי הנדרש.
- **חלוקת העבודה** – ניתן לחלק את המשימה הגדולה לכמה מתכנתים, כך שכל אחד מהם מטפל בתת-משימה מוגדרת שתהווה חלק מהמשימה הכללית.
- **שימוש יעיל בספריות** קיימות – לדוגמא: השימוש במחלקה Math המכילה אוסף של פעולות מתמטיות.

כאשר ניגשים לפתור בעיה תכנותית, יש לפרק את המשימה הכללית לתת-משימות המבצעות כל אחת פעולה מוגדרת, ואחר כך לחבר הכול לתכנית השלמה. כל תת-משימה היא פעולה עצמאית הפועלת על משתנים משלה, והתכנית הראשית מזמנת את הפעולות לפי הצורך.

קריאה לפעולה מתוך התכנית גורמת לתכנית להסתעף למקום אחר, לבצע את ההוראות הכתובות בו ולחזור להמשך ביצוע התכנית מהשורה שאחרי השורה שגרמה לסיעוף.

את הפעולה ניתן לזמן מספר פעמים במהלך התכנית.

1. פעולה שאינה מקבלת דבר ואינה מחזירה דבר

בעיה 1

מטרת הבעיה ופתרונה: הצגת פירוק לתת-בעיות ופתרון כל תת-בעיה בפעולה נפרדת.

> להלן תכנית המאפשרת להציג על המסך צורה גיאומטרית, בהתאם לבחירת המשתמש.
> הצורות האפשריות תהיינה: ריבוע, מלבן או משולש.

פירוק הבעיה לתת משימות

1. ציור ריבוע.
2. ציור מלבן.
3. ציור משולש.
4. כדי לאפשר בחירה בין האפשרויות השונות, נוסיף תפריט בחירה.

לו היינו כותבים את התכנית בכלים שרכשנו עד כה, קרוב לוודאי שהייתה מתקבלת תכנית עמוסה בהוראות קוד ושהיה צורך בטבלת מעקב או הרצה כדי להבין מה היא מבצעת. נציג פתרון המשתמש בפעולות מיוחדות לביצוע כל אחת מהמשימות שקבענו:

האלגוריתם

1. *הצג תפריט-בחירה()*
2. *קלוט את בחירת המשתמש ב-* choice
3. *אם ערכו של* choice *הוא* 1
   צייר-ריבוע()
4. *אם ערכו של* choice *הוא* 2
   צייר-מלבן()
5. *אם ערכו של* choice *הוא* 3
   צייר-משולש()

התכנית

```
using System;
public class Shapes
{
      //---  5x5 פעולה המציירת ריבוע בגודל ---
      static void DrawSquare()
      {
            for (int i = 0; i < 5; i++)
            {
                  for (int j = 0; j < 5; j++)
                        Console.Write(" *");
                  Console.WriteLine();
            }
      }
```

```
        //---  5x10 פעולה המציירת מלבן בגודל  ---
        static void DrawRectangle()
        {
              for (int i = 0; i < 5; i++)
              {
                    for (int j = 0; j < 10; j++)
                          Console.Write(" *");
              Console.WriteLine();
              }
        }

        //---  פעולה המציירת משולש ישר-זווית  ---
        //---    שאורך כל אחד מניצביו הוא 5   ---
        static void DrawTriangle()
        {
              for (int i = 0; i < 5; i++)
              {
                    for (int j = 0; j <= i; j++)
                          Console.Write(" *");
                    Console.WriteLine();
              }
        }

        //---  פעולה המציגה תפריט בחירה  ---
        static void ShowMenu()
        {
              Console.WriteLine("Geometric shape menu: ");
              Console.WriteLine("    1.  Square      ");
              Console.WriteLine("    2.  Rectangle   ");
              Console.WriteLine("    3.  Triangle    ");
              Console.WriteLine();
        }

        static void Main()
        {
              int choice;

              ShowMenu();
              Console.Write("Type shape number --> ");
              choice = int.Parse(Console.ReadLine());

              if (choice == 1)
                    DrawSquare();
              if (choice == 2)
                    DrawRectangle();
              if (choice == 3)
                    DrawTriangle();
        }
  }
```

ההוראות drawSquare(), drawRectangle(), drawTriangle() ו-showMenu() הן משפטי זימון לפעולות (methods) המבצעות כל אחת תפקיד מוגדר.

לכל אחת מהפעולות מבנה דומה המתחיל בכותרת הפעולה, ואחריה גוף הפעולה התחום בתוך סוגריים מסולסלים:

```
//---  תיעוד הפעולה  ---
static void  שם-הפעולה ()
{
        //  גוף הפעולה
}
```

כותרת הפעולה מזהה את הפעולה, ומהווה ממשק בינה לבין התכנית המזמנת את הפעולה, כלומר - מספקת מידע איך להשתמש בפעולה:

רמת הרשאה – רכיב זה מאפשר לקבוע למי מותר לגשת לפעולה. המילה **public** מציינת גישה ציבורית, כלומר ניתן לגשת לפעולה מכל מקום בתכנית. בשלב זה של העבודה, רכיב זה הוא רשות ולא חובה, ולכן לא השתמשנו בו בתכנית.

static – פעולה של מחלקה מתחילה תמיד במילה static המציינת שהפעולה היא class member, כלומר – פעולה השייכת למחלקה ולא פעולה השייכת לעצם.

טיפוס הערך המוחזר – יש פעולות שמחזירות ערך ובמקרה זה הן חייבות לציין את טיפוס הערך המוחזר. פעולות שלא מחזירות ערך, חייבות לציין זאת באמצעות המלה השמורה void (ריק).
הפעולות המוצגות בתכנית הן פעולות מסוג זה.

שם הפעולה – השם שבו נזמן את הפעולה. מקובל לתת לפעולה שם משמעותי המרמז על תפקידה. שם פעולה יתחיל באות גדולה. שם פעולה המכיל שתי מילים או יותר, יחובר למילה אחת בדרך המקובלת בשפת C#, כלומר – כל מילה תתחיל באות גדולה. למשל: DrawSquare().
שים ♥: אין לתת לפעולה שם שהוא מילה שמורה או הוראה בשפה.

גוף הפעולה – גוף הפעולה יהיה תחום בתוך סוגריים מסולסלים ויכלול את הוראות הביצוע של הפעולה. ההוראות הן הוראות השפה המוכרות.
הפעולה יכולה להגדיר משתני עזר לביצוע החישובים. משתנים אלו קרויים **משתנים פנימיים** או **משתנים מקומיים** והם מוכרים רק בתוך הפעולה שבה הם הוגדרו, גם אם קיימים משתנים בשם זהה בפעולה אחרת. שם המשתנה הפנימי יהיה לפי כללי מתן שמות בשפת C#. שים ♥: אין לתת למשתנה פנימי שם זהה לשם הפעולה שהוא מוגדר בה.

תיעוד הפעולה – מקובל לתעד כל פעולה בתכנית. (תכנית הלימודים מחייבת תיעוד). התיעוד יכלול: תיאור של מה שהפעולה מבצעת ומה היא מחזירה. אם הפעולה מקבלת ערכים (פרמטרים), נרשום תיאור של הערכים האלו. אם על הערכים האלה לקיים תנאי כלשהו כדי שהפעולה תפעל כראוי, נרשום תנאים אלו כהנחות שאנו מניחים לגביהם (למשל: הנחה: המספר אינו שלילי).

דוגמאות לתיעוד:

❍ פעולה המקבלת ...(תיאור הפרמטרים)... ומחזירה ...(מה מחזירה הפעולה)...
הנחות: ...(רשימת ההנחות שמניחה הפעולה לגבי הפרמטרים)...

❍ פעולה המקבלת ...(תיאור הפרמטרים)... ומבצעת ...(מה מבצעת הפעולה)...
הנחות: ...(רשימת ההנחות שמניחה הפעולה לגבי הפרמטרים)...

❍ טענת כניסה: ...(אילו פרמטרים מקבלת הפעולה ומהן ההנחות לגביהם)
טענת יציאה: ...(מה עושה / מחזירה הפעולה)

משפט הזימון לפעולה

ההוראה בתכנית המפעילה את הפעולה נקראת **משפט זימון**. את הפעולה מזמנים בציון שמה בתוספת סוגריים עגולים. למעשה, ההבחנה בין שם של משתנה לשם של פעולה, הוא בתוספת הסוגריים העגולים בשם הפעולה. למשל. למשל: ()Console.WriteLine, ()ShowMenu.

את הפעולות אפשר לכתוב במחלקה הראשית לפני הפעולה Main או אחריה.

סוף פתרון בעיה 1

2. פעולה המקבלת פרמטרים ואינה מחזירה דבר

בעיה 2

מטרת פתרון הבעיה: העברת פרמטרים לפעולה

נשנה את התכנית כך שתצייר צורות גיאומטריות בגדלים המותאמים לבקשת המשתמש בתכנית.
לאחר שיוצג התפריט ויבחר המשתמש את הצורה הרצויה, הוא יתבקש להקליד את מידות הצורה. ערכים אלו יועברו כ**פרמטרים** לפעולה המתאימה, והיא תצייר את הצורה במידות המבוקשות.

פעולה המקבלת ערך מהתכנית

```
//---  תיעוד הפעולה  ---
static void שם-הפעולה (טיפוס-פרמטר-1 שם-פרמטר-1, טיפוס-פרמטר-2 שם-פרמטר-2)
{
        //  גוף הפעולה
}
```

הפרמטרים שבשורת הכותרת, הנקראים גם **פרמטרים פורמאליים**, הם משתנים השייכים לפעולה, ומקבלים את ערכם מהתכנית באמצעות ההוראה המזמנת בתכנית, ושם הם נקראים: **פרמטרים אקטואליים** או **ארגומנטים**.

טיפוס-הפרמטר – הוא טיפוס הנתונים של הפרמטר המועבר לפעולה.
שם-הפרמטר – הוא שם המשתנה כפי שיוכר בפעולה.

- מספר הפרמטרים המועברים אינו מוגבל, ויש להצהיר על כל אחד מהם בנפרד. אם יש יותר מפרמטר אחד, יש להפריד ביניהם בסימני פסיק.
- שם הפרמטר הפורמאלי (המוכר בפעולה) אינו חייב להיות זהה לשם הפרמטר האקטואלי (המוכר בתכנית המזמנת).
- סדר הרישום ומספר הארגומנטים שיופיעו במשפט הזימון לפעולה, חייב להיות תואם לסדר ולטיפוס המוצהר ברשימת הפרמטרים. למשל: פעולה בשם compute המקבלת כפרמטר שלושה משתנים מספריים: שלם, שלם וממשי:

```
static void Compute (int a, int b, double x)
```

משפטי זימון אפשריים לפעולה יהיו:

```
compute(3, 12, 4.25);
```

או:

```
int n1 = -4, n2 = 5;
double x = 0.25;
compute (n1, n2, x);
```

ניסיון לזמן את הפעולה עם ארגומנטים בסדר שונה מהסדר שנקבע בכותרת הפעולה יגרום לשגיאה.

- הפרמטר מקבל את ערכו מהפעולה המזמנת. המשמעות – אין לקלוט בתוכו ערך חדש בתוך הפעולה.
- פרמטרים מטיפוס בסיסי (int, double וכד') מעבירים לפעולה את ערכם, כלומר – בפעולה נוצר עותק שלהם. כל שינוי בערכי הפרמטרים בתוך הפעולה, לא ישפיע או ישנו את ערכי הארגומנטים בתכנית המזמנת.

התכנית המקבלת כפרמטר את גודל הצורה הגיאומטרית

```
using System;
public class Shapes2
{
      //---   side  אורך צלע הריבוע   :טענת כניסה   ---
      //---   טענת יציאה: מצייר ריבוע שאורך צלעו side   ---
      static void DrawSquare(int side)
      {
            for (int i = 0; i < side; i++)
            {
                  for (int j = 0; j < side; j++)
                        Console.Write(" *");
                  Console.WriteLine();
            }
      }

      //---   טענת כניסה: length ו-width  אורך צלעות המלבן   ---
      //---   טענת יציאה: מצייר מלבן שאורך צלעותיו length ו-width   ---
      static void DrawRectangle(int length, int width)
      {
            for (int i = 0; i < width; i++)
            {
                  for (int j = 0; j < length; j++)
                        Console.Write(" *");
            Console.WriteLine();
            }
      }

      //---   טענת כניסה: trigSide  אורך ניצב משולש ישר זווית   ---
      //---   טענת יציאה:  מצייר משולש שאורך כל ניצב הוא trigSide   ---
      static void DrawTriangle(int trigSide)
      {
            for (int i = 0; i < trigSide; i++)
            {
                  for (int j = 0; j <= i; j++)
                        Console.Write(" *");
                  Console.WriteLine();
            }
      }
```

```
        //---  פעולה המציגה תפריט בחירה  ---
        static void ShowMenu()
        {
              Console.WriteLine("Geometric shape menu: ");
              Console.WriteLine("    1.  Square      ");
              Console.WriteLine("    2.  Rectangle   ");
              Console.WriteLine("    3.  Triangle    ");
              Console.WriteLine();
        }

        static void Main()
        {
              int choice;
              int side1, side2;

              ShowMenu();
              Console.Write("Type shape number --> ");
              choice = int.Parse(Console.ReadLine());

              if (choice == 1)
              {
                    Console.Write("type square side --> ");
                    side1 = int.Parse(Console.ReadLine());
                    DrawSquare(side1);
              }

              if (choice == 2)
              {
                    Console.Write("type rectangle length --> ");
                    side1 = int.Parse(Console.ReadLine());
                    Console.Write("type rectangle width --> ");
                    side2 = int.Parse(Console.ReadLine());
                    DrawRectangle(side1, side2);
              }

              if (choice == 3)
              {
                    Console.Write("type trianble base --> ");
                    side1 = int.Parse(Console.ReadLine());
                    DrawTriangle(side1);
              }
        }
}
```

סוף פתרון בעיה 2

הפעולה ()Main

מבט נוסף בתכנית, מראה שגם הפעולה הראשית ()Main היא פעולה סטטית המקבלת פרמטר ומחזירה void.

ההבדל בינה לבין הפעולות האחרות:

- הפעולה ()Main חייבת לקבל הרשאת גישה public כדי להיות מופעלת.
- כאשר מריצים את התכנית, המחשב מחפש את הפעולה ()Main וממנה תתחיל הריצה.

3. פעולה המקבלת פרמטרים ומחזירה ערך

בעיה 3 - שאלה 9 בגרות 2000

מטרת פתרון הבעיה: הדגמת השימוש בפעולות המחזירות ערך, שימוש במסננת קלט.

נגדיר "משקל" של מספר תלת-ספרתי כסכום של מכפלת שתי הספרות הראשונות של המספר ושל מכפלת שתי הספרות האחרונות שלו.

לדוגמא: ה"משקל" של המספר 327 הוא: $3 * 2 + 2 * 7 = 20$

א. כתוב פעולה שתקבל מספר תלת ספרתי ותחזיר את ה"משקל" שלו.

ב. כתוב קטע תכנית (או פעולה) שיקלוט מספרים תלת-ספרתיים ויחשב עבור כל מספר את ה"משקל" שלו, באמצעות הפעולה שכתבת בסעיף א'.
קטע התכנית יחשב וידפיס את סכום ה"משקלים". הקלט יסתיים כאשר סכום ה"משקלים" יהיה גדול מ-100.

פירוק הבעיה לתת-משימות

1. פעולה המקבלת מספר תלת ספרתי ומחזירה את המשקל שלו.
2. קלט מספרים תלת-ספרתיים, חישוב והדפסת משקלו של כל מספר.
3. הדפסת סכום המשקלים.

פירוק מעמיק יותר של הבעיה לתת-משימות

1. פעולה המקבלת מספר תלת ספרתי ומחזירה את המשקל שלו.
   פעולת עזר: פעולה המקבלת מספר דו-ספרתי ומחזירה את מכפלת ספרותיו.
2. קלט מספרים תלת-ספרתיים סיכום והדפסת משקלו של כל מספר.
   פעולת עזר: פעולה המחייבת קליטת מספר תלת-ספרתי חיובי, ומחזירה אותו.
3. הדפסת סכום המשקלים.

בחירת משתנים

num – מספר שלם, שומר את המספר התלת ספרתי.

sum – מספר שלם, שומר את סכום המשקלים.

האלגוריתם

1. *אפס את הסכום*
2. *כל עוד סכום המשקלים קטן או שווה ל-100 בצע:*
   2.1 *קלוט מספר תלת ספרתי במשתנה* num

2.2 הוסף לסכום את משקלו של num

3. הצג את סכום המשקלים.

פעולה המחזירה ערך:

```
//---   תיעוד הפעולה   ---//
static טיפוס-הערך-המוחזר  שם-הפעולה (רשימת פרמטרים)
{
        // גוף הפעולה
        return ערך להחזרה ;
}
```

טיפוס הערך המוחזר – כאמור, פעולות יכולות להחזיר ערך. ללא מידע לגבי הערך המוחזר, לא נדע כיצד להפעיל את הפעולה בהצלחה. עד כה, ראינו פעולות שלא מחזירות ערך (void), ועתה נכיר פעולות שמחזירות ערך כלשהו, למשל מספר שלם או ממשי, ערך בוליאני או תו. המשמעות היא שאם מוחזר ערך שאינו void, התכנית המזמנת תציין מה לעשות עם הערך המוחזר.
הערך חוזר אל השורה שהתבצע בה הזימון לפעולה.

משפט return – החזרת ערך מפעולה מתבצעת באמצעות משפט return. טיפוס הערך המוחזר מוגדר בכותרת הפעולה.
אם נקבע כי הפעולה מחזירה ערך, יוחזר הערך באמצעות משפט return.
אם נקבע כי הפעולה אינה מחזירה ערך, לא יופיע משפט ה-return בגוף הפעולה.

כאמור, במשפט זימון לפעולה המחזירה ערך, יש לציין מה לעשות עם הערך המוחזר (השמה למשתנה, הדפסתו, בדיקה אם הוא מקיים תנאי מסוים או העברתו כארגומנט לפעולה אחרת). אם מתבצעת השמה למשתנה הרי טיפוס המשתנה המקבל את הערך המוחזר חייב להיות תואם את טיפוס הערך המוחזר.

אם נקבע בכותרת הפעולה שהיא מחזירה ערך, חובה להחזיר ערך במשפט return.

כדי לחשב את משקל המספר, יש לחשב פעמיים מכפלה של מספר דו-ספרתי. פעם אחת את מכפלת שתי הספרות השמאליות של המספר (מאות ועשרות) ופעם שנייה את מכפלת שתי הספרות הימניות של המספר (עשרות ואחדות):

```
//---   פעולה המקבלת מספר דו-ספרתי   ---
//---     ומחזירה את מכפלת ספרותיו   ---
static int MultiplyDigits(int num)
{
      int d1 = num / 10;
      int d2 = num % 10;
      return d1 * d2 ;
}
```

הדגשים

- הפעולה הצהירה במשפט הכותרת כי היא מחזירה מספר שלם. המספר מוחזר בהוראת **return**.
- הפעולה מניחה כי המספר שהתקבל הוא מספר דו-ספרתי, ואין זה מתפקידה לוודא זאת.
תפקידה של התכנית המשתמשת בפעולה לבדוק ולהבטיח את קיומו של התנאי, כלומר לשלוח כפרמטר מספר דו-ספרתי תקין.

משפטי זימון אפשריים לפעולה MultiplyDigits()

השמת הערך המוחזר במשתנה:

☑
```
int mult = MultiplyDigits(37);
```

שילוב הערך המוחזר בבדיקת תנאי לוגי:

☑
```
if (MultiplyDigits(num) > x) …(הוראה)…;
```

הדפסת הערך המוחזר מהפעולה:

☑
```
Console.Write("Multiply Digits : " + MultiplyDigits(2*b + 3));
```

שליחת הערך המוחזר מהפעולה כפרמטר לפעולה אחרת:

☑
```
double x = Math.Sqrt(MultiplyDigits(m));
```

שים ♥ : הארגומנטים (הערכים הנשלחים) לפעולה יכולים להיות קבועים מספריים, שמות של משתנים, ביטויים חישוביים או הערך המוחזר מפעולה אחרת. תחילה יחושב ערך הביטוי או הפעולה שבסוגריים ורק אחר כך תתבצע הפעולה.

"משקלו" של מספר מוגדר כסכום המכפלות של שתי הספרות השמאליות ושל שתי הספרות הימניות במספר תלת-ספרתי, ולכן נוכל להגדיר את הפעולה באופן הבא:

```
static int NumWeight(int num)
{
      int left = MultiplyDigits(num / 10);
      int right = MultiplyDigits(num % 100);
      return left + right;
}
```

או באופן הבא:

```
static int NumWeight(int num)
{
      return MultiplyDigits(num / 10) + MultiplyDigits(num % 100);
}
```

התכנית אמורה לקלוט סדרה של מספרים תלת-ספרתיים ולחשב את משקלם. כדי להיות בטוחים שהמספרים שייקלטו יהיו מספרים תלת-ספרתיים, ניצור מסננת קלט שתחזיר מספרים מתאימים.

מסננת קלט היא קטע קוד לקליטת ערך העונה לקריטריון מסוים. הדרישה שלנו היא לקליטת מספר תלת-ספרתי חיובי, כלומר – מספר שנמצא בין 100 ל-999. מכאן שמספר שאינו עונה לקריטריון הוא מספר הקטן מ-100 או מספר הגדול מ-999.

האלגוריתם למסננת הקלט

1. *קלוט מספר בתוך המשתנה* num.
2. *כל עוד* num *קטן מ-100 או* num *גדול מ-999 בצע:*
   2.1 *קלוט מספר חדש בתוך המשתנה* num

נכתוב פעולה בשם GetNum שתחזיר מספר תלת-ספרתי כנדרש :

```
static int GetNum()
{
      int num;

      Console.Write("type a positive 3 digits number --> ");
      num = int.Parse(Console.ReadLine());
      while (num < 100  ||  num > 999)
      {
            Console.WriteLine("  Error !  ");
            Console.Write("type a positive 3 digits number --> ");
            num = int.Parse(Console.ReadLine());
      }
      return num;
}
```

הערה : בתכנית הלימודים לא נדרשת מסננת קלט, אלא אם נכתב במפורש שחובה לבדוק את תקינות הקלט.

התכנית המלאה המחשבת את סכום המשקלים

```
/**
 *     "משקל" של מספר - סכום מכפלת שתי הספרות הראשונות
 *         ומכפלת שתי הספרות האחרונות
 *       א. פעולה המקבלת מספר תלת ספרתי ומחזירה את משקלו
 *        ב. תכנית הקולטת מספרים תלת-ספרתיים ומדפיסה את
 *            סכום המשקלים עד שיתקבל סכום גדול מ-100
 *
 */
public class T9_2000
{
      //---   פעולה המקבלת מספר דו-ספרתי  ---
      //---     ומחזירה את מכפלת ספרותיו  ---
      static int MultiplyDigits(int num)
      {
            int d1 = num / 10;
            int d2 = num % 10;
            return d1 * d2 ;
      }

      //---     פעולה המקבלת מספר תלת-ספרתי  ---
      //---  ומחזירה את "משקלו" עפ"י ההגדרה  ---
      static int NumWeight(int num)
      {
            return MultiplyDigits(num / 10) + MultiplyDigits(num % 100);
      }

      //---  פעולה המחזירה מספר תלת-ספרתי חיובי  ---
      static int GetNum()
      {
            int num;

            Console.Write("type a positive 3 digits number --> ");
            num = int.Parse(Console.ReadLine());
            while (num < 100  ||  num > 999)
            {
                  Console.WriteLine("  Error !  ");
                  Console.Write("type a positive 3 digits number -->");
```

```
            num = int.Parse(Console.ReadLine());
        }
        return num;
    }


    //--- התכנית הראשית ---
    static void Main()
    {
        int num;
        sum = 0;

        while (sum <= 100)
        {
            num = GetNum();
            sum = sum + NumWeight (num);
        }
        Console.WriteLine("total weight: " + sum);
    }

}
```

סוף פתרון בעיה 3

4. פעולות על מערכים

בעיה 4 – שאלה 7 בגרות 2000

מטרת פתרון הבעיה: הדגמת השימוש בפעולות על מערכים.

> נתון מערך חד-ממדי בגודל 80 המכיל מספרים. כתוב תכנית שתקלוט נתון למשתנה k. התכנית תבדוק אם סכום k האיברים הראשונים במערך גדול מסכום שאר איברי המערך.
> אם כן – תדפיס את סכום k האיברים הראשונים במערך.
>
> התכנית תבדוק את תקינות הקלט k שיתאים לתנאי הבעיה.

מערך הוא עצם. הכרזה על מערך בתכנית יוצרת הפנייה לעצם מסוג מערך. בניית מחלקה ובה תכונה מסוג עצם חורגת מנושאי תכנית הלימודים, ולפיכך יופעלו על המערך פעולות סטטיות.

פעולה המקבלת מערך כפרמטר

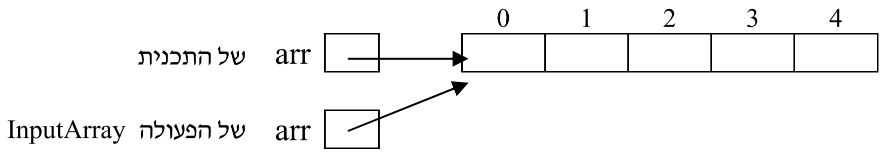
כאשר אנו שולחים מערך כפרמטר לפעולה, אנו בעצם שולחים את ההפניה למערך עצמו, ולכן, כל שינוי במערך בתוך הפעולה משפיע גם על המערך שבתכנית המזמנת את הפעולה.

פעולה הקולטת ערכים לתוך מערך

הפעולה מקבלת מערך כפרמטר וקולטת ערכים לאיבריו. מכיוון שמועברת הפנייה למערך עצמו, כל שינוי בערכי המערך בתוך הפעולה ישנה את ערכי המערך שהפנייתו הועברה כפרמטר לפעולה. לכן אין החזרת ערך מהפעולה ולפיכך אין בפעולה משפט return.

```
static void InputArray(int[] arr)
{
      for (int i = 0 ; i < arr.Length ; i++)
      {
            Console.Write("type number " + i + " --> ");
            arr[i] = int.Parse(Console.ReadLine());
      }
}
```

האיור הבא ממחיש את דרך העברת המערך כפרמטר לפעולה:



פעולה המחזירה ערך מסוג מערך

את המערך נוכל להגדיר ב-main אך נוכל גם לכתוב פעולה שתקלוט מהמשתמש את גודל המערך, תיצור מערך חדש ותקלוט בתוכו ערכים בלולאה פנימית או באמצעות הפעולה InputArray ולבסוף תחזיר את ההפניה למערך זה:

```
//--- פעולה המחזירה מערך מאותחל ומלא בערכים ---
static int[] GetArray()
{
      Console.Write(" Type array size --> ");
      int n = int.Parse(Console.ReadLine());

      int [] arr = new int[n];

      for (int i = 0 ; i < arr.Length ; i++)
      {
            Console.Write("type number " + i + " --> ");
            arr[i] = int.Parse(Console.ReadLine());
      }

      return arr;
}
```

אפשר להמיר קטע זה
במשפט הזימון:
`InputArray(arr);`

משפט הזימון לפעולה, מתוך התכנית, יהיה: `int [] arr = GetArray();`

הערה: שאלות בגרות רבות, העוסקות במערכים, מניחות כי המערך נתון, ולכן אין צורך בפעולות הקלט. בפתרון הבחינה נוכל להסתפק בכתיבת השורה: `int [] arr = GetArray();` המגדירה את המערך הנתון וקובעת את שמו בתכנית. מובן שאם נרצה להריץ את התכנית, נצטרך לממש את הפעולה GetArray.

האלגוריתם לפתרון הבעיה

(1) הגדר את arr כמערך של שלמים בגודל 80 וקלוט בתוכו מספרים שלמים

(2) קלוט ב-k מספר שלם בין 0 לגודל המערך (כולל)

(3) חשב ב-sum1 את סכום k האיברים הראשונים במערך

(4) חשב ב-sum2 את סכום האיברים ממקום k ועד סוף המערך

(5) אם הערך שב-sum1 גדול מהערך שב-sum2
הדפס את sum1

הערה:

- יכולנו להחליף את שורות 5-3 באלגוריתם בהוראה:

(3) אם סכום k האיברים הראשונים גדול מסכום שאר האיברים
הדפס את סכום k האיברים הראשונים

דבר שהיה אולי קריא יותר, אבל פחות יעיל שכן היינו צריכים להפעיל את ההוראה המחשבת את סכום k האיברים הראשונים פעמיים.

```
public class T7_2000
{
	//---  פעולה המחזירה מספר שלם וחיובי ---
	//---        בתחום איברי המערך (0..n) ---
	static int GetNum(int n)
	{
		int num;

		do {
			Console.Write("type a positive int between 0 and "
			                   + n + " --> ");
			num = int.Parse(Console.ReadLine());
		} while (num < 0 || num > n);

		return num;
	}

	//--- פעולה המחזירה מערך מאותחל ומלא בערכים ---
	static int[] GetArr()
	{
		Console.Write(" type array size --> ");
		int n = int.Parse(Console.ReadLine());

		int [] arr = new int[n];
		for (int i = 0 ; i < arr.Length ; i++)
		{
			Console.Write("type number " + i + " --> ");
			arr[i] = int.Parse(Console.ReadLine());
		}

		return arr;
	}

	//---  פעולה המחזירה את סכום איברי המערך החל ממקום from  ---
	//---                         ועד המקום to (לא כולל)  ---
	static int ArrSum(int [] arr, int from, int to)
	{
		int sum = 0;

		for (int i = from ; i < to ; i++)
			sum = sum + arr[i];
		return sum;
	}

	public static void Main()
	{
		int [] arr = GetArr();
		int k = GetNum(arr.Length);

		int sum1 = ArrSum(arr, 0, k);
		int sum2 = ArrSum(arr, k, arr.Length);

		if (sum1 > sum2)
			Console.WriteLine("sum of first elements is: " + sum1);
	}
}
```

סוף פתרון בעיה 4

.5 מחלקת שירות

לאחר שהרצנו כמה תכניות המטפלות במערכים, גילינו שיש פעולות החוזרות על עצמן בכל תכנית. למשל – יצירת מערך, קלט למערך, הדפסת מערך, חיפוש במערך, סכום האיברים המערך וכדו׳.

פתרון אחד לתרגילים אלו יהיה לכתוב את הפעולות החוזרות הדרושות בכל תכנית ותכנית. הדרך הנכונה יותר תהיה לקבץ את כל הפעולות האלה בתוך מחלקה מיוחדת, שתספק את כל השירותים למערך, ושילובה של המחלקה בתכנית.

מחלקת שירות היא מחלקה המכילה אוסף של פעולות, שניתן להשתמש בהן ממחלקות אחרות. אוסף הפעולות במחלקת שירות מסוימת עוסק בדרך כלל באותו נושא. למשל - הכרנו את מחלקת השירות Math המכילה אוסף של פעולות המבצעות חישובים מתמטיים. נוכל לכתוב מחלקת שירות משלנו בשם Array שתספק פעולות המטפלות במערך חד-ממדי, או מחלקת שירות בשם Matrix המספקת פעולות המטפלות במערך דו-ממדי (מטריצה).

היתרון שביצירת מחלקת שירות, מעבר לחיסכון בכתיבה חוזרת של אותו קוד, הוא בכך שאנו משתמשים בפעולות שנבדקו בעבר וידוע שהן פועלות כהלכה.

מחלקת השירות נכתבת כמחלקה נפרדת. כדי שהתכנית תוכל להשתמש בפעולות של מחלקת השירות, יש לספק למהדר את מיקומה. בשלב זה נדאג שמחלקת השירות תישמר באותה תיקייה שנמצאת בה מחלקת התכנית.

בתוך המחלקה נגדיר את התכונות הדרושות למחלקה – אובייקט הקלט, ואת אוסף הפעולות המטפלות במערך. מכיוון שהפעולות תהיינה במחלקה Array ונשתמש בהן מתוך מחלקה אחרת, נקפיד להוסיף לכל פעולה הרשאת גישה **public**.

יצירת מחלקת שירות לטיפול במערך חד-ממדי

```
public class Array
{

      //---   פעולה הקולטת מספרים לתוך מערך של שלמים  ---
      public static void InputArray(int [] arr)
      {
            for (int i = 0 ; i < arr.Length ; i++)
            {
                  Console.Write("type number " + i + " --> ");
                  arr[i] = int.Parse(Console.ReadLine());
            }
      }

      //---   פעולה הקולטת מספרים לתוך מערך של ממשיים  ---
      public static void InputArray(double [] arr)
      {
            for (int i = 0 ; i < arr.Length ; i++)
            {
                  Console.Write("type number " + i + " --> ");
                  arr[i] = double.Parse(Console.ReadLine());
            }
      }
```

```
        //---  פעולה המקבלת מערך של מספרים שלמים  ---
        //---  ומדפיסה את ערכיו בשורה  ---
        public static void PrintArray(int [] arr)
        {
              for (int i = 0 ; i < arr.Length ; i++)
                    Console.Write(arr[i] + "  ");
              Console.WriteLine ();

        }

        //---  פעולה המקבלת מערך של מספרים ממשיים  ---
        //---  ומדפיסה את ערכיו בשורה  ---
        public static void PrintArray (double [] arr)
        {
              for (int i = 0 ; i < arr.Length ; i++)
                    Console.Write(arr[i] + "  ");
              Console.WriteLine();
        }
}
```

נוכל להוסיף פעולות נוספות לפי הצורך.

העמסת פעולות

במחלקה הוגדרו שתי פעולות בשם `InputArray` ושתי פעולות בשם `PrintArray`. מצב שבו יש כמה פעולות באותו שם, הנבדלות זו מזו בשורת הפרמטרים, נקרא **העמסת פעולות** (method overloading). פעולה אחת מקבלת כפרמטר מערך של מספרים שלמים, והפעולה האחרת מקבלת מערך של מספרים ממשיים. המהדר ידע לזהות את הפעולה המתאימה לפי סוג המערך המועבר.

תכנית המשתמשת בפעולות המוגדרות במחלקת השירות Array

```
public class ArrCheck
{
      public static void Main()
      {
            int [] intArr = new int [5];
            double [] doubleArr = new double [7];

            Array.InputArray(intArr);
            Array.InputArray(doubleArr);

            Array.PrintArray(intArr);
            Array.PrintArray(doubleArr);
      }

}
```

כדי שהמהדר ידע שמדובר בפעולות של מחלקת השרות, נוסיף לכל משפט זימון את שם המחלקה שבה מוגדרת הפעולה, בדיוק כפי שעשינו כשהשתמשנו בפעולות של המחלקה Math:

שם-המחלקה.שם-הפעולה();

כדי להציג את איברי המערך, נציין שהפעולה `PrintArray` נמצאת במחלקה Array. שליחת המערך `intArr` כפרמטר לפעולה, תגרום לבחירת הפעולה המקבלת מערך של int .

תבניות – פרק 3

החלפת ערכים בין שני משתנים

נתבונן בבעיה הבאה:

> בחנות למוצרי חשמל הוחלפו בטעות מחיריהם של הדיסקמן והווקמן המוצעים למכירה במחיר מבצע. מחירו האמיתי של הווקמן נשמר במחשב החנות במשתנה diskman ומחיר הדיסקמן נשמר במשתנה walkman. עזרו לבעל החנות לתקן את הטעות על ידי השלמת האלגוריתם:
>
> 1. *השם ב-_______ את ערכו של _______*
> 2. *השם ב-walkman את ערכו של _______*
> 3. *השם ב-_______ את ערכו של _______*

בבעיה זו יש להחליף את ערכיהם של diskman ושל walkman. כדי להחליף בין ערכי המשתנים נצטרך להשתמש במשתנה עזר temp, שישמור את ערכו ההתחלתי של walkman, ולכן האלגוריתם ייראה כך:

1. *השם ב-temp את ערכו של walkman*
2. *השם ב-walkman את ערכו של diskman*
3. *השם ב-diskman את ערכו של temp*

בפתרון בעיה זו השתמשנו **בתבנית** של החלפה בין שני ערכים, בדומה לאלגוריתם שבפתרון בעיה 5 בפרק 3. נתבונן בשני האלגוריתמים הללו:

1. *השם ב-temp את ערכו של a*	1. *השם ב-temp את ערכו של walkman*
2. *השם ב-a את ערכו של b*	2. *השם ב-walkman את ערכו של diskman*
3. *השם ב-b את ערכו של temp*	3. *השם ב-diskman את ערכו של temp*

נשים לב, כי אם נקביל את המשתנים a ו-b למשתנים walkman ו-diskman, בהתאמה, נקבל שני אלגוריתמים זהים. תבנית זו, ***החלפת ערכים בין שני משתנים*** מופיעה באלגוריתמים רבים ולרוב משמשת כתבנית בסיס של פעולות סידור ערכים, למשל עבור מיון ערכים בסדרה.

לתבנית זו של ***החלפת ערכים בין שני משתנים*** ולכל התבניות שתוגדרנה בהמשך יש כמה מרכיבים המאפיינים אותן.

נגדיר באופן כללי את **מאפייניה של תבנית** עם הסבר קצר לכל מאפיין:

שם התבנית: השם מבטא בצורה תמציתית את המשימה לביצוע או את דרך ביצועה.

נקודת מוצא: נקודת המוצא מציינת את המצב התחילי הנתון של המשימה לביצוע, כלומר: שמות המשתנים ובהקשרים מסוימים גם טיפוסיהם. התבניות מתפתחות עם ההתקדמות בחומר הלימוד ולכן ייתכן כי לתבנית אחת תהיינה נקודות מוצא שונות, למשל, פעם נקודת המוצא תהיה שני מספרים, ופעם אחרת שלושה מספרים.

מטרה: המטרה מתארת את המצב הסופי, הפלט הדרוש או ערך שיש להחזיר עם תום הביצוע.

אלגוריתם: האלגוריתם מתאר מתכונת לביצוע המשימה. האלגוריתם הוא לב התבנית. לעיתים בתוך האלגוריתם יהיה שימוש בתבנית אחרת.

יישום ב-#C: יישום האלגוריתם בשפת #C.

נציג את התבנית ***החלפת ערכים בין שני משתנים***, על פי המאפיינים שהכרנו:

שם התבנית: החלפת ערכים בין שני משתנים

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: החלפת הערכים ההתחלתיים בין שני המשתנים

אלגוריתם:

1. *השם ב-*temp *את ערכו של* element1
2. *השם ב-*element1 *את ערכו של* element2
3. *השם ב-*element2 *את ערכו של* temp

יישום ב-#C:

```
temp = element1;
element1 = element2;
element2 = temp;
```

שימו ♥: ביישום התבנית נעשה שימוש במשתנה עזר temp כדי להבטיח שלא יאבד אף אחד מהערכים ההתחלתיים של שני המשתנים.

בעל החנות של מוצרי החשמל יכול להשתמש בתבנית שתיארנו. נוכל לכתוב

החלף את ערכי המשתנים diskman ו-walkman

מאחר שזו תבנית מוכרת לנו, הרי השימוש בתבנית כהוראה באלגוריתם (במקרה זה, אלגוריתם בן שורה אחת), מסביר מהן הפעולות שיש לבצע. האלגוריתם שנכתב הוא קצר יותר, אבל גם ברור יותר. אמנם הוראה אחת בו מייצגת כעת כמה הוראות, אבל היא מסבירה היטב את תפקידן של אותן הוראות.

שאלה 1

ישמו בשפת C# את השימוש של בעל החנות למוצרי חשמל בתבנית.

שאלה 2

איתמר ויאיר משחקים במשחק הקלפים "טאקי". בתחילת המשחק קיבל כל אחד מהם מספר קלפים זהה. המנצח במשחק הוא השחקן שאין בידו קלפים. במהלך המשחק, כאשר איתמר הבחין כי מספר הקלפים שבידו גדול בהרבה ממספר הקלפים שבידי יאיר החליט להשתמש בקלף "החלף קלפים", שמשמעותו החלפת הקלפים בין שני השחקנים.

נתון אלגוריתם, שהקלט שלו הוא מספר הקלפים שיש לאיתמר וליאיר לפני ביצוע ההחלפה והפלט שלו הוא מספר הקלפים של כל אחד מהם לאחר ביצוע ההחלפה וכן מספר הקלפים שאיתמר הצליח להיפטר במסגרת ההחלפה:

1. קלוט מספר קלפים של איתמר ב-itamar
2. קלוט מספר קלפים של יאיר ב-yair
3. השם ב-temp את ערכו של itamar
4. השם ב-itamar את ערכו של yair
5. השם ב-yair את ערכו של temp
6. הצג כפלט את ההודעה "מספר הקלפים של איתמר לאחר ההחלפה" ואת ערכו של itamar
7. הצג כפלט את ההודעה "מספר הקלפים של יאיר לאחר ההחלפה" ואת ערכו של yair
8. השם ב-diff את ערכו של הביטוי החשבוני yair - itamar
9. הצג כפלט את ההודעה "מספר הקלפים שמהם איתמר הצליח להיפטר במסגרת ההחלפה" ואת ערכו של diff

א. ציינו מהן השורות באלגוריתם המממשות את התבנית ***החלפת ערכים בין שני משתנים***.

ב. מהי נקודת המוצא של התבנית בשימוש זה?

ג. כתבו אלגוריתם שקול לאלגוריתם הנתון תוך שימוש בתבנית ***החלפת ערכים בין שני משתנים***.

ד. כתבו אלגוריתם שקול לאלגוריתם הנתון המבצע אותה מטרה **ללא** שימוש בתבנית ***החלפת ערכים בין שני משתנים***.

שאלה 3

דרך נוספת להחליף ערכים בין שני משתנים היא להשים בתחילה את ערכו של element2 ב-temp. השלימו את קטע התוכנית המתאים להצעה זאת:

```
temp = element2;
_______________
_______________
```

היפוך סדר האיברים בסדרה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 1: כתבו אלגוריתם, שהקלט שלו הוא שני מספרים שלמים הנקלטים למשתנים big ו-small, כך שהמספר הגדול נקלט ל-small והמספר הקטן נקלט ל-big. על האלגוריתם להפוך את ערכי המשתנים, כך שבמשתנה big יישמר הערך הגדול ובמשתנה small יישמר הערך הקטן, ולהציג כפלט את הערכים לאחר ההיפוך.

בעיה 2: כל הספרים בספריה העירונית מקוטלגים על פי מספרים קטלוגיים. על מדף כלשהו בספריה מסודרים כל הספרים בסדר עולה על פי מספריהם הקטלוגיים פרט לשני ספרים שהחליפו בטעות את מקומם זה בזה. כתבו אלגוריתם, שמטרתו לדאוג לכך שכל הספרים במדף יהיו מסודרים בסדר עולה על פי מספריהם הקטלוגיים של הספרים. הקלט של האלגוריתם הוא המקומות של הספרים שאינם מונחים במקומותיהם ומספריהם הקטלוגיים, בהתאמה. על האלגוריתם להפוך את סדר הספרים ולהציג כפלט את המקומות והמספרים הקטלוגיים של הספרים לאחר ההיפוך.

אנו רואים כי בשתי הבעיות האלגוריתמיות עלינו להפוך סדר של שני ערכים. זהו ליבה של התבנית ***היפוך סדר האיברים בסדרה***, במקרה זה של סדרה בת שני איברים. היפוך סדר האיברים בסדרה שימושי בהקשרים בהם הסדרה נתונה בסדר כלשהו ויש צורך להפכו. למשל, כאשר מעוניינים להפוך סדרה המסודרת בסדר עולה לאותה סדרה המסודרת בסדר יורד.

שימו ♥: עבור סדרה בת שני איברים, האלגוריתמים של התבנית ***החלפת ערכים בין שני משתנים*** ושל התבנית ***היפוך סדר האיברים בסדרה*** הם זהים, כי כדי להפוך סדר של שני ערכים בלבד יש למעשה להחליף בין שני ערכי המשתנים. חשוב לציין שזהו מקרה פרטי, ולפעולת ההיפוך יש משמעות עבור סדרה שבה יותר משני איברים. בהמשך נראה דוגמאות להיפוך סדרה שבה לפחות 3 איברים.

נגדיר את מאפייני התבנית:

שם התבנית: היפוך סדר האיברים בסדרה

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: היפוך הערכים בין שני המשתנים

אלגוריתם:

החלף את ערכי element1 ו-element2

הנה הפתרונות של שתי הבעיות האלגוריתמיות:

פתרון בעיה 1	פתרון בעיה 2
*1. קלוט מספר גדול ל-*small *2. קלוט מספר קטן ל-*big *3. הפוך את סדר האיברים בסדרה* small, big *4. הצג כפלט "הערך הגדול",* big *5. הצג כפלט "הערך הקטן",* small	*1. קלוט מקום ספר ראשון ב-*place1 *2. קלוט מספר קטלוגי של ספר ראשון ב-*num1 *3. קלוט מקום ספר שני ב-*place2 *4. קלוט מספר קטלוגי של ספר שני ב-*num2 *5. הפוך את סדר האיברים בסדרה* num1, num2 *6. הצג כפלט "מיקום הספר",* num1*, "ב-",* place1 *7. הצג כפלט "מיקום הספר",* num2*, "ב-",* place2

שאלה 4

ישמו כל אחד מן האלגוריתמים כקטע תוכנית בשפת C#.

שימו לב כי כדי ליישם את התבנית ***היפוך סדר האיברים בסדרה*** עליכם להשתמש ביישום של התבנית ***החלפת ערכים בין שני משתנים***.

שאלה 5

נתונים 3 ערכים במשתנים element1, element2 ו-element3. המורה ביקשה מהתלמידים להציע אלגוריתם עבור היפוך סדר הערכים בסדרת המשתנים.

אוהד הציע את האלגוריתם הבא:

1. הפוך את סדר האיברים בסדרה element1, element3

2. הפוך את סדר האיברים בסדרה element3, element1

האלגוריתם של אוהד שגוי.

א. תנו דוגמה לערכים ב-element1, element2 ו-element3 שעבורה ניתן לראות כי האלגוריתם שגוי.

ב. תנו שתי דוגמאות שונות לערכים ב-element1, element2 ו-element3, שעבורן לא ניתן לראות כי האלגוריתם שגוי. רשמו מהו המאפיין של כל אחת מהדוגמאות.

ג. הסבירו במלים מדוע האלגוריתם שאוהד הציע שגוי.

ד. תקנו את האלגוריתם.

ה. ישמו את האלגוריתם כקטע תוכנית בשפת C#.

ממוצע של סדרת מספרים

נתבונן בשלוש הבעיות האלגוריתמיות הבאות:

בעיה 1: ציון שנתי של מקצוע בתעודה נקבע על פי הממוצע של הציונים במחצית א' ובמחצית ב'. כתבו אלגוריתם, שהקלט שלו הוא ציוניו של אלי במחצית א' ובמחצית ב' במקצוע מדעי המחשב והפלט שלו הוא הציון השנתי של אלי במדעי המחשב.

בעיה 2: הציון הבית-ספרי של תלמיד נקבע על פי הממוצע של הציון השנתי וציון המתכונת. כתבו אלגוריתם, שהקלט שלו הוא ציונו של אלי בבחינת המתכונת במדעי המחשב והפלט שלו הוא ציונו הבית-ספרי של אלי במדעי המחשב.

בעיה 3: ציון סופי של תלמיד במקצוע נקבע על פי הממוצע של הציון הבית-ספרי והציון בבחינת הבגרות. כתבו אלגוריתם, שהקלט שלו הוא ציונו של אלי בבחינת הבגרות במדעי המחשב והפלט שלו הוא ציונו הסופי של אלי במדעי המחשב.

בשלוש הבעיות האלגוריתמיות יש שימוש בתבנית ממוצע. ממוצע הוא מדד סטטיסטי וחישובו הוא אחד החישובים הבסיסיים עבור סדרת ערכים מספריים. כדי לחשב ממוצע של סדרה יש לחשב תחילה את הסכום הכולל של הסדרה ולאחר מכן לחלק במספר הערכים בסדרה.

נגדיר את מאפייני התבנית ***ממוצע של סדרת מספרים***, עבור סדרה בת שני מספרים:

שם התבנית: ממוצע של סדרת מספרים

נקודת מוצא: שני מספרים ב-num1 ו-num2

מטרה: חישוב הממוצע של שני המספרים

אלגוריתם:

השם ב-sum את ערכו של הביטוי החשבוני num1 + num2

השם ב-average את ערכו של הביטוי החשבוני sum / 2

יישום ב-#C:

```
sum = num1 + num2;
average = (double) sum / 2;
```

נציג עתה את הפתרון של שלוש הבעיות האלגוריתמיות ברצף:

1. *קלוט ציון מחצית א' ב-*semester1
2. *קלוט ציון מחצית ב' ב-*semester2

3. חשב ממוצע של סדרת המספרים semester1, semester2 והשם את הערך המחושב ב-yearGrade

4. הצג כפלט "ציון השנתי של אלי במדעי המחשב", yearGrade

5. קלוט ציון בחינת מתכונת ב-finalExam

6. חשב ממוצע של סדרת המספרים yearGrade, finalExam והשם את הערך המחושב ב-schoolGrade

7. הצג כפלט "ציון הבית-ספרי של אלי במדעי המחשב", schoolGrade

8. קלוט ציון בחינת בגרות ב-matriculation

9. חשב ממוצע של סדרת המספרים schoolGrade, matriculation והשם את הערך המחושב ב-finalGrade

10. הצג כפלט "ציון הסופי של אלי במדעי המחשב", finalGrade

שאלה 6

ישמו את האלגוריתם כקטע תוכנית בשפת #C.

שאלה 7

עידן טוען כי ניתן לכתוב את היישום ב-#C עבור התבנית ***ממוצע של סדרת מספרים*** בת שני מספרים ממשיים בהוראה אחת ואין צורך לפצל לשתי הוראות, ולכן הציע את היישום הבא:

```
average = num1 + num2 / 2;
```

חגית טוענת כי היישום שעידן הציע שגוי.

א. תנו דוגמה לערכי num1 ו-num2, עבורה האלגוריתם שעידן הציע נותן פלט נכון.

ב. תנו דוגמה לערכי num1 ו-num2, עבורה ניתן לראות כי טענתה של חגית נכונה.

ג. תקנו את היישום שעידן הציע.

שאלה 8

לפניכם שימוש בתבנית:

חשב ממוצע של סדרת המספרים num1, num2, num3 והצג כפלט את הערך המחושב

א. מהי נקודת המוצא של התבנית?

ב. כתבו את היישום ב-#C עבור השימוש המתואר.

שאלה 9

בפינת החי "צבי הנינג'ה" ישנם 3 צבים. האחראי על פינת החי מעוניין לבצע חישובים סטטיסטיים על התפתחות הצבים. לצורך זה הוא שוקל את הצבים אחת לחודש ורושם את ממוצע משקלי שלושת הצבים.

נתון קטע התוכנית הבא שהקלט שלו הוא משקלי שלושת הצבים בחודש ינואר והפלט שלו אמור להיות ממוצע משקליהם:

```
weight1 = double.Parse(Console.ReadLine());
weight2 = double.Parse(Console.ReadLine());
weight3 = double.Parse(Console.ReadLine());
sum = weight1 + weight2;
average = sum / 2;
sum = average + weight3;
average = sum / 2;
Console.WriteLine(average);
```

קטע התוכנית שגוי.

א. תנו דוגמת קלט שעבורה ניתן לראות כי קטע התוכנית אינו משיג את המטרה.

ב. תנו דוגמת קלט שעבורה קטע התוכנית משיג את המטרה.

ג. בקטע התוכנית ישנו שימוש כפול בתבנית: עבור כל אחד מהשימושים, ציינו את ההוראות המתאימות לו ותארו את השימוש בתבנית.

ד. תקנו את קטע התוכנית.

הזזה מעגלית בסדרה

הזזה מעגלית בסדרה היא תבנית הנחוצה לעיבוד אשר בו יש להזיז באופן אחיד את כל הערכים בסדרה, תוך הקפדה על כך שערכו של אף ערך לא יאבד. ניתן להזיז את ערכי הסדרה שמאלה או ימינה.

הזזה מעגלית שמאלה מתבצעת על ידי שמירת ערכו של המשתנה השמאלי ביותר בסדרה במשתנה זמני ולאחר מכן, השמה של כל ערך של משתנה למשתנה שמשמאלו. כלומר, השמת ערכו של המשתנה השני במשתנה הראשון, השמת ערכו של המשתנה השלישי במשתנה השני, וכך הלאה. לבסוף השמת ערכו של המשתנה הזמני במשתנה האחרון.

הזזה מעגלית ימינה מתבצעת על ידי שמירת ערכו של המשתנה הימני ביותר בסדרה במשתנה זמני ולאחר מכן, השמה של כל ערך של משתנה למשתנה שמימינו. כלומר, השמת ערכו של המשתנה הלפני אחרון במשתנה האחרון, וכך הלאה, ולבסוף השמת ערכו של המשתנה הזמני במשתנה הראשון.

שימו ♥: עבור סדרה בת שני איברים האלגוריתמים של התבנית ***החלפת ערכים בין שני משתנים*** ושל התבנית ***הזזה מעגלית בסדרה*** הם זהים, כי כדי להזיז מעגלית שמאלה או ימינה שני ערכים בלבד יש למעשה להחליף בין שני ערכי המשתנים. חשוב לציין שזהו מקרה פרטי, ולפעולת ההזזה המעגלית יש משמעות עבור סדרה שבה יותר משני איברים. נציג הזזה מעגלית של סדרה שבה 3 איברים, ובהמשך נראה דוגמאות להזזה מעגלית של סדרה שבה לפחות 3 איברים.

נפריד את מאפייני התבנית ***הזזה מעגלית בסדרה*** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית ***הזזה מעגלית שמאלה בסדרה*** ואחר כך נציג את מאפייני התבנית ***הזזה מעגלית ימינה בסדרה***.

שם התבנית: הזזה מעגלית שמאלה בסדרה

נקודת מוצא: שלושה ערכים במשתנים element1, element2 ו-element3

מטרה: הזזה מעגלית שמאלה של שלושת המשתנים

אלגוריתם:

החלף את ערכי המשתנים element1 ו-element2

החלף את ערכי המשתנים element2 ו-element3

שם התבנית: הזזה מעגלית ימינה בסדרה

נקודת מוצא: שלושה ערכים במשתנים element1, element2 ו-element3

מטרה: הזזה מעגלית ימינה של שלושת המשתנים

אלגוריתם:

החלף את ערכי המשתנים element2 ו-element3

החלף את ערכי המשתנים element1 ו-element2

שאלה 10

נתונה סדרה של שלושה ערכים: element1, element2, element3.

לפניכם שימוש בתבנית של הזזה מעגלית שמאלה עבור הסדרה:

הזז מעגלית שמאלה את איברי הסדרה element1, element2, element3

א. כתבו אלגוריתם המתאים לתבנית, שאינו משתמש בתבנית החלפת ערכים, ויישמו אותו על ידי כתיבת קטע תוכנית בשפת C#.

ב. כתבו שימוש בתבנית של ***הזזה מעגלית ימינה*** עבור שלושת הערכים הנתונים, לאחר מכן כתבו אלגוריתם המתאים לתבנית, שאינו משתמש בתבנית החלפת ערכים, ויישמו אותו על ידי כתיבת קטע תוכנית בשפת C#.

שאלה 11

במשחק הכיסאות המוזיקליים משתתפים שלושה ילדים היושבים על שלושה כסאות. בעת הפעלת המוזיקה כל ילד זז מעגלית שמאלה ומתיישב בכסא שמשמאלו.

נתון האלגוריתם הבא שהקלט שלו הוא 3 מספרים שלמים המייצגים את מספרי הילדים, והפלט שלו הוא מספרי הילדים לאחר שתי הזזות מעגליות שמאלה:

1. *קלוט שלושה מספרים שלמים ב-* child1, child2, child3
2. *הזז מעגלית שמאלה את איברי הסדרה* child1, child2, child3
3. *הזז מעגלית שמאלה את איברי הסדרה* child1, child2, child3
4. *הצג כפלט את הערכים של* child1, child2, child3

א. מה יהיה הפלט עבור הקלט 6 3 8?

ב. תנו דוגמת קלט שעבורה הפלט יהיה 9 4 2 .

ג. באלגוריתם נעשה שימוש כפול בתבנית ***הזזה מעגלית שמאלה בסדרה***. כתבו אלגוריתם השקול לאלגוריתם הנתון תוך שימוש **יחיד** בתבנית אחרת.

ד. ישמו את האלגוריתם שכתבתם בסעיף ג׳ כקטע תוכנית בשפת C#.

שאלה 12

ביום ספורט היתולי התחרו 4 קבוצות a, b, c, d בארבע תחנות שונות. נקבע כי המעבר בין התחנות ייעשה בצורה מעגלית ימינה. נתון האלגוריתם החלקי הבא, המתאר את החלפת הקבוצות:

1. *השם ב-temp את ערכו של* d
2. *השם ב-d את ערכו של* ________
3. *השם ב-c את ערכו של* b
4. *השם ב-________ את ערכו של* a
5. *השם ב-________ את ערכו של* ________

א. השלימו את האלגוריתם.

ב. לאחר שכל הקבוצות סיימו את הסיבוב הראשון בתחרות קבעו מארגני יום הספורט שהמעבר בין התחנות בסיבוב השני ייעשה בצורה מעגלית שמאלה. כתבו אלגוריתם שיתאר את החלפת הקבוצות בסיבוב השני.

שאלה 13

נתון האלגוריתם הבא, שהקלט שלו הוא ארבעה מספרים שלמים:

1. *קלוט 4 מספרים שלמים ב-*element1, element2, element3, element4
2. *הזז מעגלית שמאלה את איברי הסדרה* element1, element2, element3
3. *הזז מעגלית ימינה את איברי הסדרה* element1, element3, element4
4. *הצג כפלט את הערכים של* element1, element2, element3, element4

א. מה יהיה הפלט עבור הקלט 2 15 -7 6?

ב. תנו דוגמת קלט שעבורה הפלט יהיה 4 6 19 10.

ג. מהי מטרת האלגוריתם?

ד. כתבו אלגוריתם שקול לאלגוריתם הנתון תוך שימוש **כפול** בתבנית ***היפוך סדר האיברים בסדרה*** בת שני ערכים.

תבניות – פרק 4

חלוקת כמות פריטים לקבוצות בגודל נתון

נתבונן בבעיה הבאה:

> במפעל זכוכית אורזים 8 כוסות בקופסת קרטון. מחיר קופסת קרטון הוא 1.5 ₪. נתון אלגוריתם חלקי שהקלט שלו הוא מספר הכוסות המיועדות לאריזה, והפלט שלו הוא: מספר הקופסאות המלאות (בכוסות) שניתן לארוז, המחיר הכולל של הקרטון הדרוש לאריזה ומספר הכוסות שנותרו בתפזורת. השלימו את האלגוריתם:
>
> 1. *קלוט מספר כוסות המיועדות לאריזה ב-*glasses
> 2. *חשב את* _______ *על ידי* _______ *והשם ב-*boxes
> 3. *חשב את* _______ *על ידי* _______ *והשם ב-*price
> 4. *חשב את* _______ *על ידי* _______ *והשם ב-*left
> 5. *הצג כפלט את הערך* boxes, *את הערך* price *ואת הערך* left

בבעיה זו יש לחשב את המספר המירבי של קופסאות מלאות על ידי חישוב מנת החלוקה של מספר הכוסות ב-8. את חישוב מספר הכוסות שנותרו בתפזורת נבצע באמצעות חישוב שארית החלוקה של מספר הכוסות המיועדות לאריזה ב-8. התבנית של חלוקת כמות פריטים לקבוצות בגודל נתון, באמצעות חישובי מנה ושארית, כפי שמשמשת בפתרון בעיה זו, דומה לחישובים שהוצגו בפתרון בעיה 2 בפרק הלימוד. תבנית זו היא בסיסית ביותר ושימושית בחישובים רבים של מספרים שלמים. התבנית משמשת הן כתבנית עיקרית בבעיות חישוב של מספרים שלמים והן כתבנית המשולבת בחישובים מורכבים שונים.

נתבונן בשני האלגוריתמים הללו:

1. *קלוט מספר כוסות המיועדות לאריזה ב-*glasses	1. *קלוט מספר הבלונים ב-*amount
2. *חשב את מספר הקופסאות המלאות על ידי מנת החלוקה של* glasses *ב-8 והשם את התוצאה ב-*boxes	2. *חשב את מספר השקיות המלאות על ידי מנת החלוקה של* amount *ב-20 והשם את התוצאה ב-*bags
3. *חשב את מחיר הקופסאות על ידי* boxes*1.5 *והשם ב-*price	
4. *חשב את מספר הכוסות שנותרו בתפזורת על ידי שארית החלוקה של* glasses *ב-8 והשם ב-*left	3. *חשב את מספר הבלונים שנותרו פזורים על ידי שארית החלוקה של* amount *ב-20 והשם את התוצאה ב-*remainder
5. *הצג כפלט את הערך* boxes, *את הערך* price *ואת הערך* left	4. *הצג כפלט את הערך* bags *ואת הערך* remainder

אנו רואים כי בשני האלגוריתמים מתבצעת חלוקת כמות פריטים לקבוצות: בבעיה הראשונה כמות הפריטים היא מספר הכוסות המיועדות לאריזה וגודל כל קבוצה (כוסות בקופסת קרטון) הוא 8 ובבעיה השנייה כמות הפריטים היא מספר הגוגואים וגודל כל קבוצה (מספר גוגואים בשקית) הוא 20.

נפריד את מאפייני התבנית ***חלוקת כמות פריטים לקבוצות בגודל נתון*** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית ***מנת החלוקה של כמות פריטים לקבוצות*** ואחר כך נציג את מאפייני התבנית ***שארית החלוקה של כמות פריטים לקבוצות***.

שם התבנית: מנת החלוקה לקבוצות של כמות פריטים

נקודת מוצא: שני מספרים שלמים חיוביים, quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)

מטרה: מספר הקבוצות המלאות מחלוקה של quantity הפריטים לקבוצות בגודל num

אלגוריתם:

השם ב-groups את מנת החלוקה של quantity ב-num

יישום ב-#C:

```
groups = quantity / num;
```

שם התבנית: שארית החלוקה לקבוצות של כמות פריטים

נקודת מוצא: שני מספרים שלמים חיוביים, quantity (כמות הפריטים) ו-num (מספר הפריטים בקבוצה)

מטרה: מספר הפריטים העודף בחלוקה של quantity הפריטים לקבוצות בגודל num

אלגוריתם:

השם ב-remainder את שארית החלוקה של quantity ב-num

יישום ב-#C:

```
remainder = quantity % num;
```

שאלה 1

ישמו את האלגוריתם לפתרון בעיית הכוסות בשפת #C.

שאלה 2

א. מהם הערכים האפשריים עבור מספר הכוסות שעשויות להישאר בתפזורת לאחר החלוקה לקופסאות מלאות?

ב. אחד העובדים במפעל הזכוכית הציע הצעת ייעול לאריזת הכוסות, כדי להקטין את מספר הכוסות שיישארו בתפזורת: חלוקת מספר הכוסות הנותרות לקופסאות קרטון קטנות בגודל 2, שעלות כל אחת מהן היא 0.5 ₪.

1. מהם הערכים האפשריים עבור מספר הכוסות שעשויות להישאר בתפזורת לאחר החלוקה לשני סוגי הקופסאות?
2. הרחיבו את האלגוריתם על פי הצעת הייעול של העובד, כך שהפלט של האלגוריתם יהיה המחיר הכולל של הקרטון עבור שני סוגי הקופסאות ומספר הכוסות שיישארו בתפזורת לאחר החלוקה.
3. ישמו את האלגוריתם שכתבתם בסעיף ב בשפת C#.

פירוק מספר חיובי לספרותיו

התבנית של הפרדת הספרות של מספר, ובפרט הספרה הימנית ביותר – ספרת האחדות, היא שימושית בהקשרים רבים במדעי המחשב, למשל, לצורך סיווג מספרים על פי ספרת האחדות שלהם.

לעת עתה נתמקד בתבנית של פירוק מספר דו-ספרתי חיובי לספרותיו, בהדרגה נרחיב עבור מספר תלת-ספרתי חיובי ובהמשך נראה תבנית כללית עבור מספר חיובי כלשהו. במספר דו-ספרתי ישנן שתי ספרות: הפרדת ספרת האחדות נעשית על ידי חישוב שארית החלוקה של המספר ב-10 והפרדת ספרת העשרות נעשית על ידי חישוב של מנת החלוקה של המספר ב-10.

נפריד את מאפייני התבנית ***פירוק מספר חיובי לספרותיו*** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית ***ספרת האחדות של מספר*** ואחר כך נציג את מאפייני התבנית ***ספרת העשרות של מספר***. כפי שצוין קודם, נקודת המוצא של התבנית היא מספר דו-ספרתי חיובי.

שם התבנית: ספרת האחדות של מספר

נקודת מוצא: מספר דו-ספרתי חיובי num

מטרה: חישוב ספרת האחדות של num

אלגוריתם:

השם ב-units את שארית החלוקה של num ב-10

יישום ב-#C:

```
units = num % 10;
```

שם התבנית: ספרת העשרות של מספר

נקודת מוצא: מספר דו-ספרתי חיובי num

מטרה: חישוב ספרת העשרות של num

אלגוריתם:

השם ב-tens את מנת החלוקה של num ב-10

יישום ב-#C:

```
tens = num / 10;
```

שאלה 3

נתון מספר דו-ספרתי חיובי num ועליו הופעלו שתי התבניות:

1. חשב את ספרת האחדות של num והצג אותה כפלט

2. חשב את ספרת העשרות של num והצג אותה כפלט

א. תנו שתי דוגמאות שונות לערכים אפשריים ל-num שעבורם הערך הראשון שיוצג כפלט הוא 5.

ב. תנו שתי דוגמאות שונות לערכים אפשריים ל-num שעבורם הערך השני שיוצג כפלט הוא 8.

ג. רשמו את כל זוגות הערכים שעשויים להתקבל בפלט אם ידוע שהמספר num גדול מ-40, מתחלק ב-5 ללא שארית ואינו זוגי.

ד. רשמו את כל הערכים האפשריים ל-num כך שעבור סדרת ההוראות הבאה יוצג הערך 3:

1. חשב את ספרת האחדות של num

2. חשב את ספרת העשרות של num

3. הצג כפלט את סכום הערכים שחושבו

ה. נתונה סדרת ההוראות הבאה:

1. חשב את ספרת האחדות של num

2. חשב את ספרת העשרות של num

3. הצג כפלט את הערך המוחלט של הפרש הערכים שחושבו

1. רשמו את כל הערכים האפשריים ל-num עבורם יוצג כפלט הערך 1.

2. ישמו את ההוראה בשפת C#.

שאלה 4

עינת הציעה ליישם את התבנית ***פירוק מספר לספרותיו*** עבור מספר דו-ספרתי חיובי num באמצעות התבנית ***חלוקת כמות פריטים לקבוצות בגודל נתון***.

א. מהי כמות הפריטים? הסבירו.

ב. מהו גודל הקבוצות? הסבירו.

ג. השלימו את השימוש בתבניות על פי הצעתה של עינת:

1. חשב את שארית החלוקה של ______ פריטים ל-______ קבוצות והצג אותה כפלט

2. חשב את מנת החלוקה של ______ פריטים ל-______ קבוצות והצג אותה כפלט

שאלה 5

המורה ביקשה מתלמידיה לכתוב אלגוריתם שהקלט שלו הוא מספר תלת-ספרתי num והפלט שלו הוא שלוש ספרותיו של num, על פי הרעיון הבא, המתבסס על פירוק המספר לשני מספרים: ספרת האחדות, ומנת החלוקה של num ב-10. לאחר מכן יש לבצע פירוק נוסף של המנה

שחושבה, כמספר דו-ספרתי.

א. השלימו את האלגוריתם, ונסו להשתמש לשם כך בתבניות.

1. *קלוט מספר תלת-ספרתי ב-*num
2. *חשב את ספרת האחדות על ידי* ________ *והשם ב-*units
3. *חשב את המספר הדו-ספרתי המתקבל מהסרת ספרת האחדות על ידי* ________ *והשם ב-*doubleDigit
4. *חשב את ספרת העשרות של* num *על ידי* ________ *והשם ב-*tens
5. *חשב את ספרת המאות של* num *על ידי* ________ *והשם ב-*hundreds
6. *הצג כפלט את הערך* units*, את הערך* tens *ואת הערך* hundreds

ב. ישמו את האלגוריתם בשפת C#.

ג. אלון הציע דרך אחרת לפירוק המספר. לפניכם האלגוריתם:

1. *קלוט מספר תלת-ספרתי ב-*num
2. *חשב את ספרת המאות של* num *על ידי חישוב* ***מנת החלוקה של*** num ***פריטים ל-***100 ***קבוצות*** *והשם ב-*hundreds
3. *חשב את המספר הדו-ספרתי המתקבל מהסרת ספרת המאות על ידי חישוב* ***שארית החלוקה של*** num ***פריטים ל-***100 ***קבוצות*** *והשם ב-*doubleDigit
4. *חשב את ספרת העשרות של* num *על ידי חישוב* ***ספרת העשרות של*** doubleDigit *והשם ב-*tens
5. *חשב את ספרת האחדות של* num *על ידי חישוב* ***ספרת האחדות של*** doubleDigit *והשם ב-*units
6. *הצג כפלט את הערך* units*, את הערך* tens *ואת הערך* hundreds

ד. הסבירו את הרעיון לפתרון עליו מתבסס אלון. בהסבר התייחסו לשימוש בתבנית ***חלוקת כמות פריטים לקבוצות בגודל נתון***.

ה. ישמו את האלגוריתם בשפת C#.

שאלה 6

בבניין משרדים בן 9 קומות מסומן כל חדר באמצעות קוד של מספר תלת-ספרתי: ספרת המאות מציינת את מספר הקומה בה נמצא החדר וספרות האחדות והעשרות מציינות את מספר החדר בקומה.

א. כתבו אלגוריתם, שהקלט שלו הוא קוד חדר והפלט שלו הוא מספר הקומה בה נמצא החדר ומספר החדר בקומה.

ב. ציינו באילו תבניות השתמשתם לפתרון הבעיה.

ג. ישמו את האלגוריתם בשפת C#.

בניית מספר

התבנית של בניית מספר היא שימושית בהקשרים רבים. אולם, הבנייה של מרכיב ממספר מרכיבים מצומצמים יותר שימושית גם בהקשרים אחרים במדעי המחשב, למשל: בניית מילה מאותיות בודדות, בניית משפט ממילים בודדות, בניית צורה גרפית מקווים בודדים וכדומה.

לעת עתה נתמקד בתבנית של הרכבת מספר דו-ספרתי משתי ספרות, בהדרגה נרחיב עבור בניית מספר תלת-ספרתי ובהמשך נראה תבנית כללית עבור מספר חיובי כלשהו.

כדי להרכיב מספר משתי ספרות נכפיל את הספרה המיועדת להיות ספרת העשרות ב-10 ונחבר לתוצאה את הספרה המיועדת להיות ספרת האחדות.

שם התבנית: בניית מספר

נקודת מוצא: שתי ספרות left ו-right

מטרה: בניית מספר דו-ספרתי מהספרות הנתונות

אלגוריתם:

השם ב-num את הערך של הביטוי החשבוני left * 10 + right

יישום ב-#C:

```
num = left * 10 + right;
```

שימו ♥: כאשר אנו משתמשים בתבנית ***בניית מספר*** חשוב להקפיד על תיאור חד-משמעי: נסכים כי הספרה הראשונה שנציין תהיה ספרת העשרות והשנייה ספרת האחדות. למשל, הכוונה בשימוש *בנה מספר מ-1 ו-3* היא לבנות את המספר 13 (ולא את המספר 31). בדומה, בשימוש בתבנית לשלוש ספרות נציין קודם את ספרת המאות, אחר כך את ספרת העשרות ולבסוף את ספרת האחדות.

שאלה 7

נתון אלגוריתם חלקי, שהקלט שלו הוא מספר דו-ספרתי חיובי num, והפלט שלו הוא מספר דו-ספרתי חדש, שערך ספרת האחדות בו גדולה ב-1 מספרת האחדות של המספר הנתון num, וערך ספרת העשרות בו קטנה ב-1 מספרת העשרות של המספר הנתון num.

הניחו שבמספר הנקלט ספרת האחדות שונה מ-9 וספרת העשרות שונה מ-0.

א. השלימו את האלגוריתם:

1. *קלוט מספר דו-ספרתי חיובי ב-*num
2. *חשב את ספרת האחדות על ידי* ________ *והשם ב-*units
3. *חשב את ספרת העשרות של* num *על ידי* ________ *והשם ב-*tens

4. חשב את ערכה של ספרת האחדות של המספר החדש על ידי ________ והשם ב-newUnits

5. חשב את ערכה של ספרת העשרות של המספר החדש על ידי ________ והשם ב-newTens

6. בנה מספר מ-________ ו-________ והצג כפלט את הערך שהתקבל

א. ציינו מהן התבניות שהשתמשתם בהן בהשלמת האלגוריתם.

ב. מהו הפלט עבור הקלט 67?

ג. מהו הקלט שהפלט עבורו הוא 45?

ד. מהו הערך הקטן ביותר שיוצג כפלט? עבור איזה ערך של num יוצג כפלט ערך זה?

ה. מהו הערך הגדול ביותר שיוצג כפלט? עבור איזה ערך של num יוצג כפלט ערך זה?

ו. ישמו את האלגוריתם בשפת C#.

ז. כתבו את האלגוריתם **ללא** שימוש בתבנית ***בניית מספר***.

שאלה 8

א. כתבו אלגוריתם, שהקלט שלו הוא מספר תלת-ספרתי num והפלט שלו הוא **כל** המספרים הדו-ספרתיים שניתן להרכיב משלוש הספרות של num.

ב. ציינו את מספר הפעמים בהם השתמשתם בתבנית ***בניית מספר***.

ג. עבור אילו ערכים של num יוצג כפלט ערך יחיד (החוזר על עצמו)?

ד. ישמו את האלגוריתם בשפת C#.

שאלה 9

נתון אלגוריתם חלקי שהקלט שלו הוא 3 ספרות והפלט שלו הוא מספר תלת-ספרתי המורכב משלוש הספרות:

1. קלוט ספרת מאות ב-hundreds, ספרת עשרות ב-tens וספרת אחדות ב-units

2. בנה מספר מ-hundreds ו-tens והשם ב-temp את הערך שהתקבל

3. בנה מספר מ- ________ ו- ________ והשם ב-num את המספר התלת-ספרתי שהתקבל

4. הצג כפלט את ערכו של num

א. השלימו את האלגוריתם.

ב. הסבירו את תפקידו של המשתנה temp.

ג. ישמו את האלגוריתם בשפת C#.

ד. ילנה הציעה תבנית לאלגוריתם של בניית מספר משלוש ספרות **ללא** שימוש **כפול** בתבנית ***בניית מספר*** עבור שתי ספרות. לפניכם האלגוריתם החלקי:

1. *קלוט ספרת מאות ב-hundreds, ספרת עשרות ב-tens וספרת אחדות ב-units*
2. *השם ב-num* את ________ + 10*________ + 100*________
3. *הצג כפלט את ערכו של* num

השלימו את האלגוריתם.

ה. נתון שימוש בתבנית:

בנה מספר מ-digit, digit ו-digit והצג כפלט את הערך שהתקבל

מה יוצג כפלט עבור הערך 8 ב-digit?

שאלה 10

נתון אלגוריתם חלקי, שהקלט שלו הוא מספר ארבע-ספרתי חיובי num:

1. *קלוט מספר ארבע-ספרתי חיובי ב-num*
2. *חשב את ספרת האחדות על ידי ________ והשם ב-units*
3. *חשב את ספרת העשרות של* num *על ידי ________ והשם ב-tens*
4. *חשב את ספרת המאות של* num *על ידי ________ והשם ב-hundreds*
5. *חשב את ספרת האלפים של* num *על ידי חישוב מנת החלוקה של ______ פריטים ל-______ קבוצות והשם ב-thousands*
6. *בנה מספר מ-tens ו-units והכפל אותו ב-100*
7. *בנה מספר מ-thousands ו-hundreds*
8. *הצג כפלט את סכום המספרים שחושבו*

א. השלימו את האלגוריתם.

ב. מהו הפלט עבור הקלט 5243?

ג. מהו הקלט עבור הפלט 1197?

ד. מהי מטרת האלגוריתם?

ה. ניתן לכתוב אלגוריתם המבצע אותה מטרה **ללא** שימוש בתבנית ***בניית מספר*** עבור שתי ספרות. לפניכם האלגוריתם החלקי:

1. *קלוט מספר ארבע-ספרתי חיובי ב-num*
2. *חשב את שתי הספרות השמאליות על ידי ________ והשם ב-left*
3. *חשב את שתי הספרות הימניות על ידי ________ והשם ב-right*
4. *הצג כפלט את הערך של* ________ * 100 + ________

השלימו את האלגוריתם.

אחד השימושים של התבנית ***בניית מספר*** הוא לצורך היפוך ספרותיו של מספר נתון. נראה זאת בשתי השאלות הבאות:

שאלה 11

נתון אלגוריתם חלקי, שהקלט שלו הוא מספר דו-ספרתי חיובי num, והפלט שלו הוא מספר דו-ספרתי שסדר ספרותיו הפוך מסדר הספרות במספר הנקלט num:

א. השלימו את האלגוריתם:

1. *קלוט מספר דו-ספרתי חיובי ב-num*
2. *חשב את ספרת האחדות על ידי ________ והשם ב-units*
3. *חשב את ספרת העשרות של num על ידי ________ והשם ב-tens*
4. *בנה מספר מ-________ ו-________ והשם ב-reverseNum*
5. *הצג כפלט את ערכו של reverseNum*

ב. כתבו את כל הערכים האפשריים ל-num עבורם הפלט יהיה זהה לקלט.

ג. ישמו את האלגוריתם בשפת #C.

שאלה 12

נתון אלגוריתם חלקי שהקלט שלו הוא מספר תלת-ספרתי חיובי num, והפלט שלו הוא מספר תלת-ספרתי המורכב משלוש הספרות של num אך בסדר הפוך.

1. *קלוט מספר תלת-ספרתי חיובי ב-num*
2. *חשב את ספרת האחדות על ידי ________ והשם ב-units*
3. *חשב את ספרת העשרות של num על ידי ________ והשם ב-tens*
4. *חשב את ספרת המאות של num על ידי ________ והשם ב-hundreds*
5. *בנה מספר מ-________, ________ ו- ________ והשם ב-reverseNum*
6. *הצג כפלט את ערכו של reverseNum*

א. השלימו את האלגוריתם.

ב. ישמו את האלגוריתם בשפת #C.

תבניות – פרק 5

מציאת מקסימום ומינימום בסדרה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 1: תלמיד רשאי לגשת לבחינת הבגרות במתמטיקה בשני מועדים שונים: מועד א' ומועד ב'. ציונו של התלמיד בבחינה נקבע על פי הציון הגבוה מבין השניים. כתבו אלגוריתם שהקלט שלו הוא ציוניה של לירון (כמספרים שלמים) בבחינת הבגרות במתמטיקה במועד א' ובמועד ב', והפלט שלו הוא הציון הקובע של לירון בבחינה.

בעיה 2: כתבו אלגוריתם שהקלט שלו הוא שתי אותיות גדולות ב-ABC והפלט שלו הוא האות המאוחרת על פי הסדר המילוני. הניחו שהאותיות שונות זו מזו.

אנו רואים כי בשתי הבעיות האלגוריתמיות יש למצוא את המקסימום בסדרה בת שני ערכים. בבעיה הראשונה יש למצוא את המקסימום בסדרה בת שני ערכים מספריים ובבעיה השנייה יש למצוא את המקסימום בסדרה בת שני ערכים תוויים. באופן דומה ישנן בעיות אלגוריתמיות שעבורן יש למצוא את המינימום בסדרת ערכים. מציאת הערך הגדול או הקטן ביותר בסדרת ערכים הינה אחת התבניות הבסיסיות ביותר במדעי המחשב. תבנית זו שימושית הן בפני עצמה והן כמרכיב בתבניות מורכבות יותר כמו תבניות של מיון סדרת ערכים, שנכיר בהמשך לימודינו.

נתבונן בשני האלגוריתמים הללו:

1. קלוט ציונים של לירון ב-math1 וב-math2 2. השם ב-max את ערכו של math1 3. אם math2 > max 3.1. השם ב-max את ערכו של math2 4. הצג כפלט את הערך max	1. קלוט שתי אותיות ב-letter1 וב-letter2 2. השם ב-max את ערכו של letter1 3. אם letter2 > max 3.1. השם ב-max את ערכו של letter2 4. הצג כפלט את הערך max

אנו רואים כי בשני הפתרונות נקבע באופן שרירותי כי הערך של המשתנה הראשון הוא המקסימלי ולכן נשמר ערכו במשתנה max. לאחר מכן, נבדק ערכו של המשתנה השני: אם הוא גדול יותר מהמקסימום שנקבע אז ערכו של max מוחלף בערך האיבר השני.

הבחירה השרירותית במשתנה הראשון כערך התחלתי ל-max נראית לכאורה מיותרת. ואכן, ניתן לוותר עליה, ולהשתמש בהוראה במבנה *אם... אחרת...*, המשווה בין שני הערכים ומשימה בהתאם לתוצאת ההשוואה את ערכו של אחד מהם ב-max (בדומה למה שנעשה בשאלה 5.6). בכל

זאת, נעדיף את הפתרונות כפי שהוצגו, משום שקל יהיה להרחיבם למציאת מקסימום בסדרה שבה יותר משני איברים, כפי שנראה בשלב מאוחר יותר.

נפריד את מאפייני התבנית ***מציאת מקסימום ומינימום בסדרה*** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית ***מציאת מקסימום בסדרה*** ואחר כך נציג את מאפייני התבנית ***מציאת מינימום בסדרה***.

שם התבנית: מציאת מקסימום בסדרה

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: מציאת הערך הגדול ביותר מבין שני הערכים

אלגוריתם:

1. *השם ב-*max *את הערך של* element1
2. *אם* element2 > max
   2.1 *השם ב-*max *את הערך של* element2

יישום ב-#C:

```
max = element1;
if (element2 > max)
{
      max = element2;
}
```

שם התבנית: מציאת מינימום בסדרה

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: מציאת הערך הקטן ביותר מבין שני הערכים

אלגוריתם:

1. *השם ב-*min *את הערך של* element1
2. *אם* element2 < min
   2.1 *השם ב-*min *את הערך של* element2

יישום ב-#C:

```
min = element1;
if (element2 < min)
{
      min = element2;
}
```

שאלה 1

א.

1. ישמו בשפת #C את האלגוריתם לחישוב הציון הקובע של לירון במתמטיקה.
2. הרחיבו את התוכנית שכתבתם בסעיף א.1 כך שיוצג כפלט גם מועד הבחינה (א' או ב') שבו הושג הציון הגבוה יותר.
3. איזה מועד בחינה יוצג כפלט במקרה שציוניה של לירון זהים בשני המועדים? הסבירו.

ב.

1. ישמו בשפת #C את האלגוריתם למציאת האות המאוחרת יותר.
2. שנו את התוכנית שכתבתם בסעיף ב.1 כך שתוצג כפלט האות המוקדמת יותר על פי סדר מילוני.

שאלה 2

נתון אלגוריתם שהקלט שלו הוא שני מספרים ממשיים:

1. *קלוט שני מספרים ממשיים ב-num1 ו-num2*
2. *מצא מקסימום בסדרה num1,num2 והשם את ערכו ב-max*
3. *מצא מינימום בסדרה num1,num2 והשם את ערכו ב-min*
4. *הצג כפלט את ערכו של הביטוי החשבוני max – min*

א. מהו הפלט עבור הקלט 6.9 5.3?

ב. תנו שתי דוגמאות קלט שונות שעבורן יוצג כפלט הערך 9.1.

ג. מהי מטרת האלגוריתם?

ד. כתבו אלגוריתם המשיג את אותה המטרה **ללא** שימוש בתבניות ***מציאת מקסימום בסדרה*** ו-***מציאת מינימום בסדרה***.

שאלה 3

נתון אלגוריתם שהקלט שלו הוא שני מספרים שלמים שונים והפלט שלו הוא המספר הדו-ספרתי הקטן ביותר מבין ערכי הקלט. הניחו ש**לפחות** אחד מבין שני המספרים שבקלט הוא דו-ספרתי.

1. *קלוט שני מספרים שלמים ב-num1, num2*
2. *מצא מינימום בסדרה num1, num2 והצג כפלט את ערכו*

האלגוריתם שגוי.

א. תנו דוגמת קלט שעבורה לא ניתן לראות כי האלגוריתם שגוי. מהו המאפיין של הדוגמה?

ב. תנו שתי דוגמאות קלט שונות שעבורן ניתן לראות כי האלגוריתם שגוי. מהו המאפיין של כל אחת מהדוגמאות?

ג. הסבירו במלים מדוע האלגוריתם שגוי.

ד. תקנו את האלגוריתם.

ה. ישמו את האלגוריתם כקטע תוכנית בשפת #C.

שאלה 4

נתון אלגוריתם שהקלט שלו הוא 3 מספרים שלמים השונים זה מזה:

1. קלוט 3 מספרים שלמים ב-num1, num2 וב-num3
2. מצא מינימום בסדרה num1, num2 והשם את ערכו ב-min
3. מצא מינימום בסדרה min, num3 והשם את ערכו ב-min
4. הצג כפלט את ערכו של min

א. מה יוצג כפלט עבור הקלט 57 34 816?

ב. תנו שלוש דוגמאות קלט שונות שעבורן יהיה הפלט 5. מהו המאפיין של כל אחת מדוגמאות הקלט?

ג. ישמו את האלגוריתם בשפת C#.

שאלה 5 (שאלה זו מתאימה לאחר לימוד סעיף 5.3 – קינון של הוראה לביצוע בתנאי)

נתונה הבעיה האלגוריתמית הבאה:

כתבו אלגוריתם שהקלט שלו הוא 3 גבהים של שחקני כדורסל והפלט שלו הוא הגובה המקסימלי מבין שלושת נתוני הקלט. הניחו שהגבהים שונים זה מזה.

זוהי למעשה בעיית מציאת מקסימום בסדרה בת 3 ערכים. אנה, בן, ענר ומשה הציעו אלגוריתמים שונים לפתרון הבעיה.

האלגוריתם של אנה:

1. קלוט 3 גבהים של שחקני הכדורסל ב-height1 , height2 וב-height3
2. השם ב-max את ערכו של height1
3. אם height2 > max
   3.1. השם ב-max את ערכו של height2
4. אם height3 > max
   4.1. השם ב-max את ערכו של height3
5. הצג כפלט את ערכו של max

האלגוריתם של בן:

1. קלוט 3 גבהים של שחקני הכדורסל ב-height1 , height2 וב-height3
2. אם height1 > height2 וגם height2 > height3
   2.1. הצג כפלט את ערכו של height1
3. אם height2 > height3 וגם height3 > height1
   3.1. הצג כפלט את ערכו של height2

4. אם height3 > height1 וגם height1 > height2

4.1. הצג כפלט את ערכו של height3

האלגוריתם של ענר:

1. קלוט 3 גבהים של שחקני הכדורסל ב-height1, height2 וב-height3
2. אם height1 > height2 וגם height1 > height3
   2.1. הצג כפלט את ערכו של height1
3. אם height2 > height1 וגם height2 > height3
   3.1. הצג כפלט את ערכו של height2
4. אם height3 > height1 וגם height3 > height2
   4.1. הצג כפלט את ערכו של height3

האלגוריתם של משה:

1. קלוט 3 גבהים של שחקני הכדורסל ב-height1, height2 וב-height3
2. אם height1 > height2
   2.1. אם height1 > height3
      2.1.1. הצג כפלט את ערכו של height1
   2.2. אחרת
      2.2.1. הצג כפלט את ערכו של height3
3. אחרת
   3.1. אם height2 > height3
      3.1.1. הצג כפלט את ערכו של height2
   3.2. אחרת
      3.2.1. הצג כפלט את ערכו של height3

עבור **כל אחד** מהאלגוריתמים המוצעים ענו על הסעיפים הבאים:

א. אם האלגוריתם נכון:

1. בנו טבלת מעקב עבור הקלט 1.98 2.05 1.94.
2. הסבירו במלים את הרעיון עליו מתבסס האלגוריתם.

ב. אם האלגוריתם אינו נכון:

1. תנו לפחות דוגמת קלט אחת שעבורה ניתן לראות כי האלגוריתם שגוי. אפיינו את דוגמת הקלט.

2. תנו לפחות דוגמת קלט אחת שעבורה **לא** ניתן לראות כי האלגוריתם שגוי. אפיינו את דוגמת הקלט.

שאלה 6

נתון אלגוריתם חלקי שהקלט שלו הוא 4 מספרים שלמים השונים זה מזה והפלט שלו הוא המספר הקטן ביותר מבין נתוני הקלט:

1. *קלוט 4 מספרים שלמים ב-*num1 ,num2 ,num3 *ו-*num4
2. *מצא מינימום בסדרה ________,________ והשם ב-________*
3. *מצא מינימום בסדרה ________,________ והשם ב-________*
4. *מצא מינימום בסדרה ________,________ והשם ב-________*
5. *הצג כפלט את ערכו של* min

א. הציעו שתי דרכים שונות להשלמת האלגוריתם. עבור כל אחת מדרכי הפתרון שהצעתם הסבירו את הרעיון עליו היא מתבססת.

ב. באלגוריתם זה יש שימוש שלוש פעמים בתבנית ***מציאת מינימום בסדרה*** בת שני ערכים. רשמו אלגוריתם שקול (כלומר, המשיג אותה מטרה), המשתמש בתבנית ***מציאת מינימום בסדרה*** בת שלושה ערכים.

שאלה 7

א. כתבו אלגוריתם שהקלט שלו הוא 4 מספרים שלמים והפלט שלו הוא המספר הגדול ביותר מבין הארבעה וכן מספר הפעמים שהופיע מספר זה בקלט.

לדוגמה, עבור הקלט: 54 13 54 54 הפלט המתאים הוא: 3 54.

ב. ישמו את האלגוריתם בשפת #C.

שאלה 8

א. כתבו אלגוריתם שהקלט שלו הוא מספר דו-ספרתי, והפלט שלו הוא המספר הגדול יותר מבין המספר הנתון והמספר המתקבל מהיפוך ספרותיו של המספר הנתון.

ב. ציינו באילו תבניות השתמשתם בפתרון הבעיה.

ג. ישמו את האלגוריתם בשפת #C.

סידור ערכים בסדרה

סידור ערכים בסדרה היא תבנית של מיון ערכי סדרה נתונה. ניתן לסדר את ערכי הסדרה בסדר עולה או לסדרם בסדר יורד.

התבנית ***סידור ערכים בסדר עולה בסדרה*** מביאה למצב בו איברי הסדרה ממוינים בסדר עולה. מכאן נובע כי הערך המינימלי בסדרה נמצא בקצה השמאלי של הסדרה והערך המקסימלי נמצא בקצה הימני של הסדרה.

התבנית ***סידור ערכים בסדר יורד בסדרה*** מביאה למצב בו איברי הסדרה ממוינים בסדר יורד. מכאן נובע כי הערך המקסימלי בסדרה נמצא בקצה השמאלי של הסדרה והערך המינימלי נמצא בקצה הימני של הסדרה.

נפריד את מאפייני התבנית ***סידור ערכים בסדרה*** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית ***סידור ערכים בסדר עולה בסדרה*** ואחר כך נציג את מאפייני התבנית ***סידור ערכים בסדר עולה בסדרה***.

תחילה נציג את התבניות לסידור ערכים בסדרה בת שני ערכים בסדר עולה ובסדר יורד, לאחר מכן נרחיב לסדרה בת 3 ערכים ובהמשך נראה אלגוריתם כללי יותר לסידור מספר גדול יותר של ערכים בסדרה.

שם התבנית: סידור ערכים בסדר עולה בסדרה

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: השמת הערך הקטן יותר ב-element1 והערך הגדול יותר ב-element2

אלגוריתם:

1. *אם* element1 > element2

1.2 *החלף את ערכי* element1 *ו-*element2

שם התבנית: סידור ערכים בסדר יורד בסדרה

נקודת מוצא: שני ערכים במשתנים element1 ו-element2

מטרה: השמת הערך הגדול יותר ב-element1 והערך הקטן יותר ב-element2

אלגוריתם:

1. *אם* element1 < element2

1.2 *החלף את ערכי* element1 *ו-*element2

שאלה 9

ישמו כל אחד מן האלגוריתמים לסידור הערכים element1 ו-element2 כקטע תוכנית בשפת C#.

שימו לב כי כדי ליישם את התבניות של ***סידור ערכים בסדרה*** עליכם להשתמש ביישום של התבנית ***החלפת ערכים בין שני משתנים***.

שאלה 10

א. פתחו אלגוריתם שהקלט שלו הוא מספר דו-ספרתי num והפלט שלו הוא המספר הגדול ביותר שניתן להרכיב מספרות המספר הנתון.

ב. ציינו באילו תבניות השתמשתם עבור כתיבת האלגוריתם.

ג. ישמו את האלגוריתם בשפת C#.

שאלה 11

נתונה סדרה של שלושה ערכים: element1, element2, element3. ידוע כי הערך השני בגודלו **אינו** נמצא ב-element2. נתון אלגוריתם חלקי, שמטרתו לסדר את ערכי הסדרה בסדר עולה:

1. *אם* element1 < element2 *// תיאור קיום תנאי:* ________

1.1. *החלף את ערכי* ________ *ו-* ________

1.2. *סדר בסדר עולה את איברי הסדרה* ________, ________

2. *אחרת // תיאור אי-קיום תנאי:* ________

2.1. *החלף את ערכי* ________ *ו-* ________

2.2. *סדר בסדר עולה את איברי הסדרה* ________, ________

א. השלימו את האלגוריתם. הוסיפו תיאורי קיום ואי-קיום תנאי במקומות המסומנים.

ב. שנו את האלגוריתם שכתבתם כך שיסדר את ערכי הסדרה בסדר **יורד**.

ג. ישמו את שני האלגוריתמים כקטעי תוכניות בשפת C#.

שאלה 12

א. כתבו אלגוריתם שהקלט שלו הוא סדרה של 3 מספרים שלמים שונים והפלט שלו הוא סידור של המספרים, כך שעבור הסידור המתקבל שני הערכים המוחלטים של הפרשי המספרים יהיו בסדר עולה.

ב. ציינו באילו תבניות השתמשתם עבור כתיבת האלגוריתם.

ג. ישמו את האלגוריתם בשפת C#.

שאלה 13

נתון אלגוריתם, שהקלט שלו הוא סדרה של שלושה ערכים element1, element2, element3:

1. *קלוט שלושה מספרים שלמים ב-*element1, element2, element3

2. *סדר בסדר עולה את איברי הסדרה* element1, element2

3. *סדר בסדר עולה את איברי הסדרה* element2, element3

4. *סדר בסדר עולה את איברי הסדרה* element1, element2

5. *הצג כפלט את הערכים של* element1, element2, element3

א. מה יהיה הפלט עבור הקלט 6 13 8?

ב. מהי מטרת האלגוריתם?

ג. באלגוריתם נעשה שלוש פעמים שימוש בתבנית ***סידור ערכים בסדר עולה בסדרה***.

1. תנו דוגמה לקלט שעבורו מתבצעת החלפת ערכי המשתנים שלוש פעמים.
2. תנו דוגמה לקלט שעבורו לא מתבצעת החלפת ערכי המשתנים כלל.

ד. ישמו את האלגוריתם בשפת C#.

ערכים עוקבים

שם התבנית: ערכים עוקבים?

נקודת מוצא: שני ערכים element1 ו-element2

מטרה: קביעת הערך true אם element2 עוקב ל-element1, וקביעת הערך false אם element2 אינו עוקב ל-element1

אלגוריתם (ביטוי בוליאני):

element1 + 1 = element2

יישום ב-#C:

```
(element1 + 1 == element2)
```

התבנית ***ערכים עוקבים?*** היא תבנית המחשבת ערך בוליאני, כלומר, true או false. משום כך, האלגוריתם שמממש את התבנית, וכמוהו גם יישומו בשפת #C כוללים למעשה ביטוי בוליאני. את הערך המחושב על ידי התבנית ניתן לשלב בביטוי בוליאני.

שימו ♥: הערכים element1 ו-element2 עשויים להיות מספרים שלמים, תווים וכן כל זוג ערכים בדידים שניתנים לסידור.

שאלה 14

לפניכם שימוש בתבנית ***ערכים עוקבים?***:

1. *אם* 5 ערך עוקב ל-num

1.1. *הצג כפלט "אמת"*

2. *אחרת*

2.1. *הצג כפלט "שקר"*

א. תנו דוגמה לערך של num שעבורו יוצג כפלט "אמת".

ב. תנו שתי דוגמאות לערך של num שעבורו יוצג כפלט "שקר".

שאלה 15

נתונים שני מספרים שלמים num1 ו-num2. עבור כל אחד מהסעיפים הבאים כתבו ביטוי בוליאני מתאים:

א. המספר הראשון אינו עוקב למספר השני.

ב. שני המספרים עוקבים (הסדר אינו משנה).

שאלה 16

נתונים שני תווים ch1 ו-ch2. עבור כל אחד מהסעיפים הבאים כתבו ביטוי בוליאני מתאים:

א. שני התווים הם אותיות קטנות עוקבות ב-abc או ששני התווים הם אותיות גדולות עוקבות ב-ABC.

ב. שני התווים הם ספרות עוקבות (בין '0' ל-'9').

זוגיות מספר

בדיקת זוגיות של מספר שימושית בהקשרים רבים במדעי המחשב. למשל, נזכור כי בזיכרון המחשב נשמרים ערכים כסדרות של סיביות. אם מתייחסים לסדרת סיביות כאל מספר, אז בדיקת הזוגיות של המספר מעידה האם הסיבית הימנית ביותר במספר היא 0 או 1.

נפריד את מאפייני התבנית ***זוגיות מספר*** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית ***מספר זוגי?*** ואחר כך נציג את מאפייני התבנית ***מספר אי-זוגי?***. שתי התבניות מחשבות ערכים בוליאניים, בדומה לתבנית ***ערכים עוקבים?***.

שם התבנית: מספר זוגי?

נקודת מוצא: מספר שלם num

מטרה: קביעת הערך true אם num זוגי וקביעת הערך false אם num אי-זוגי

אלגוריתם (ביטוי בוליאני):

שארית החלוקה של num פריטים ל-2 קבוצות שווה ל-0

שם התבנית: מספר אי-זוגי?

נקודת מוצא: מספר שלם num

מטרה: קביעת הערך true אם num אי-זוגי וקביעת הערך false אם num זוגי

אלגוריתם (ביטוי בוליאני):

שארית החלוקה של num פריטים ל-2 קבוצות שווה ל-1

שימו ♥: שתי התת-תבניות ***מספר זוגי?*** ו-***מספר אי-זוגי?*** הן שתי תבניות המשלימות זו את זו. כלומר, עבור מספר שלם מסוים, חישוב הערך true בשימוש בתבנית אחת יגרור חישוב הערך false בשימוש בתבנית השנייה, ולהיפך.

שאלה 17

נתונים שני מספרים שלמים num1 ו-num2. לפניכם מספר ביטויים בוליאניים חלקיים המחזירים ערך true אם אחד משני המספרים זוגי והאחר אי-זוגי, ו-false אחרת.
השלימו את הביטויים הבוליאניים:

א. (________מספר זוגי **וגם** ________ מספר אי-זוגי)

או

(________מספר זוגי **וגם** ________ מספר אי-זוגי)

ב. (שארית החלוקה של num1 פריטים ל-2 קבוצות **שווה ל**-1) ________ (שארית החלוקה של num1 פריטים ל-2 קבוצות **שווה ל**-1)

ג. (________מספר זוגי)

שאלה 18

נתון אלגוריתם שהקלט שלו הוא מספר שלם num:

1. **קלוט מספר שלם ב**-num
   1.1. **אם** num מספר זוגי
      1.1.1. **הקטן את ערכו של** num **פי** 2
   1.2. **אחרת**
      1.2.1. **הגדל את ערכו של** num **פי** 2
2. **הצג כפלט את ערכו של** num

א. מה יהיה הפלט עבור הקלט 15? היעזרו בטבלת מעקב.

ב. תנו שתי דוגמאות קלט **שונות**, שעבורן יוצג כפלט הערך 10.

ג. ישמו את האלגוריתם בשפת #C.

שאלה 19

בחברת "מודיעין אזרחי" לכל לקוח יש מספר ייחודי משלו. החברה מצפינה את מספרי לקוחותיה באופן הבא: כל ספרה אי-זוגית הופכת לזוגית על ידי הפחתה של 1. שאר הספרות נותרות ללא שינוי. למשל, עבור מספר הלקוח 921 יתקבל הקוד 820 ועבור מספר הלקוח 129 יתקבל הקוד 28 (הספרה 1 הפכה ל-0, ומשום שהייתה ספרה מובילה נעלמה).

א. כתבו אלגוריתם, שהקלט שלו הוא מספר תלת-ספרתי של לקוח והפלט שלו הוא המספר המוצפן.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם.

ג. לאחר שנה החליטה החברה להצפין שוב את מספרי לקוחותיה. הפעם בחרה בשיטת הצפנה זו: כל ספרה אי-זוגית הופכת לזוגית על ידי הוספת 1 ב"צורה מעגלית" (כלומר, 9 הופכת ל-0). שאר הספרות נשארות ללא שינוי. למשל, עבור מספר הלקוח 976 יתקבל הקוד 86 ועבור מספר הלקוח 439 יתקבל הקוד 440. הרחיבו את האלגוריתם שכתבתם גם עבור שיטת ההצפנה השנייה.

ד. ישמו את האלגוריתם בשפת #C.

מחלק של מספר

התבנית ***מחלק של?*** היא הרחבה של התבנית ***מספר זוגי?***. נאמר על מספר שלם num2 שהוא מחלק את המספר השלם num1 אם num1 מתחלק ב-num2 ללא שארית. למעשה, התבנית ***מספר זוגי?*** בודקת האם 2 הוא מחלק של מספר נתון, ולכן שימוש בתבנית ***מחלק של?*** עבור שני מספרים num1 ו-num2, כאשר ערכו של num2 הוא 2, שקול לשימוש בתבנית ***מספר זוגי?*** עבור num1. כמו ***מספר זוגי?*** גם ***מחלק של?*** היא תבנית המחשבת ערך בוליאני.

שם התבנית: מחלק של?

נקודת מוצא: שני מספרים שלמים num1 ו-num2

מטרה: קביעת הערך true אם num2 מחלק את num1 וקביעת הערך false אם num2 אינו מחלק את num1

אלגוריתם (ביטוי בוליאני):

שארית החלוקה של num1 *פריטים ל-*num2 *קבוצות שווה ל-*0

שאלה 20

נתונים שני מספרים שלמים num1 ו-num2. עבור כל אחד מהסעיפים הבאים כתבו ביטוי בוליאני מתאים:

א. סכום שני המספרים מתחלק ב-4 ללא שארית.

ב. המספר הראשון זוגי ואינו מתחלק ב-3.

ג. המספר הראשון הוא מספר דו-ספרתי הגדול מ-50, וספרת העשרות שלו שווה לספרת האחדות של המספר השני.

שאלה 21

נתון הביטוי הבוליאני הבא עבור מספר שלם חיובי num:

(num *מספר זוגי*) ***וגם*** (*שארית החלוקה של* num *פריטים ל-*5 *קבוצות שווה ל-*0)

א. כתבו את הביטוי הבוליאני בשפת #C.

ב. תנו דוגמה לערך של num, שעבורו יהיה ערך הביטוי true. הסבירו.

ג. תנו דוגמה לערך של num, שעבורו יהיה ערך הביטוי false. הסבירו.

ד. כתבו ביטוי בוליאני פשוט השקול לביטוי הנתון.

שאלה 22

מורה מעוניינת לחלק את תלמידיה לזוגות לצורך מבצע התרמה שבו הם נוטלים חלק.

א. כתבו תוכנית בשפת #C, שהקלט שלה הוא מספר התלמידים בכיתה והפלט שלה הוא הודעה האם ניתן להתאים בן-זוג לכל תלמיד.

ב. מאחר שהמורה ראתה כי לא ניתן להתאים בן-זוג לכל תלמיד החליטה לחלק את הכיתה לקבוצות של שלושה תלמידים.

1. שנו את התוכנית שכתבתם בסעיף א כך שתוצג כפלט הודעה האם ניתן לחלק את הכיתה לקבוצות של שלושה תלמידים.

2. האם עתה נפתרה בעייתה של המורה? אם כן, הסבירו. אם לא, תנו דוגמת קלט שעבורה ניתן לראות כי המורה שגתה בהחלטתה.

שאלה 23

א. כתבו אלגוריתם שהקלט שלו הוא זמן המיוצג בשעות ובדקות, והפלט שלו הוא הזמן הנותר בשעות ובדקות עד שעת חצות.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם.

ג. ישמו את האלגוריתם בשפת C#.

תבניות – פרק 7

מנייה וצבירה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 1: עליה על מתקן "רכבת הרים" מותרת רק לילדים שגובהם עולה על 1.40 מטר וגילם עולה על 8. כתבו אלגוריתם שהקלט שלו 40 זוגות ערכים המייצגים את הגובה והגיל של 40 ילדים, והפלט שלו הוא מספר הילדים הראשים לעלות למתקן.

בעיה 2: מנהל האתר הבית-ספרי "ישראלנד" מעוניין לדעת מהו מספר הכניסות לאתר ביממה. כתבו אלגוריתם שהקלט שלו הוא סדרה של קודי משתמשים, המסתיימת בזקיף 0, והפלט שלו הוא מספר הכניסות לאתר ביממה.

בשתי הבעיות האלגוריתמיות עלינו **למנות** (לספור) ערכים. בבעיה 1 יש למנות את מספר הילדים הרשאים לעלות למתקן "רכבת הרים" ובבעיה 2 יש למנות את מספר הכניסות לאתר ביממה. מנייה היא אחת התבניות השימושיות ביותר בפיתוח אלגוריתמים. השימוש בתבנית זו נעשה בכל פעם שיש למנות כמות הופעות של ערך.

נתבונן בשני האלגוריתמים הבאים, הפותרים את שתי הבעיות שלעיל:

1. אתחל את count ב-0 2. **בצע** 40 **פעמים**: 2.1. קלוט גובה ב-height וגיל ב-age 2.2. **אם** height > 1.4 **וגם** age > 8 2.2.1. הגדל את count ב-1 3. הצג כפלט את ערכו של count	1. אתחל את count ב-0 2. קלוט קוד משתמש ב-code 3. **כל עוד** code <> 0 **בצע**: 3.1. הגדל את count ב-1 3.2. קלוט קוד משתמש ב-code 4. הצג כפלט את ערכו של count

בשני האלגוריתמים ערכו של המונה (count) מאותחל לפני הלולאה ל-0 ובגוף הלולאה גדל ערכו של המונה ב-1. שימו לב כי בפתרון לבעיה 1 מספר הפעמים שמבוצע גוף הלולאה ידוע מראש (40) ואילו בפתרון לבעיה 2 מספר הפעמים שמבוצע גוף הלולאה אינו ידוע מראש (סדרת נתוני הקלט מסתיימת עם קליטת הזקיף 0).

עתה נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 3: כתבו אלגוריתם שהקלט שלו הוא 15 מספרים ממשיים והפלט שלו הוא סכום המספרים החיוביים.

בעיה 4: בתחרות הרמת משקולות משקלה של המשקולת ההתחלתית הוא 20 ק״ג. כל המתחרים בתחרות מסוגלים להרים משקולת זו. עם התקדמות התחרות נוספות עוד ועוד משקולות למשקולת ההתחלתית. כתבו אלגוריתם שהקלט שלו הוא סדרה של זוגות נתונים: הנתון הראשון הוא משקל המשקולת אותה מוסיפים למשקל המצטבר שהרים מתחרה, והנתון השני הוא התו 'y' אם המתחרה הצליח להרים את המשקל המצטבר כולל המשקולת החדשה, והתו 'n' אם לא הצליח. התו 'n' יופיע רק בזוג הנתונים האחרון. הפלט של האלגוריתם הוא המשקל המצטבר של המשקולות שהמתחרה הצליח להרים.

בשתי הבעיות האלגוריתמיות עלינו **לצבור** (בדוגמה זו, לחשב סכום) ערכים. בבעיה 3 יש לצבור מספרים חיוביים ובבעיה השנייה יש לצבור את משקלי המשקולות. תבנית הצבירה, בדומה לתבנית מנייה, הינה תבנית שימושית ביותר בפיתוח אלגוריתמים. השימוש בתבנית זו נעשה בכל פעם שיש לצבור סדרת ערכים בצורה כלשהי, למשל על ידי סכום או על ידי מכפלה.

נתבונן בשני האלגוריתמים, הפותרים את שתי הבעיות שלעיל:

1. אתחל את sum ב-0 2. **בצע** 15 **פעמים**: 2.1. קלוט מספר ממשי ב-num 2.2. **אם** num > 0 2.1.1. הוסף ל- sum את num 3. הצג כפלט את ערכו של sum	1. אתחל את sum ב-20 2. קלוט משקל משקולת ב-weight ואת המשך ב- continuation 3. **כל עוד** 'n' <> continuation **בצע**: 3.1. הוסף ל- sum את weight 3.2. קלוט משקל משקולת ב-weight ואת המשך ב-continuation 4. הצג כפלט את ערכו של sum

בשני האלגוריתמים ערכו של הצובר (sum) מאותחל לפני הלולאה לערך תחילי, ולהבדיל ממונה, ערך זה אינו בהכרח 0. בפתרון לבעיה 3 ערכו מאותחל ב-0 ובפתרון לבעיה 4 ערכו מאותחל ב-20. בגוף הלולאה משתנה ערכו של הצובר בערך כלשהו, השונה מ-1 (שימו לב כי ייתכן שערכו של הצובר עלול לקטון במהלך הצבירה, למשל, במקרה של צבירת ערכים שליליים).

נציג כעת את מאפייני שתי התבניות. ראשית, נציג את מאפייני התבנית ***מנייה*** ואחר כך נציג את מאפייני התבנית ***צבירה***. עבור כל אחת מהתבניות נראה אלגוריתם עבור ביצוע חוזר באורך הידוע מראש וכן אלגוריתם לביצוע חוזר בתנאי.

שם התבנית: מנייה

נקודת מוצא: אורך סדרת נתוני הקלט limit, סדרת ערכי הקלט , תנאי מנייה condition

מטרה: מנייה של ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

אלגוריתם:

1. אתחל את count ב-0
2. בצע limit פעמים:
   2.1. קלוט ערך ב-element
   2.2. אם element מקיים את condition
      2.2.1. הגדל את count ב-1

יישום ב-#C:

```
count = 0;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (condition)
    {
        count++;
    }
}
```

שם התבנית: מנייה

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, תנאי מנייה conditionToCount

מטרה: מנייה של ערכי הקלט המקיימים את התנאי conditiontoCount. משך ביצוע המנייה תלוי בתנאי conditionToEnd.

אלגוריתם:

1. *אתחל את* count *ב-*0
2. *קלוט ערך ב-*element
3. ***כל עוד לא מתקיים*** conditionToEnd ***בצע:***
   - 3.1. ***אם*** element *מקיים את* conditionToCount
     - 3.1.1 *הגדל את* count *ב-*1
   - 3.2. *קלוט ערך ב-*element

יישום ב-#C:

```
count = 0;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (conditionToCount)
    {
        count++;
    }
    element = int.Parse(Console.ReadLine());
}
```

ישנן שתי צורות בסיסיות של צבירה: צבירת סכום וצבירת מכפלה. אנו נראה תחילה את התבניות של צבירת סכום שמשך ביצועה ידוע מראש וצבירת סכום שמשך ביצועה תלוי בתנאי.

שם התבנית: צבירת סכום

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערך תחילי של הצובר initial, ערכי הקלט, תנאי צבירה condition

מטרה: צבירת הסכום של הערך initial ושל סכום ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

אלגוריתם:

1. אתחל את sum ב-initial
2. בצע limit פעמים:
    2.1. קלוט ערך ב-element
    2.2. אם element מקיים את condition
        2.2.1. הוסף ל-sum את ערכו של element

יישום ב-#C:

```
sum = initial;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (condition)
    {
        sum += element;
    }
}
```

שם התבנית: צבירת סכום

נקודת מוצא: תנאי סיום conditionToEnd, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה conditionToSum

מטרה: צבירת סכום של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToSum. משך ביצוע הצבירה תלוי בתנאי conditionToEnd.

אלגוריתם:

1. *אתחל את* sum *ב*-initial
2. *קלוט ערך ב*-element
3. *כל עוד לא מתקיים* conditionToEnd *בצע:*
   - 3.1 *אם* element *מקיים את* conditionToSum
     - 3.1.1 *הוסף ל*-sum *את ערכו של* element
   - 3.2 *קלוט ערך ב*-element

יישום ב-#C:

```
sum = initial;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (conditionToSum)
    {
        sum += element;
    }
    element = int.Parse(Console.ReadLine());
}
```

עתה נראה את התבנית של צבירת מכפלה שמשך ביצועה ידוע מראש ושל צבירת מכפלה שמשך ביצועה תלוי בתנאי.

שם התבנית: צבירת מכפלה

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה condition

מטרה: צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי condition, מתוך סדרת קלט שאורכה limit

אלגוריתם:

1. *אתחל את* mult *ב-*initial
2. *בצע* limit *פעמים:*
   - 2.1. *קלוט ערך ב-*element
   - 2.2. *אם* element *מקיים את* condition
     - 2.2.1. *הכפל את* mult *ב-*element *והשם את המכפלה ב-*mult

יישום ב-#C:

```
mult = initial;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (condition)
    {
        mult *= element;
    }
}
```

שם התבנית: צבירת מכפלה

נקודת מוצא: תנאי סיום conditionToEnd, ערך התחלתי של הצובר initial, ערכי הקלט, תנאי צבירה conditionToMult

מטרה: צבירת מכפלה של הערך initial ושל ערכי הקלט המקיימים את התנאי conditionToMult. משך ביצוע הצבירה תלוי בתנאי conditionToEnd.

אלגוריתם:

1. *אתחל את* mult *ב*-initial
2. *קלוט ערך ב*-element
3. *כל עוד לא מתקיים* conditionToEnd **בצע:**
   3.1. **אם** element *מקיים את* conditionToMult
      3.1.1. *הכפל את* mult *ב*-element *והשם את המכפלה ב*-mult
   3.2. *קלוט ערך ב*-element

יישום ב-#C:

```
mult = initial;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (conditionToMult)
    {
        mult *= element;
    }
    element = int.Parse(Console.ReadLine());
}
```

שימו ♥ : ערכו של initial חייב להיות שונה מ-0, שהרי מכפלת 0 בכל מספר תשאיר את ערכו של הצובר 0. ערכו של הצובר יכול להיות שלילי, וערכו של הצובר עשוי לקטון אם ערכו של element גדול מ-0 וקטן מ-1 (כלומר, הוא שבר), או אם נצברים גם ערכים שליליים.

ועוד שימו ♥ : הבעיה האלגוריתמית אינה חייבת לכלול תנאים למנייה ולצבירה (condition, conditionToSum ו-conditionToMult). ייתכנו מקרים, כפי שאכן ראינו בבעיות האלגוריתמיות 2 ו-4, שהמנייה והצבירה מתבצעות ללא תנאי.

שאלה 1

א. ישמו את פתרונותיהן של ארבע הבעיות האלגוריתמיות 1-4 בשפת #C.

ב. בהתייחס לבעיה 1: שנו את התוכנית כך שתציג כפלט את מספר הילדים ש**אינם** רשאים לעלות למתקן "רכבת הרים".

ג. בהתייחס לבעיה 2: שנו את התוכנית כך שתציג כפלט את מספר הכניסות לאתר ביממה רק עבור המשתמשים שהקוד שלהם בין 100 ל-200.

ד. בהתייחס לבעיה 3: הרחיבו את התוכנית כך שתציג כפלט גם את סכום המספרים השליליים.

ה. בהתייחס לבעיה 4: הרחיבו את התוכנית כך שתציג כפלט גם את מספר המשקולות הכולל שהמתחרה הצליח להרים.

שאלה 2

נתונה סדרת הקלט הבאה: 0 15 -6 14 -3 2

א. מה יוצג כפלט עבור כל אחד מהשימושים השונים בתבניות:

1. *מנה את האיברים האי-חיוביים בסדרת הקלט שאורכה 6 והצג כפלט את הערך שהתקבל*
2. *מנה את האיברים הזוגיים בסדרת הקלט המסתיימת בזקיף 0 והצג כפלט את הערך שהתקבל*
3. *חשב את הסכום המצטבר של האיברים המתחלקים ב-3 בסדרת הקלט שאורכה 6 והצג כפלט את הערך שהתקבל*
4. *חשב את הסכום המצטבר של 7 עם כל האיברים בסדרת הקלט המסתיימת בזקיף 0 והצג כפלט את הערך שהתקבל*
5. *חשב את המכפלה המצטברת של האיברים האי-זוגיים בסדרת הקלט המסתיימת בזקיף 5 והצג כפלט את הערך שהתקבל*
6. *חשב את המכפלה המצטברת של 4 בכל האיברים בסדרת הקלט המסתיימת בזקיף 15 והצג כפלט את הערך שהתקבל*

ב. ישמו כל אחד מן השימושים בשפת C#.

שאלה 3

א. בכיתה בת 41 תלמידים קיימו בחירות לנציג מועצת תלמידים. שני תלמידים הציגו את מועמדותם וכל תלמיד בכיתה הצביע עבור אחד המועמדים. נתון אלגוריתם שהקלט שלו הוא סדרה באורך 41 של המספרים 1 ו-2 המייצגים את מספרי המועמדים, והפלט שלו הוא מספרו של המועמד שזכה ברוב קולות.

1. אתחל את count1 ב-0

2. אתחל את count2 ב-0

3. בצע 41 פעמים:

3.1. קלוט מספר של מועמד ב-candidate

3.2. אם candidate שווה ל-1

3.2.1. הגדל את count1 ב-1

3.3. אחרת

3.3.1. הגדל את count2 ב-1

4. אם count1 > count2

4.1. הצג כפלט "מועמד מספר 1 זכה ברוב קולות"

5. אחרת

5.1. הצג כפלט "מועמד מספר 2 זכה ברוב קולות"

באלגוריתם הנתון ישנו שילוב בשימוש של שני מונים: ציינו מהם ומהו תפקידם של אחד מהם.

ב. נתון אלגוריתם חלקי, השקול לאלגוריתם הנתון, אך משתמש במונה אחד. השלימו:

1. מנה את האיברים ה________ בסדרת הקלט שאורכה 41 והשם ב-count את הערך המתקבל

2. אם ________

2.1. הצג כפלט "מועמד מספר1 זכה ברוב קולות"

3. אחרת

3.1. הצג כפלט "מועמד מספר 2 זכה ברוב קולות"

שאלה 4

א. בסופרמרקט השכונתי בודקים מדי יום את הפדיון היומי ואת מספר הקונים. פתחו אלגוריתם שהקלט שלו הוא סדרת מספרים המייצגת את תשלומי הקונים, שבסיומה הזקיף 1-, והפלט שלו הוא מספר הקונים באותו יום והודעה המציינת אם ביום זה סך הפדיון מקניות בסכומים הגבוהים מ-500 ₪ עלה על סך הפדיון מקניות בסכומים שאינם גבוהים מ-500 ₪. ישמו את האלגוריתם בשפת C#.

ב. ציינו באילו תבניות השתמשתם בסעיף א וכיצד **שילבתם** ביניהן.

שאלה 5

תת-סדרה תחילית של המספרים הטבעיים היא סדרה מהצורה 1, 2, 3, … n. בניסיון לבחון את "כוחה של המכפלה לעומת הסכום" הטילה המורה על תלמידיה לסכם את ערכיהן של תת-סדרות תחיליות של המספרים הטבעיים וכן להכפיל את ערכיהן.

א. פתחו אלגוריתם שאינו מקבל קלט, והפלט שלו הוא האורך הקטן ביותר של תת-סדרה תחילית של המספרים הטבעיים, שעבורה ערכה של המכפלה המצטברת יהיה לפחות פי 100 מערכו של הסכום המצטבר. ישמו את האלגוריתם בשפת C#.

ב. בפתרון בעיה זו יש שימוש בצובר מכפלה. מהו ערכו התחילי של צובר זה? הסבירו.

ג. ציינו באילו תבניות נוספות השתמשתם וכיצד **שילבתם** ביניהן.

שאלה 6

לקראת בחינות בגרות בית הספר מעוניין לתת שיעורי תגבור ל-75 תלמידים משתי כיתות שונות. שיעורי התגבור יכולים להתקיים לאחר הלימודים, בימים ראשון ושלישי. כדי להיערך הוחלט להעביר שאלון, שבעזרתו ניתן יהיה לקבוע כמה תלמידים רוצים שיעורי תגבור ומהו היום המועדף על רובם. כל תלמיד רשם בשאלון את מספר כיתתו (1 או 2) ואת מספר היום שאותו הוא מעדיף (1 או 3). תלמיד שאינו מעוניין בתגבור רשם את מספר כיתתו ואת המספר 0.

כתבו אלגוריתם, שהקלט שלו הוא הערכים שרשמו 75 התלמידים, והפלט שלו כולל את:

1. מספר התלמידים המעוניינים בשיעורי תגבור.
2. היום בשבוע שמרבית התלמידים מעדיפים.
3. מספר החדרים הדרושים לשיעורי התגבור (בכל חדר ילמדו 15 תלמידים לכל היותר).
4. מספר הכיתה שבה נרשמו פחות תלמידים לשיעורי התגבור.

א. ישמו את האלגוריתם בשפת C#.

ב. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.

שאלה 7

א. פתחו אלגוריתם אשר מקבל כקלט סדרת גילאים של זוגות נשואים (גיל הבעל וגיל האישה). סוף הקלט יצוין על ידי זוג הערכים 0 0. הפלט יהיה אחוז הזוגות, שבהם גיל האישה גבוה מגילו של הבעל. ישמו את האלגוריתם בשפת C#.

ב. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.

שאלה 8

א. פתחו אלגוריתם אשר מקבל כקלט רשימת מספרים שלמים תלת-ספרתיים המסתיימת במספר שאינו תלת-ספרתי, והפלט שלו הוא כמות המספרים שספרת העשרות שלהם זוגית או שסכומן של ספרות המספר הוא אי-זוגי. ישמו את האלגוריתם בשפת C#.

ב. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.

ממוצע של סדרת מספרים

נתבונן בבעיה האלגוריתמית הבאה:

> כתבו אלגוריתם שהקלט שלו כולל 31 מספרים המייצגים את הטמפרטורה בכל אחד מימי חודש יולי, והפלט שלו הוא ממוצע הטמפרטורות בחודש זה.

לפתרון הבעיה עלינו לחשב ממוצע של ערכים מספריים. בפרק 3 היכרנו את התבנית ***ממוצע של סדרת מספרים***, עבור סדרה בת שני מספרים. עתה נרחיב את התבנית עבור סדרת מספרים באורך כלשהו. כדי לחשב ממוצע של סדרת מספרים יש לחשב תחילה את הסכום הכולל של איברי הסדרה ולאחר מכן לחלקו במספר הערכים בסדרה. התבנית של ממוצע, כפי שמשמשת בפתרון בעיה זו, דומה לחישוב הממוצע שהוצג בפתרון בעיה 3 בפרק הלימוד.

נתבונן בשני האלגוריתמים הבאים, הראשון לפתרון הבעיה שלעיל, והשני הוא האלגוריתם שניתן בפרק 7 כפתרון לבעיה 3:

1. *אתחל את* sum *ב*-0
2. ***בצע*** 31 *פעמים:*
   2.1. *קלוט טמפרטורה של יום בחודש יולי ב*-temperature
   2.2. *הוסף לערכו של* sum *את ערכו של* temperature
3. *השם ב*-average *את ערכו של הביטוי החשבוני* sum/31
4. *הצג כפלט את ערכו של* average

---

1. *אתחל את* sum *ב*-0
2. *אתחל את* counterLarge *ב*-0
3. *קלוט ערך אי-שלילי שלם ב*-length
4. ***בצע*** length *פעמים:*
   4.1. *קלוט ערך ממשי ב*-num
   4.2. *הוסף את ערכו של* num *לסכום המצטבר השמור ב*-sum
   4.3. *אם ערכו של* num *גדול מ*-50
      4.3.1. *הגדל ב*-1 *את ערכו של* counterLarge
5. *חשב את ממוצע הערכים על ידי* sum/length *והשם ב*-average
6. *הצג כפלט את ערכו של* average
7. *הצג כפלט את ערכו של* counterLarge

בשני האלגוריתמים אותחל ערכו של sum ב-0 ואז חושב הסכום המצטבר בתוך sum. בתום הביצוע החוזר מתבצעת חלוקה של sum במספר הערכים שסוכמו. עבור הבעיה הנתונה מספר הערכים שסוכמו הוא 31, כמספר הימים בחודש יולי. בפתרון בעיה 3 שהוצגה בפרק הלימוד מספר הערכים המסוכמים הוא Length.

נציג את מאפייני התבנית לחישוב ממוצע עבור ביצוע חוזר התלוי בתנאי. שימו לב כי האלגוריתם של התבנית משלב במקביל תבנית צבירה (כדי לסכום את הערכים) ותבנית מנייה (כדי לדעת כמה ערכים נצברו).

שם התבנית: ממוצע של סדרת מספרים

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, תנאי conditionToInclude

מטרה: חישוב ממוצע של ערכי הקלט המקיימים את התנאי conditionToInclude. ביצוע החישוב תלוי בתנאי conditionToEnd.

אלגוריתם:

1. אתחל את count ב-0
2. אתחל את sum ב-0
3. קלוט ערך ב-element
4. **כל עוד לא** מתקיים conditionToEnd **בצע:**
   4.1. **אם** element מקיים את conditionToInclude
      4.1.1. הגדל את count ב-1
      4.1.2. הוסף לערכו של sum את element
   4.2. קלוט ערך ב-element
5. השם ב-average את ערכו של הביטוי החשבוני sum / count

יישום ב-#C:

```
count = 0;
sum = 0;
element = int.Parse(Console.ReadLine());
while (!onditionToEnd)
{
    if (conditionToInclude)
    {
        count++;
        sum += element;
    }
    element = int.Parse(Console.ReadLine());
}
average = (double) sum / count;
```

בדומה לדרך בה נהגנו עבור תבניות ***מנייה*** ו-***צבירה*** ניתן גם כאן להציג תת-תבנית המתאימה לביצוע חוזר שאורכו ידוע מראש. במקרה של ביצוע חוזר שאורכו ידוע מראש (והוא שווה ל-limit) אין צורך במניית הערכים. ולכן האלגוריתם הוא פשוט יותר:

1. *חשב את הסכום המצטבר של איברי סדרת הקלט שאורכה* limit
2. *חלק את הסכום המצטבר שחושב ב*-limit

שימו ♥: עבור התבנית של ממוצע של סדרת מספרים יש להניח שבקלט יש לפחות ערך אחד המקיים את התנאי, וזאת כדי להימנע מחלוקה ב-0.

ועוד שימו ♥: בדומה לתבניות ***מנייה*** ו-***צבירה***, גם בתבנית ***ממוצע*** ניתן להשתמש ללא תנאי (conditionToInclude), כלומר לחשב את ממוצע כל האיברים בסדרה.

שאלה 9

א. יישמו את האלגוריתם לפתרון הבעיה של הטמפרטורה הממוצעת בחודש יולי בשפת C#.

ב. שנו את התוכנית שכתבתם בסעיף א כך שתחשב את הטמפרטורה הממוצעת רק עבור הימים שבהם הטמפרטורה הייתה מעל 30 מעלות צלסיוס.

ג. מעוניינים להשוות בין הטמפרטורה הממוצעת בחודש יולי בשנה זו לבין הטמפרטורה הממוצעת בחודש יולי בשנה הקודמת, ולהציג כפלט הודעה המציינת באיזו שנה מבין השתיים הטמפרטורה הממוצעת הייתה גבוהה יותר. הקלט כולל קודם כל את הערכים הנתונים עבור חודש יולי בשנה זו, ואחר כך את הנתונים עבור חודש יולי בשנה הקודמת. השלימו את האלגוריתם החלקי:

1. *חשב את הממוצע של סדרת ערכי הקלט שאורכה ______ והשם אותו ב*- averageCurrentYear
2. *חשב את הממוצע של סדרת ערכי הקלט שאורכה ______ והשם אותו ב*- averageLastYear
3. *אם* ________
   3.1. *הצג כפלט* ________
4. *אחרת אם* ________
   4.1. *הצג כפלט* ________
5. *אחרת*
   5.1. *הצג כפלט* ________

שאלה 10

באוניברסיטת "בית הסטודנט" הוחלט כי בקורס אשר בו ממוצע ציוני הבחינה של הסטודנטים הוא מתחת ל-55 ייערך מבחן חוזר. נתון אלגוריתם חלקי שהקלט שלו הוא סדרת ציוני הסטודנטים בקורס מסוים, רשימה המסתיימת בזקיף 1-, והפלט שלו הוא הודעה האם יש לקיים מבחן חוזר.

1. ________

2. ________

3. *קלוט ערך ב-*mark

4. *כל עוד* mark *שונה מ-*1- *בצע:*

4.1 ________

4.2 *קלוט ערך ב-*mark

5. *השם ב-*average *את ערכו של הביטוי החשבוני*________

6. *אם*________

6.1 ________

א. השלימו את האלגוריתם.

ב. ניתן לכתוב את האלגוריתם בעזרת תבנית. השלימו:

1. *חשב את*______________ *סדרת ערכי הקלט המסתיימת על ידי הזקיף* 1-.

2. *אם* ________

2.1 ________

ג. ישמו את האלגוריתם בשפת C#.

שאלה 11

בשירות המטאורולוגי מחשבים בכל שנה את ממוצע המשקעים החודשי.

א. כתבו אלגוריתם שהקלט שלו הוא 12 מספרים המייצגים את כמות המשקעים בכל אחד מחודשי השנה, והפלט שלו הוא ממוצע המשקעים לחודש.

ב. הרחיבו את האלגוריתם כך שיציג כפלט גם את ממוצע ששת החודשים הראשונים בשנה ואת ממוצע ששת החודשים האחרונים בשנה.

ג. ישמו את האלגוריתם בשפת C#.

שאלה 12

במפעל מסוים התבקש מנהל מחלקת משאבי אנוש לבדוק האם גילן הממוצע של הנשים במפעל נמוך מגילם הממוצע של הגברים במפעל.

א. כתבו אלגוריתם שהקלט שלו הוא 100 זוגות נתונים המייצגים את מין העובד ('m' – גבר, 'f'- אישה) ושנת הלידה של העובד, והפלט שלו הוא הודעה מתאימה של מנהל מחלקת משאבי האנוש לאחר הבדיקה.

ב. הרחיבו את האלגוריתם שכתבתם בסעיף א כך שיציג כפלט את מספר העובדים הכולל שאמור לצאת לגמלאות ב-5 השנים הקרובות (הניחו כי גברים יוצאים לפנסיה בגיל 67 ונשים בגיל 62).

ג. ישמו את האלגוריתם בשפת #C.

ד. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן.

מציאת מקסימום או מינימום בסדרה

נתבונן בבעיה האלגוריתמית הבאה:

> למכרז זורמות הצעות כספיות שונות. ההצעה הכספית הגבוהה ביותר היא זו הזוכה במכרז. כתבו אלגוריתם שהקלט שלו הוא סדרה המסתיימת בזקיף 0, של מספרים המייצגים את ההצעות הכספיות, והפלט שלו הוא סכומה של ההצעה הזוכה. הניחו שבקלט יש לפחות הצעה אחת וכן הניחו שקיימת הצעה יחידה שזוכה.

לפתרון הבעיה עלינו לחשב את הערך המקסימלי מבין סדרת הערכים המספריים הנקלטים. בפרק 5 ראינו את התבנית של ***מציאת מקסימום ומינימום בסדרה*** בת שניים או שלושה ערכים. עתה נרחיב את התבנית עבור סדרה באורך כלשהו. התבנית של ***מציאת מקסימום ומינימום בסדרה***, כפי שמשמשת בפתרון בעיה זו, דומה לתבנית ***מציאת מקסימום ומינימום בסדרה*** שהוצגה בבעיה 5 בפרק הלימוד.

נתבונן בשני האלגוריתמים הבאים, הראשון לפתרון הבעיה שלעיל, והשני הוא האלגוריתם שניתן בפרק 7 לפתרון בעיה 5:

1. *קלוט הצעת מחיר ב-*max
2. *קלוט הצעת מחיר ב-*price
3. *כל עוד* price ≠ 0 *בצע:*
   3.1. *אם* price > max
      3.1.1. *השם ב-*max *את ערכו של* price
   3.2. *קלוט הצעת מחיר ב-*price
4. *הצג כפלט את ערכו של* max

---

1. *קלוט את מספר התשובות ב-*howMany
2. *קלוט את הסכום הראשון ב-*balance
3. *השם את ערכו של* balance *ב-*max
4. *בצע* howMany-1 *פעמים:*
   4.1. *קלוט את הסכום הבא ב-*balance
   4.2. *אם* max < balance
      4.2.1. *השם את ערכו של* balance *ב-*max
5. *הצג כפלט את ערכו של* max

בשתי הבעיות נדרשנו למצוא את הערך הגדול ביותר בסדרת ערכים. בשלב ראשון, קבענו באופן שרירותי כי הערך של המשתנה הראשון הוא המקסימלי ולכן נשמר ערכו במשתנה max. לאחר מכן, נבדקו על פי סדר קליטתם ערכיהם של שאר נתוני הקלט. עבור כל נתון קלט שערכו גדול יותר מהמקסימום הנוכחי, הוחלף ערכו של max בערכו של נתון הקלט. באופן דומה, עם שינויים קלים, עובד האלגוריתם עבור מציאת הערך הקטן ביותר בסדרה.

נפריד את מאפייני התבנית ***מציאת מקסימום ומינימום בסדרה*** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית ***מציאת מקסימום בסדרה*** ואחר כך נציג את מאפייני התבנית ***מציאת מינימום בסדרה***. עבור התבנית ***מציאת מקסימום*** נתייחס לביצוע חוזר באורך הידוע מראש ואילו עבור התבנית ***מציאת מינימום*** נתייחס לביצוע חוזר התלוי בתנאי. באופן דומה ניתן, עבור כל אחת מהתבניות, לטפל במקרה שאליו לא התייחסנו.

שם התבנית: מציאת מקסימום בסדרה

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט

מטרה: מציאת הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit

אלגוריתם:

1. קלוט ערך ב-max
2. בצע limit - 1 פעמים:
   2.1. קלוט ערך ב-element
   2.2 אם element > max
      2.2.1 השם ב-max את הערך של element

יישום ב-#C:

```
max = int.Parse(Console.ReadLine());
for (i = 2; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (element > max)
    {
        max = element;
    }
}
```

שם התבנית: מציאת מינימום בסדרה

נקודת מוצא: תנאי סיום conditionToEndMinSearch, ערכי הקלט

מטרה: מציאת הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי conditionTtoEndMinSearch

אלגוריתם:

1. קלוט ערך ב-min
2. קלוט ערך ב-element
3. כל עוד לא מתקיים conditionToEndMinSearch בצע:
   3.1. אם element < min
      3.1.1. השם ב-min את הערך של element
   3.2. קלוט ערך ב-element

יישום ב-#C:

```
min = int.Parse(Console.ReadLine());
element = int.Parse(Console.ReadLine());
while (!conditionToEndMinSearch)
{
    if (element < min)
    {
        min = element;
    }

    element = int.Parse(Console.ReadLine());
}
```

שימו ♥ : לפני ההוראה לביצוע חוזר כבר נקראים שני ערכים מהקלט. אנו נניח שהתנאי לסיום (conditionToEndMinSearch) יכול להתקיים עבור האיבר השני ואילך. לכן, למשל, אם תנאי הסיום מבטא קריאת זקיף, הרי שסדרת הקלט מכילה לפחות איבר אחד.

ועוד שימו ♥ : אם בקלט יש שני ערכים או יותר השווים למקסימום (או למינימום) אז האלגוריתמים עבור התבניות ***מציאת מקסימום*** ו-***מציאת מינימום*** מוצאים את המופע **הראשון** של המקסימום (או המינימום) בנתוני הקלט.

שאלה 13

א. השלימו את השימוש בתבנית עבור בעיית המכרז:

מצא מקסימום בסדרת ערכי הקלט ______________ *והצג את ערכו כפלט*

ב. ישמו את האלגוריתם של המכרז בשפת #C.

ג. הרחיבו את האלגוריתם כך שיוצג כפלט גם סכום ההצעה השנייה בגודלה. ישמו את האלגוריתם המתקבל בשפת #C.

שאלה 14

מדענים העוסקים בתחום המטאורולוגיה החליטו לגלות מהי הטמפרטורה הנמוכה ביותר ומהי הטמפרטורה הגבוהה ביותר בצהרי היום בחודש נובמבר באזור הקוטב הצפוני, בו הטמפרטורה תמיד מתחת ל-0 מעלות צלסיוס. לצורך כך, המדענים מדדו את הטמפרטורה במשך 30 יום במהלך חודש נובמבר, מדי יום ביומו, בשעה 12:00 בצהריים.

נתון אלגוריתם **שגוי**, שהקלט שלו הוא 30 ערכי הטמפרטורות והפלט שלו הוא הערך הנמוך ביותר והערך הגדול ביותר:

1. *אתחל את* maxTemperature *ב*-0

2. *אתחל את* minTemperature *ב*-0

3. *בצע* 30 *פעמים:*

3.1 *קלוט ערך טמפרטורה יומית ב*-temperature

3.2 *אם* temperature > maxTemperature

3.2.1 *השם ב*-maxTemperature *את* temperature

3.3 *אחרת אם* temperature < minTemperature

3.3.1 *השם ב*- minTemperature *את* temperature

4. *הצג כפלט "הטמפרטורה הנמוכה ביותר בחודש נובמבר היא"* minTemperature

5. *הצג כפלט "הטמפרטורה הגבוהה ביותר בחודש נובמבר היא"* maxTemperature

א. הסבירו במלים מדוע האלגוריתם שגוי.

ב. תקנו את האלגוריתם.

שאלה 15

קבוצת אנשים מוגדרת כ"הומוגנית" אם טווח הגילאים של חבריה אינו עולה על 5 שנים, אחרת הקבוצה מוגדרת כ"הטרוגנית".

א. כתבו אלגוריתם שהקלט שלו הוא מספר האנשים בקבוצה, n, ולאחר מכן סדרה של n מספרים המייצגים את גילאי החברים בקבוצה. הפלט שלו הוא הודעה האם הקבוצה "הומוגנית" או "הטרוגנית".

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

ג. ישמו את האלגוריתם בשפת C#.

מציאת ערך נלווה למקסימום או למינימום בסדרה

היכרנו את התבנית של מציאת מקסימום או מינימום בסדרה אולם לעיתים אנו מעוניינים לאו דווקא בערכים של המקסימום והמינימום אלא, למשל, במיקומם היחסי בקלט. דוגמה למקרה כזה הוא מציאת **היום** בחודש ינואר בו ירדה כמות משקעים מקסימלית.

נתבונן בבעיה האלגוריתמית הבאה:

> נרחיב את בעיית המכרז שהוצגה בסעיף הדן בתבנית ***מציאת מקסימום ומינימום בסדרה*** כך שהקלט הוא סדרה של **זוגות** נתונים, כאשר כל זוג מכיל את הקוד של מגיש ההצעה (מספר שלם) ואת סכום ההצעה הכספית. הסדרה מסתיימת עם קליטת הזקיף 0 כאיבר שני בזוג. הפלט של האלגוריתם הוא הקוד של מגיש ההצעה הזוכה.

לפתרון הבעיה עלינו לחשב את ההצעה המקסימלית מבין סדרת ההצעות הכספיות הנתונות בקלט, אבל אנו נדרשים **בנוסף** לשמור גם את קוד מגיש ההצעה. התבנית של ***מציאת ערך נלווה למקסימום ולמינימום בסדרה***, כפי שמשמשת בפתרון בעיה זו, דומה לתבנית ***מציאת ערך נלווה למקסימום ולמינימום בסדרה*** שהוצגה בבעיה 6 בפרק הלימוד.

נתבונן בשני האלגוריתמים הבאים, האחד לפתרון הבעיה שלעיל, והשני הוא האלגוריתם שניתן בפרק 7 לפתרון בעיה 6:

1. *קלוט קוד מגיש הצעה ב-*winCode
2. *קלוט הצעת מחיר ב-*max
3. *קלוט קוד נוסף של מגיש הצעה ב-*code
4. *קלוט הצעת מחיר נוספת ב-*price
5. ***כל עוד*** $price \neq 0$ ***בצע:***
   - 5.1. *אם* price > max
     - 5.1.1. *השם ב-*max *את ערכו של* price
     - 5.1.2. *השם ב-*winCode *את ערכו של* code
   - 5.2. *קלוט קוד מגיש הצעה ב-*code
   - 5.3. *קלוט הצעת מחיר ב-*price
6. *הצג כפלט את ערכו של* winCode

1. קלוט את אורך השיר הראשון ב-currentSongLength
2. השם את ערכו של currentSongLength ב-longest
3. השם את ערכו של currentSongLength ב-shortest
4. השם ל-placeLongest את הערך 1
5. השם ל-placeShortest את הערך 1
6. בצע 99 פעמים:
   - 6.1. קלוט את המספר הבא ב-currentSongLength
   - 6.2. אם currentSongLength > longest
     - 6.2.1. השם את currentSongLength ב-longest
     - 6.2.2. השם את מקומו של השיר הנוכחי ב-placeLongest
   - 6.3. אם currentSongLength < shortest
     - 6.3.1. השם את currentSongLength ב-shortest
     - 6.3.2. השם את מקומו של השיר הנוכחי ב-placeShortest
7. הצג כפלט את ערכו של placeLongest ואת ערכו של placeShortest

בשתי הבעיות נדרשנו למצוא ערך נלווה לערך הגדול ביותר בסדרת ערכים. בבעיה הנתונה בתחילת הסעיף הערך הנלווה הוא קוד מגיש ההצעה ובבעיה 6 בפרק הלימוד הערך הנלווה הוא מקומו של השיר הארוך ביותר.

כדי למצוא את הערך הנלווה לערך הגדול ביותר בסדרת ערכים נתבסס על התבנית למציאת מקסימום: בכל פעם שנעדכן את ערכו של max (וכמובן גם באתחול) נשמור במשתנה נוסף את ערכו של הערך הנלווה ל-max. באופן דומה, עם שינויים קלים, מתבצע האלגוריתם עבור מציאת הערך הנלווה לערך הקטן ביותר בסדרה.

בתבנית של מציאת ערך נלווה למקסימום ולמינימום בסדרה נדגים כערך הנלווה את מקום המקסימום או המינימום, אבל כאמור, ערך נלווה יכול להיות כל ערך שהוא, כמו הקוד הנלווה להצעה, בבעיה שלעיל.

נפריד את מאפייני התבנית ***מציאת ערך נלווה למקסימום ולמינימום בסדרה*** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית ***מציאת ערך נלווה למקסימום בסדרה*** ואחר כך נציג את מאפייני התבנית ***מציאת ערך נלווה למינימום בסדרה***. עבור התת-תבנית הראשונה נתייחס לביצוע חוזר באורך הידוע מראש, ואילו עבור השנייה נתייחס לביצוע חוזר התלוי בתנאי. משום הדמיון בין שתי התת-תבניות, קל לפתח אלגוריתם מתאים למקרה האחר, עבור כל אחת מהן.

שם התבנית: מציאת ערך נלווה למקסימום בסדרה

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט

מטרה: מציאת מיקום הערך הגדול ביותר בסדרת ערכי הקלט שאורכה הוא limit

אלגוריתם:

1. *קלוט ערך ב*-max
2. *אתחל את* placeOfMax *ב*-1
3. ***בצע*** limit -1 ***פעמים:***

   3.1. *קלוט ערך ב*-element

   3.2. *אם* element > max

   3.2.1. *השם ב*-max *את הערך של* element

   3.2.2 *השם ב*-placeOfMax *את מקומו של* element *בקלט*

יישום ב-#C:

```
max = int.Parse(Console.ReadLine());
placeOfMax = 1;
for (i = 2; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (element > max)
    {
        max = element;
        placeOfMax = i;
    }
}
```

שם התבנית: מציאת ערך נלווה למינימום בסדרה

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט

מטרה: מציאת מיקום הערך הקטן ביותר מבין ערכי הקלט, עד אשר מתקיים התנאי conditionTtoEnd

אלגוריתם:

1. קלוט ערך ב-min
2. אתחל את placeOfMin ב-1
3. קלוט ערך ב-element
4. אתחל את currentPlace ב-2
5. **כל עוד לא מתקיים conditionToEnd בצע:**
   5.1. אם element < min
      5.1.1. השם ב-min את הערך של element
      5.1.2. השם ב-placeOfMin את הערך של currentPlace
   5.2. הגדל את currentPlace ב-1
   5.3. קלוט ערך ב-element

יישום ב-#C:

```
min = int.Parse(Console.ReadLine());
placeOfMin = 1;
element = int.Parse(Console.ReadLine());
currentPlace = 2;
while (!conditionToEnd)
{
    if (element < min)
    {
        min = element;
        placeOfMin = currentPlace;
    }
    currentPlace++;
    element = int.Parse(Console.ReadLine());
}
```

שימו ♥ : לפני ההוראה לביצוע חוזר כבר נקראים שני ערכים מהקלט. אנו נניח שהתנאי לסיום (conditionToEnd) יכול להתקיים עבור האיבר השני ואילך. לכן, למשל, אם תנאי הסיום מבטא קריאת זקיף, הרי שסדרת הקלט מכילה לפחות איבר אחד.

לא פעם מופיע הערך המינימלי או המקסימלי בסדרה יותר מפעם אחת. בתבנית המוצגת מחושב מקום המופע **הראשון** של הערך המקסימלי או המינימלי. שינוי מזערי באלגוריתם יאפשר את חישוב מקום המופע **האחרון**. שאלה 16 מתייחסת לנקודה זאת. עם ההתקדמות בחומר הלימוד נתייחס גם לסוגיית מציאת **כל** המופעים של הערך המקסימלי או המינימלי.

שאלה 16

א. ישמו את האלגוריתם המורחב של המכרז בשפת C#.

ב. עתה, הניחו שבקלט יש שתי הצעות זוכות, כלומר שהערך הגבוה ביותר של הצעה הוצע על ידי שני מגישים. שנו את האלגוריתם, ואחר כך את התוכנית שכתבתם בסעיף א, כך שיוצג כפלט הקוד של מגיש ההצעה השנייה (במקום של הראשונה).

שאלה 17

חנוכייה אופיינית היא חנוכייה בת 9 קנים, שאחד מהם גבוה מן השאר ומשמש כשמש. חנוכיה כזו נחשבת "סימטרית" אם השמש ממוקם במרכז החנוכיה (ארבעה קנים ממוקמים מימינו של השמש וארבעה ממוקמים משמאלו). חנוכיה כזו נחשבת "צידית" אם השמש ממוקם בקצה אחד שלה (שאר הקנים ממוקמים משמאלו של השמש, או מימינו).בכל מקרה אחר, נחשבת החנוכיה "מיוחדת".

א. כתבו אלגוריתם שהקלט שלו הוא 9 מספרים המייצגים את גבהי הקנים של חנוכייה אופיינית, ונתונים לפי סדר מיקומם (משמאל לימין), והפלט שלו הוא הודעה המציינת את סוג החנוכיה.

ב. ישמו את האלגוריתם בשפת C#.

שאלה 18

בשכבת כיתות י' נערך מבחן משווה במדעי המחשב. המבחן נבדק על ידי המורים, וציוני התלמידים נרשמו בטופס ריכוז בו כל תלמיד מיוצג על ידי מספר סידורי. כל מורה התבקש למסור למרכז המגמה את הציון הגבוה ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה, וכן את הציון הנמוך ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה (המורה ידווח על תלמיד **אחד** מכל קטגוריה, גם אם יש יותר מתלמיד אחד שקיבל את הציון הגבוה ביותר בכיתה או יותר מתלמיד אחד שקיבל את הציון הנמוך ביותר בכיתה).

א. כתבו אלגוריתם שהקלט שלו הוא מספר התלמידים n בכיתה מסוימת, ואחריו סדרה של n ציוני התלמידים בכיתה, והפלט שלו הוא הציון הגבוה ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה, והציון הנמוך ביותר בכיתה ומספרו הסידורי של התלמיד שקיבל ציון זה.

ב. הרחיבו את האלגוריתם כך שיציג כפלט גם את הציון השני בגודלו.

ג. ישמו את האלגוריתם בשפת C#.

איסוף בקיזוז

נתבונן בבעיה האלגוריתמית הבאה:

> באוניברסיטה נערך סקר לבדיקת ההשערה: "מספר הסטודנטיות הלומדות קורסים המשלבים מחשב קטן יותר ממספר הסטודנטים בקורסים אלה". כתבו אלגוריתם שהקלט שלו הוא סדרה באורך 100 שאיבריה הם התו 'f' או התו 'm', המייצגת את מינם ('m'- סטודנט, 'f'- סטודנטית) של 100 סטודנטים הלומדים קורסים המשלבים מחשב. הפלט הוא הודעה האם ההשערה נכונה או אינה נכונה ביחס לנתונים.

בפתרון הבעיה אין חשיבות למספר הסטודנטים ומספר הסטודנטיות הלומדים קורסים המשלבים מחשב. לכן אין צורך במנייה רגילה. במקרה זה מתאים יותר למנות תוך כדי קיזוז. כלומר, נגדיר מונה יחיד, שערכו יגדל בכל פעם שנקלוט סטודנט, וערכו יקטן בכל פעם שנקלוט סטודנטית. בסיום התהליך יהיה במונה ההפרש בין מספר הסטודנטים למספר הסטודנטיות. נבדוק את ערכו של המונה: אם ערכו חיובי ההשערה נכונה. אחרת, ההשערה אינה נכונה. גם עבור שאלה 7.27 בפרק הלימוד ניתן לכתוב אלגוריתם המתבסס על איסוף בקיזוז.

נתבונן בשני האלגוריתמים הבאים, האחד לפתרון הבעיה שלעיל, והשני לפתרון שאלה 7.27:

1. אתחל את count ב-0	1. אתחל את count ב-0
2. בצע 100 פעמים:	2. קלוט קול בוחר ב-vote
2.1. קלוט מין הסטודנט ב-gender	3. כל עוד vote <> '#' בצע:
2.2. אם gender שווה ל- 'm'	3.1. אם vote שווה ל- 'A'
2.2.1. הגדל את count ב-1	3.1.1. הגדל את count ב-1
2.3 אחרת	3.2. אחרת
2.3.1. הקטן את count ב-1	3.2.1. הקטן את count ב-1
3. אם count גדול מ-0	3.3. קלוט קול בוחר ב-vote
3.1. הצג כפלט "ההשערה נכונה"	4. אם count גדול מ-0
4. אחרת	4.1. הצג כפלט "אלון זכה"
4.1. הצג כפלט "ההשערה לא נכונה"	5. אחרת
	4.2. הצג כפלט "אלון לא זכה"

התבנית של ***איסוף בקיזוז*** מתאימה גם לקיזוז באמצעות צבירה ולא רק לקיזוז באמצעות מנייה. בהמשך נראה שאלות, שבהן יש צורך לאסוף בקיזוז באמצעות צבירה.

נפריד את מאפייני התבנית ***איסוף בקיזוז*** לשתי תת-תבניות: ראשית, נציג את מאפייני התבנית ***איסוף בקיזוז באמצעות מנייה*** ואחר כך נציג את מאפייני התבנית ***איסוף בקיזוז באמצעות צבירה***. עבור התבנית הראשונה נתייחס לביצוע חוזר באורך הידוע מראש ואילו עבור השנייה נתייחס לביצוע חוזר התלוי בתנאי. כבמקרים קודמים, עבור כל אחת מהתבניות ניתן לפתח אלגוריתמים מתאימים גם למקרה האחר.

שם התבנית: איסוף בקיזוז באמצעות מנייה

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי לקיזוז conditionToDecrease

מטרה: איסוף בקיזוז באמצעות מנייה של ערכי הקלט מתוך סדרה שאורכה limit. הקיזוז מתבצע על פי התנאי conditionToDecrease

אלגוריתם:

1. אתחל את count ב-0
2. בצע limit פעמים:
   2.1. קלוט ערך ב-element
   2.2. אם element לא מקיים את conditionToDecrease
      2.2.1. הגדל את count ב-1
   2.3. אחרת
      2.3.1. הקטן את count ב-1

יישום ב-#C:

```
count = 0;
for (i = 1; i <= limit; i++)
{
    element = int.Parse(Console.ReadLine());
    if (!conditionToDecrease)
    {
        count++;
    }
    else
    {
        count--;
    }
}
```

שימו ♥ : בתבנית ***איסוף בקיזוז*** התנאי condition הוא הכרחי, ותפקידו, שליטה בקיזוז, שונה מתפקיד התנאים conditionToCount, conditionToSum או conditionToMult בתבניות ***מנייה*** ו-***צבירה***.

עתה נראה את התבנית של איסוף בקיזוז באמצעות צבירה, עבור שמשך ביצועה תלוי בתנאי.

שם התבנית: איסוף בקיזוז באמצעות צבירה

נקודת מוצא: תנאי סיום conditionToEnd, ערכי הקלט, ערך צבירה התחלתי initial, תנאי לקיזוז conditionToDecrease

מטרה: איסוף בקיזוז באמצעות צבירה של ערכי הקלט ושל הערך initial, משך הביצוע תלוי בתנאי conditionToEnd, והקיזוז מתבצע על פי התנאי conditionToDecrease

אלגוריתם:

1. *אתחל את* sum *ב*-initial
2. *קלוט ערך ב*-element
3. ***כל עוד לא מתקיים*** conditionToEnd ***בצע:***
   - 3.1 *אם* element *לא מקיים את* conditionToDecrease
     - 3.1.1 *הוסף ל*-sum *את ערכו של* element
   - 3.2 *אחרת*
     - 3.2.1 *הפחת מ*-sum *את ערכו של* element
   - 3.3 *קלוט ערך ב*-element

יישום ב-#C:

```
sum = initial;
element = int.Parse(Console.ReadLine());
while (!conditionToEnd)
{
    if (!conditionToDecrease)
    {
        sum += element;
    }
    else
    {
        sum -= element;
    }
    element = int.Parse(Console.ReadLine());
}
```

שאלה 19

ישמו את הפתרון לבעיית הסקר באוניברסיטה בשפת C#.

שאלה 20

נתונה סדרת ההוראות הבאה, שיש בה שימוש בתבנית ***איסוף בקיזוז***:

1. *אסוף בקיזוז באמצעות מנייה את איברי סדרת הקלט שאורכה 8, תוך קיזוז האיברים האי-זוגיים*
2. *הצג כפלט את הערך המתקבל*

א. תנו דוגמה לסדרת נתוני קלט של מספרים חיוביים שעבורה יוצג כפלט הערך 3.

ב. הסבירו בקצרה מהי מטרת ההוראה.

ג. השלימו את סדרת ההוראות הבאה, השקולה לסדרה הנתונה, אך משתמשת בתנאי לקיזוז אחר:

1. *אסוף בקיזוז באמצעות מנייה את איברי סדרת הקלט שאורכה 8, תוך קיזוז האיברים הזוגיים*
2. *הצג כפלט את* ______________

שאלה 21

נתון אלגוריתם שהקלט שלו הוא סדרת מספרים ממשיים:

1. *אתחל את* sum *ב-0*
2. *קלוט מספר ממשי ב-*num
3. *כל עוד* $num \neq 0$ *בצע:*
   - 3.1. *אם* $num > 0$
     - 3.1.1. *הוסף ל-*sum *את ערכו של* num
   - 3.2. *אחרת*
     - 3.2.1. *הפחת מ-*sum *את ערכו של* num
   - 3.3. *קלוט מספר ב-*num
4. *הצג כפלט את ערכו של* sum

א. תנו שתי דוגמאות קלט שונות לסדרות נתוני קלט של מספרים ממשיים, שעבורן יוצג כפלט הערך 0.

ב. הסבירו בקצרה מהי מטרת האלגוריתם.

ג. כתבו הוראה השקולה לאלגוריתם תוך שימוש בתבנית. השלימו:

אסוף בקיזוז באמצעות צבירה ל-________ של איברי סדרת הקלט המסתיימת בזקיף 0, תוך קיזוז __________________

שאלה 22

א. כתבו אלגוריתם שהקלט שלו הוא יתרת לקוח של בנק בתחילת החודש, ואחריו סדרה של מספרים, המסתיימת ב-0. כל מספר מייצג את סכום הפעולה: סכום חיובי מציין הפקדת הסכום בחשבון הבנק וסכום שלילי מציין משיכת הסכום מחשבון הבנק. הפלט של האלגוריתם הוא היתרה של הלקוח בסוף החודש.

ב. ישמו את האלגוריתם בשפת C#.

שאלה 23 (שאלה זו מתאימה לאחר לימוד סעיף 7.7 – קינון הוראות לביצוע חוזר)

מבחן במתכונת של שאלון אמריקני הועבר ל-30 סטודנטים של הקורס "דיוק בתשובות". במבחן 20 שאלות. ניקוד תשובות התלמידים התבצע באופן הבא: כל תשובה נכונה זיכתה ב-5 נקודות וכל תשובה שגויה גרמה להפחתה של 3 נקודות (בכל מקרה, לא ניתן לקבל ציון נמוך מ-0).

א. כתבו אלגוריתם, שהקלט שלו הוא 20 התשובות הנכונות ואחריהן תשובות של 30 הסטודנטים בקורס (20 תשובות לכל סטודנט). הפלט הוא מספר הסטודנטים שציונם "עובר" (לפחות 55).

ב. הרחיבו את האלגוריתם שכתבתם בסעיף א כך שיציג כפלט גם את אחוז התלמידים שקיבלו 0 במבחן.

ג. הרחיבו את האלגוריתם שכתבתם בסעיף ב כך שיציג כפלט גם את הציון הגבוה ביותר בבחינה ואת הציון הנמוך ביותר בבחינה (שאינו 0).

ד. ציינו מהן התבניות המשמשות לפתרון וכיצד שילבתם **ביניהן**.

ה. ישמו את האלגוריתם שכתבתם בסעיף ג בשפת C#.

שאלה 24 (*)

נתונים ארבעה ביטויים בוליאניים המשתמשים בתבניות, הבודקים אותו תנאי, עבור סדרת קלט נתונה:

1. *הערך המתקבל מ-איסוף בקיזוז באמצעות צבירה של איברי סדרת הקלט שאורכה 10, תוך קיזוז האיברים הזוגיים הוא זוגי*
2. *הערך המתקבל מ-מנייה של האיברים האי-זוגיים בסדרת הקלט שאורכה 10 הוא זוגי*
3. *הערך המתקבל מ-צבירת סכום איברי סדרת הקלט שאורכה 10 הוא זוגי*
4. *הערך המתקבל מ-צבירת סכום האיברים האי-זוגיים בסדרת הקלט שאורכה 10 הוא זוגי*

א. תנו דוגמה לסדרת מספרים שלמים חיוביים שעבורה הערך של כל אחד מהביטויים הבוליאניים יהיה true.

ב. נסחו במלים את התנאי שמבטאים הביטויים הבוליאניים.

ג. ישמו כל אחד מהשימושים השונים כקטעי תוכניות בשפת C#.

פירוק מספר חיובי לספרותיו

בפרק 4 הכרנו את התבנית של פירוק מספר דו-ספרתי חיובי לספרותיו. עתה נרחיב את התבנית עבור מספר שלם חיובי באורך כלשהו. פירוק המספר מתבסס על רעיון של הפרדת ספרת האחדות מהמספר, כך שהמספר שנותר קטן פי 10, וחוזר חלילה עד שלא נותרות ספרות במספר.

נראה את מאפייני התבנית ***פירוק מספר חיובי לספרותיו***:

שם התבנית: פירוק מספר חיובי לספרותיו

נקודת מוצא: מספר שלם חיובי num

מטרה: הצגה כפלט של ספרותיו של num

אלגוריתם:

1. *כל עוד* num *שונה מ*-0 *בצע*
   2. *הצג כפלט את ספרת האחדות של* num
   3. *הקטן את* num *פי* 10

יישום ב-#C:

```
while (num != 0)
{
    Console.WriteLine(num % 10);
    num /= 10;
}
```

שימו ♥: באלגוריתם של התבנית הצגנו כפלט את ספרותיו של num, אולם ניתן כמובן לבצע פעולות אחרות על ספרות המספר, כגון: מנייה, צבירה, ועוד. השאלות הבאות מתייחסות לבעיות אלגוריתמיות בהן נדרשות פעולות אחרות על ספרות המספר.

שאלה 25

א. פתחו אלגוריתם שהקלט שלו הוא מספר שלם חיובי, והפלט שלו הוא מספר הספרות במספר. ישמו את האלגוריתם בשפת #C.

ב. הרחיבו את האלגוריתם שכתבתם בסעיף א כך שיציג כפלט גם את סכום הספרות האי-זוגיות במספר .

ג. הרחיבו את האלגוריתם שכתבתם בסעיף ב כך שיציג כפלט גם הודעה המציינת אם יש במספר יותר ספרות המתחלקות ב-3 ללא שארית מאשר ספרות שאינן מתחלקות ב-3. ישמו את האלגוריתם המלא בשפת #C.

ד. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם המלא וכיצד **שילבתם** ביניהן.

שאלה 26

לפניכם קטע תוכנית בשפת #C, שהקלט שלו הוא מספר שלם חיובי, והפלט שלו אמור להיות מספר הספרות במספר. קטע התוכנית שגוי.

```
num = int.Parse(Console.ReadLine());
sum = num % 10;
while ((num / 10) > 0)
      sum += num % 10;
Console.WriteLine(sum);
```

א. הביאו דוגמת קלט שעבורה יתקבל הפלט הדרוש.

ב. הביאו דוגמת קלט שעבורה לא יתקבל הפלט הדרוש.

ג. הסבירו במלים מדוע קטע התוכנית שגוי.

ד. תקנו את קטע התוכנית כך שישיג את מטרתו עבור כל קלט חוקי.

שאלה 27

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num ומספר חד-ספרתי place המייצג מקום. הפלט שלו הוא הספרה הנמצאת במקום place במספר num (מיקום הספרות מתחיל **מימין**). אם אין במספר place ספרות יוצג כפלט הערך 1- .

למשל, עבור הקלט 2 17489 יוצג כפלט הערך 8, ועבור הקלט 6 17489 יוצג כפלט הערך 1- .

ב. ישמו את האלגוריתם בשפת #C.

שאלה 28

ספרת שורש של מספר היא ספרה בין 1 ל-9 המתקבלת מתהליך של חיבור חוזר של ספרות המספר עד אשר מתקבל מספר חד-ספרתי. למשל, ספרת השורש של המספר 30486 היא 2, כיוון שסכום ספרות המספר המקורי הוא 20 וסכום הספרות של 20 הוא 2.

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם חיובי והפלט שלו הוא ספרת השורש של המספר הנתון.

ב. ישמו את האלגוריתם בשפת #C.

שאלה 29

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם חיובי והפלט שלו הוא ההפרש בין הספרה הגדולה ביותר במספר לבין הספרה הקטנה ביותר במספר.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

ג. ישמו את האלגוריתם בשפת #C.

בניית מספר

בפרק 4 הכרנו את התבנית של בניית מספר כאשר התמקדנו בבניית מספר דו-ספרתי משתי ספרות. עתה נרחיב את התבנית לבניית מספר שלם באורך כלשהו מספרות הנקלטות בזו אחר זו. כדי להרכיב מספר מספרות צריך בכל שלב בלולאה להגדיל את המספר פי 10 ולחבר לו את ספרת הקלט החדשה. כך עד לסיום הקלט.

ניתן להרחיב את התבנית, למשל, על ידי בניית מספר ממספרים דו-ספרתיים, ואז בכל שלב יגדל המספר פי 100 ויתווסף לו המספר הדו-ספרתי התורן מהקלט.

נציג עבור התבנית ***בניית מספר*** אלגוריתם עבור ביצוע חוזר באורך הידוע מראש. ניתן להתאימו גם למקרים בהם סיום הבנייה תלוי בתנאי.

שם התבנית: בניית מספר

נקודת מוצא: אורך סדרת נתוני הקלט limit, ספרות הקלט

מטרה: בניית מספר מספרות הקלט

אלגוריתם:

1. אתחל את num ב-0
2. **בצע** limit **פעמים**:
   2.1. קלוט ספרה ב-digit
   2.2. השם ב-num את הערך של הביטוי החשבוני num * 10 + digit

יישום ב-#C:

```
num = 0;
for (i = 1; i <= limit; i++)
{
    digit = int.Parse(Console.ReadLine());
    num = num * 10 + digit;
}
```

שאלה 30

נתון האלגוריתם הבא:

1. קלוט ספרה ב-digit
2. קלוט מספר ב-limit
3. אתחל את num ב-0
4. **בצע** limit **פעמים**:

4.1. השם ב-num את הערך של הביטוי החשבוני num * 10 + digit

4.2. הצג כפלט את num

א. מה יהיה הפלט עבור הקלט 3 5?

ב. תנו דוגמה לקלט שעבורו הפלט יהיה 6.

ג. הסבירו בקצרה מהי מטרת האלגוריתם.

שאלה 31

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num, ספרה digit ומקום place. הפלט שלו הוא המספר שמתקבל מהחלפת הספרה הנמצאת במקום place (מימין) במספר num בספרה digit.

למשל, עבור הקלט 6 4 78342 יוצג כפלט הערך 76342, ועבור הקלט 9 1 13608 יוצג כפלט הערך 13608.

ב. ישמו את האלגוריתם בשפת C#.

שאלה 32

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num והפלט שלו הוא המספר המתקבל מ-num על ידי היפוך סדר ספרותיו.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

ג. ישמו את האלגוריתם בשפת C#.

האם כל הערכים בסדרה מקיימים תנאי?

נתבונן בבעיה האלגוריתמית הבאה:

> במסגרת הפעילות של עידוד "הקורא הצעיר" הוחלט להעניק פרס לבית-ספר שבו כל תלמידיו קראו לפחות ספר קריאה אחד במהלך חופשת הקיץ. פתחו אלגוריתם שהקלט שלו הוא מספר הספרים שקרא כל אחד מתלמידי בית-ספר "אמירים" שבסיומה הזקיף 1- (עבור תלמיד שלא קרא כלל ספרים ייקלט הערך 0), והפלט שלו הוא הודעה המציינת אם הפרס יוענק לבית-הספר.

לפתרון הבעיה עלינו לעבור על כל נתוני הקלט, כלומר, לבדוק עבור כל תלמיד האם קרא ספרי קריאה במהלך החופשה או לא. אבל, אם קיים לפחות תלמיד אחד שלא קרא ספר קריאה אז לבית-הספר לא יוענק הפרס ולכן אין טעם להמשיך ולבדוק את מספר ספרי הקריאה שקראו שאר תלמידי בית-הספר. אלגוריתם זה הוא תיאור של התבנית ***האם כל הערכים בסדרה מקיימים תנאי?*** . גם עבור שאלה 7.53 בפרק הלימוד ניתן לכתוב אלגוריתם המתבסס על תבנית זו.

נתבונן בשני האלגוריתמים הבאים, הראשון לפתרון הבעיה שלעיל, והשני לפתרון שאלה 7.53:

1. *אתחל את* allReaders *ב*-true
2. *קלוט מספר ספרי קריאה לתלמיד ב*-books
3. ***כל עוד*** books ≠ -1 ***וגם*** *ערכו של* allReaders *הוא* true ***בצע:***
   - 3.1. ***אם*** *ערכו של* books *הוא* 0
     - 3.1.1. *השם ב*-allReaders *את הערך* false
   - 3.2. ***אחרת***
     - 3.2.1. *קלוט מספר ספרי קריאה לתלמיד ב*-books
4. ***אם*** *ערכו של* allReaders *הוא* true
   - 4.1. *הצג כפלט "לבית-הספר אמירים מוענק הפרס"*

---

1. *אתחל את* allEven ב-true
2. *אתחל את* sum ב-0
3. *אתחל את* howMany *ב*-1
4. ***כל עוד*** howMany ≤ 20 ***וגם*** *ערכו של* allEven *הוא* true ***בצע:***
   - 4.1. *קלוט מספר חיובי ב*-num
   - 4.2. *הגדל את ערכו של* howMany *ב*-1
   - 4.3. ***אם*** num מספר אי-זוגי
     - 4.3.1. *השם ב*-allEven *את הערך* false
   - 4.4. ***אחרת***
     - 4.4.1. *הגדל את* sum *ב*-num

5. אם allEven
5.1. הצג כפלט את ערכו של sum
6. אחרת
6.1. הצג כפלט "הקלט אינו חוקי"

משמעות התבנית ***האם כל הערכים בסדרה מקיימים תנאי?*** היא בדיקה של קיום תנאי עבור **כל** ערכי הסדרה. מתבצעת סריקה של ערכי הקלט בזה אחר זה, ואם אחד הערכים אינו מקיים את התנאי אין טעם בהמשך הסריקה. בבעיה הנתונה אם ערכו של books שווה ל-0 אז נמצא תלמיד שלא קרא אף לא ספר קריאה אחד ולכן אין צורך להמשיך בסריקה. בשאלה 7.53 אם בקלט יש מספר אי-זוגי אז אין טעם להמשיך בבדיקת שאר ערכי המספרים.

נציג את מאפייני התבנית ***האם כל הערכים בסדרה מקיימים תנאי?*** עבור סדרה שאורכה ידוע מראש. ניתן להתאים את האלגוריתם למקרה שבו אורך הסדרה אינו ידוע מראש, בדומה לאלגוריתם שניתן לבעיה שלעיל.

שם התבנית: האם כל הערכים בסדרה מקיימים תנאי?

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט, תנאי condition

מטרה: קביעת הערך true אם **כל** הערכים בסדרה מקיימים את התנאי condition וקביעת הערך false אם **קיים** ערך אחד בסדרה שאינו מקיים את התנאי

אלגוריתם:

1. אתחל את all ב-true
2. אתחל את howmany ב-1
3. **כל עוד** howmany ≤ limit **וגם ערכו של** all הוא true **בצע:**
3.1. קלוט ערך ב-element
3.2. הגדל את ערכו של howMany ב-1
3.3. אם element אינו מקיים את condition
3.3.1. השם ב-all את הערך false

יישום ב-#C:

```
all = true;
howmany = 1;
while ((howmany <= limit) && (all))
{
    element = int.Parse(Console.ReadLine());
    howmany++;
    if (condition)// אינו מקיים את התנאי element
    {
        all = false;
    }
}
```

התבנית ***האם כל הערכים בסדרה מקיימים תנאי?*** מחשבת ערך בוליאני שערכו true או false. אנו משתמשים במשתנה בוליאני all שערכו התחילי הוא true, כלומר, ההנחה ***התחילית*** היא שכל ערכי הסדרה אכן מקיימים את התנאי. אם במהלך הסריקה אחד הערכים אינו מקיים את התנאי אז ההנחה התחילית שלנו מתבדה ולכן ערכו של all מקבל ערך false ו הסריקה מסתיימת.

שאלה 33

א. רשמו הוראה השקולה לאלגוריתם של "הקורא הצעיר" תוך שימוש בתבנית החדשה. השלימו:

האם כל הערכים בסדרת הקלט המסתיימת ב-__________ מקיימים את התנאי_________?

ב. ישמו את האלגוריתם בשפת C#.

שאלה 34

א. כתבו אלגוריתם שהקלט שלו הוא מספר שלם וחיובי num והפלט שלו הוא ההודעה "ספרות זהות" אם כל ספרות המספר זהות וההודעה "ספרות שונות" אם לא כל הספרות זהות.

ב. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

ג. ישמו את האלגוריתם בשפת C#.

שאלה 35

לפניכם קטע תוכנית בשפת C#, שהקלט שלו הוא סדרה של מספרים שלמים וחיוביים. סוף הקלט מצוין על ידי המספר 1-. קטע התוכנית אמור להציג כפלט את ההודעה "כל המספרים הם כפולות של 6" אם 6 מחלק של כל המספרים בסדרת הקלט. קטע התוכנית שגוי.

```
num = int.Parse(Console.ReadLine());
while (num != -1)
{
    ok = (num % 6 == 0)
    num = int.Parse(Console.ReadLine());
}
if (ok)
{
    Console.WriteLine ("All numbers are multiples of 6");
}
```

א. תנו דוגמה לסדרת קלט (לפחות 5 ערכים) עבורה לא ניתן להבחין שקטע התוכנית שגוי.

ב. תנו דוגמה לסדרת קלט (לפחות 5 ערכים) עבורה ניתן להבחין שקטע התוכנית שגוי.

ג. הסבירו במילים מדוע קטע התוכנית שגוי.

ד. תקנו את קטע התוכנית כך שיבצע את מטרתו עבור כל סדרת קלט.

האם קיים ערך בסדרה המקיים תנאי?

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 1: כתבו אלגוריתם, שהקלט שלו הוא סדרה של 15 מספרים שלמים, והפלט שלו הוא הודעה המציינת אם קיים בסדרת הקלט מספר שלילי.

בעיה 2: כתבו אלגוריתם, שהקלט שלו הוא סדרה של תווים, המסתיימת בזקיף '*', והפלט שלו הוא הודעה המציינת אם נקלטה בסדרת התווים אחת מאותיות ה-ABC או אחת מאותיות ה-abc.

לפתרון שתי הבעיות עלינו לעבור על נתוני הקלט עד למציאת ערך המקיים את התנאי. בבעיה 1 התנאי הוא האם ערך הקלט הוא מספר שלילי ובבעיה 2 התנאי הוא האם נקלטה אחת מאותיות ה-ABC או אחת מאותיות ה-abc. אם קיים ערך אחד המקיים את התנאי אז יש להפסיק את הסריקה כי ערך מתאים כבר נמצא. אלגוריתם זה הוא תיאור של התבנית ***האם קיים ערך בסדרה המקיים תנאי?***.

נתבונן בשני האלגוריתמים לפתרון בעיה 1 ובעיה 2:

1. *אתחל את* found *ב-*false
2. *אתחל את* howmany *ב-*1
3. ***כל עוד*** howmany ≤ 15 ***וגם*** *ערכו של* found *שווה ל-*false ***בצע:***
    3.1 *קלוט מספר שלם ב-*num
    3.2 *הגדל את ערכו של* howmany *ב-*1
    3.3 ***אם*** num < 0
        3.3.1 *השם ב-*found *את הערך* true
4. ***אם*** found
    4.1 *הצג כפלט "קיים מספר שלילי בסדרת הקלט"*
5. ***אחרת***
    5.1 *הצג כפלט "לא קיים מספר שלילי בסדרת הקלט"*

---

1. *אתחל את* found *ב-*false
2. *קלוט תו ב-*ch
3. ***כל עוד*** ch ≠ '*' ***וגם*** *ערכו של* found *הוא* false ***בצע:***

3.1. אם (ch ≥ 'a' וגם ch <= 'z') או (ch ≥ 'A' וגם ch ≤ 'Z')

3.1.1. השם ב-found את הערך true

3.2. אחרת

3.2.1. קלוט תו ב-ch

4. אם found

4.1. הצג כפלט "קיימת אות ב-ABC או ב-abc בסדרת הקלט"

5. אחרת

5.1. הצג כפלט "לא קיימת אות ב-ABC או ב-abc בסדרת הקלט"

נציג את מאפייני התבנית ***האם קיים ערך בסדרה המקיים תנאי?*** עבור סדרה שאורכה אינו ידוע מראש. ניתן להתאים את האלגוריתם למקרה שבו אורך הסדרה ידוע מראש, בדומה לאלגוריתם שניתן לבעיה 1 שלעיל.

שם התבנית: האם קיים ערך בסדרה המקיים תנאי?

נקודת מוצא: תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition

מטרה: קביעת הערך true אם **קיים** ערך בסדרה המקיים את התנאי condition וקביעת הערך false אם **כל** הערכים בסדרה אינם מקיימים את התנאי

אלגוריתם:

1. אתחל את found ב-false
2. קלוט ערך ב-element
3. כל עוד התנאי toEnd לא מתקיים וגם ערכו של found הוא false בצע:

3.1. אם element מקיים את condition

3.1.1. השם ב-found את הערך true

3.2. אחרת

3.2.1. קלוט ערך ב-element

יישום ב-#C:

```
found = false;
element = int.Parse(Console.ReadLine());
while (!toEnd && !found)
{
    if (condition)  // מקיים את התנאי element
    {
        found = true;
    }
    element = int.Parse(Console.ReadLine());
}
```

משמעות התבנית ***האם קיים ערך בסדרה המקיים תנאי?*** היא בדיקה של קיום תנאי עבור ערכים בסדרה. מתבצעת סריקה של ערכי הקלט בזה אחר זה, ואם אחד הערכים אכן מקיים את התנאי אין טעם בהמשך הסריקה. התבנית מחשבת ערך בוליאני שערכו true או false. לכן, אנו משתמשים במשתנה בוליאני found שערכו התחילי הוא false. כלומר, ההנחה **התחילית** היא שאין ערך בסדרה המקיים את התנאי. אם במהלך הסריקה אחד הערכים אכן מקיים את התנאי אז ההנחה התחילית שלנו מתבדה, מאחר שמצאנו ערך המקיים את התנאי. לכן found יקבל ערך true והסריקה תסתיים.

שאלה 36

א. כתבו הוראה השקולה לאלגוריתמים של כל אחת מהבעיות 1 ו-2 תוך שימוש בתבנית ***האם קיים ערך בסדרה המקיים תנאי?***. השלימו:

בעיה 1:

1. אם קיים ערך בסדרת הקלט שאורכה_________ המקיים _________

1.1. הצג כפלט "קיים מספר שלילי בסדרת הקלט"

2. אחרת

2.1. הצג כפלט "לא קיים מספר שלילי בסדרת הקלט"

בעיה 2:

1. אםקיים ערך בסדרת הקלט שמסתיימת ב-_________ המקיים _________

1.1. הצג כפלט "קיימת אות ב-ABC או ב-abc בסדרת הקלט"

2. אחרת

2.1. הצג כפלט "לא קיימת אות ב-ABC או ב-abc בסדרת הקלט"

ב. ישמו את האלגוריתמים לפתרון הבעיות 1 ו-2 בשפת C#.

ג. עבור כל אחד מהאלגוריתמים ניתן לכתוב אלגוריתם שקול תוך שימוש בתבנית ***האם כל הערכים בסדרה מקיימים תנאי?***. השלימו את ההוראות הבאות עבור בעיה 1, וכתבו הוראות מתאימות, המשתמשות בתבנית זו, לבעיה 2.

בעיה 1:

1. אם לא כל הערכים בסדרה שאורכה_________ מקיימים את התנאי _____________

1.1. הצג כפלט "קיים מספר שלילי בסדרת הקלט"

2. אחרת

2.1. הצג כפלט "לא קיים מספר שלילי בסדרת הקלט"

שאלה 37

א. כתבו אלגוריתם, שהקלט שלו מספר שלם חיובי והפלט שלו הוא הודעה המציינת אם קיימת במספר הספרה 0.

ב. שנו את האלגוריתם שכתבתם בסעיף א כך שיציג הודעה המציינת אם קיימת ספרה זוגית במספר. ציינו איזו תבנית **שילבתם** בפתרון הבעיה.

שאלה 38

בכיתה י׳ המורה העלתה הצעה לעשות קומזיץ לגיבוש 39 תלמידי הכיתה. ההצעה תתקבל רק אם אף תלמיד לא יתנגד להגיע לקומזיץ. כל תלמיד התבקש לרשום על דף את האות 'y' אם הוא מסכים להצעה או את האות 'n' אם הוא מתנגד להצעה. יש לכתוב אלגוריתם, שהקלט שלו הוא סדרה של 39 תווים כאשר התו 'y' מייצג את הסכמת התלמיד להצעה והתו 'n' מייצג את התנגדות התלמיד להצעה. הפלט הוא הודעה המציינת אם ההצעה התקבלה או לא.

לפניכם שני אלגוריתמים לפתרון הבעיה:

אלגוריתם 1:

1. *אם קיים ערך בסדרת הקלט שאורכה 39 המקיים את התנאי (answer שווה ל- 'n')*
   1.1. *הצג כפלט את ההודעה "ההצעה לא התקבלה"*
2. *אחרת*
   2.1. *הצג כפלט את ההודעה "ההצעה התקבלה"*

אלגוריתם 2:

1. *אם כל הערכים בסדרת הקלט שאורכה 39 מקיימים את התנאי (answer שווה ל- 'y')*
   1.1. *הצג כפלט את ההודעה "ההצעה התקבלה"*
2. *אחרת*
   2.1. *הצג כפלט את ההודעה "ההצעה לא התקבלה"*

ישמו את שני האלגוריתמים בשפת C#.

מציאת כל הערכים בסדרה המקיימים תנאי

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 1: כתבו אלגוריתם, שהקלט שלו הוא סדרה של 25 זוגות מספרים שלמים, והפלט שלו הוא כל זוגות המספרים המקיימים את היחס של מספרים עוקבים.

בעיה 2: כתבו אלגוריתם, שהפלט שלו הוא כל המספרים מ-1 עד 100 המקיימים את הכללים של "7 בום", כלומר: מתחלקים ב-7 ללא שארית או כוללים את הספרה 7.

לפתרון שתי הבעיות עלינו לעבור על כל הערכים בסדרה ולהציג כפלט את כל הערכים המקיימים את התנאי. בבעיה 1 התנאי הוא יחס של עוקב בין כל זוג מספרים ברשימה ובבעיה 2 התנאי הוא קיום הכללים של "7 בום". יש דמיון מסוים בין תבנית זו לשתי התבניות האחרונות, אך במקרה זה איננו יכולים להפסיק את הסריקה לפני שנגיע אל סיום סדרת הקלט. אלגוריתם זה הוא תיאור של התבנית ***מציאת כל הערכים בסדרה המקיימים תנאי***.

נתבונן בשני האלגוריתמים לפתרון בעיה 1 ובעיה 2:

1. בצע 25 פעמים:
2. קלוט זוג מספרים שלמים ב-num1 וב-num2
   2.1. אם num1 ו-num2 הם ערכים עוקבים או num2 ו-num1 הם ערכים עוקבים
      2.1.1. הצג כפלט את הערכים של num1 ושל num2

---

1. עבור כל מספר i חיובי שלם הקטן מ-100 בצע:
   1.1. אם 7 מחלק של i או ספרת העשרות של i שווה ל-7 או ספרת האחדות של i שווה ל-7
      1.1.1. הצג כפלט את הערך של i

בתבנית ***מציאת כל הערכים בסדרה המקיימים תנאי*** מתבצעת סריקה של כל ערכי הקלט בזה אחר זה. בכל פעם שנמצא ערך שמקיים את התנאי מבצעים עליו פעולה כגון הצגה כפלט, מנייה, פעולה חשבונית.

נציג את מאפייני התבנית ***מציאת כל הערכים בסדרה המקיימים תנאי*** עבור סדרה שאורכה אינו ידוע מראש. ניתן להתאים את האלגוריתם למקרה שבו אורך הסדרה ידוע מראש, בדומה

לאלגוריתם שניתן לבעיה 1 ו-2 שלעיל. מאפייני התבנית מוגדרים לפי ביצוע פעולת קלט עבור כל איבר שנמצא מקיים את התנאי. ניתן להתאימם למקרה בו נדרשת פעולה אחרת, כפי שקורה בכמה מהשאלות הבאות.

שם התבנית: מציאת כל הערכים בסדרה המקיימים תנאי

נקודת מוצא: תנאי לסיום הסדרה toEnd, ערכי הקלט, תנאי condition

מטרה: הצגה כפלט של כל ערכי הקלט המקיימים את התנאי condition

אלגוריתם:

1. *קלוט ערך ב-*element
2. *כל עוד התנאי* toEnd *לא מתקיים בצע:*
   3.1. *אם* element *מקיים את* condition
      3.1.1. *הצג כפלט את ערכו של* element
   3.2. *קלוט ערך ב-*element

יישום ב-#C:

```
element = int.Parse(Console.ReadLine());
while (!toEnd)
{
    if (condition)
    {
        Console.WriteLine(element);
    }
    element = int.Parse(Console.ReadLine());
}
```

שאלה 39

א. רשמו הוראה השקולה לאלגוריתם לפתרון בעיה 1 תוך שימוש בתבנית. השלימו :

מצא את כל הערכים בסדרת הקלט שאורכה ________ המקיימים את התנאי ________

ב. ישמו את האלגוריתם בשפת #C.

ג. רשמו הוראה השקולה לאלגוריתם לפתרון בעיה 2 תוך שימוש בתבנית. השלימו :

מצא את כל הערכים בסדרת הקלט שאורכה ________ המקיימים את התנאי ________

ד. ישמו את האלגוריתם בשפת #C.

שאלה 40

תלמידי שכבה י' בבית-ספר "היובל" השתתפו בתחרויות יום ספורט היתולי, במהלכו כל משתתף צבר נקודות. תוצאתו של תלמיד שלא השתתף היא 0. למען גיבוש הכיתות הוחלט להעניק "יום כיף" לכיתות שבהן השתתפו **כל** התלמידים בתחרויות של יום הספורט.

נתון אלגוריתם חלקי, שהקלט שלו הוא תוצאות התחרויות של כל אחת מ-10 הכיתות בבית-הספר. לכל כיתה נקלטת סדרת התוצאות של כל התלמידים עד לקליטת הזקיף 1-. הפלט של האלגוריתם הוא מספרי הכיתות שזכו ב"יום כיף".

1. עבור כל i חיובי ושלם הקטן או שווה ל-10 בצע:
   1.1. אתחל את prize ב-________
   1.2. קלוט תוצאת תלמיד ב-result
   1.3. כל עוד ________ וגם ________ בצע:
      1.3.1. אם ________
         1.3.1.1. השם ב-prize את הערך false
      1.3.2. אחרת
         1.3.2.1. ________
2. אם ________
   2.1. הצג כפלט ________

א. השלימו את האלגוריתם.

ב. ציינו מהן התבניות ה**משולבות** באלגוריתם.

ג. בהנהלת בית-הספר החליטו להעניק פרס נוסף "גיבושון" לכיתה שתלמידיה צברו את מירב הנקודות בתחרות. הרחיבו את האלגוריתם כך שיציג כפלט את מספר הכיתה שזכתה בפרס "גיבושון". איזו תבנית **שילבתם** עתה בפתרון הבעיה האלגוריתמית?

ד. ישמו את האלגוריתם בשפת #C.

שאלה 41

א. כתבו אלגוריתם, שהקלט שלו הוא סדרה של מספרים חיוביים (סוף הסדרה מצוין ע"י מספר שלילי), והפלט שלו הוא כל המספרים המתחלקים בסכום ספרותיהם וכמות המספרים המקיימים תנאי זה.

לדוגמה: עבור סדרת הקלט 7- 18 103 83 550 110 4000: המספרים 18 550 110 ו-4000 מתחלקים בסכום ספרותיהם, לכן הפלט הוא רשימת המספרים 4000 110 550 18 והודעה המציינת כי כמות המספרים המתחלקים בסכום ספרותיהם: 4

ב. ישמו את האלגוריתם בשפת #C.

מעבר על זוגות סמוכים בסדרה

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 1: כתבו אלגוריתם, שהקלט שלו הוא סדרה של 18 מספרים שלמים, והפלט שלו הוא הודעה האם הסדרה מסודרת בסדר עולה ממש.

בעיה 2: כתבו אלגוריתם, שהקלט שלו הוא סדרה של מספרים שלמים חיוביים, המסתיימת עם קליטת הזקיף 0, והפלט שלו הוא סדרה שבה מכפלות כל זוגות המספרים הזוגיים הנקלטים זה אחר זה בסמיכות. לדוגמה, עבור הקלט: 0 2 4 9 8 4 6 5 3 תוצג כפלט הסדרה: 8 32 24, המתקבלת ממכפלת 6 ב-4, מכפלת 4 ב-8 ומכפלת 4 ב-2.

לפתרון שתי הבעיות יש לעבור על כל זוגות הערכים הסמוכים בסדרה. בבעיה 1 יש לבדוק עבור כל זוג ערכים סמוכים אם הוא מקיים את היחס של הערך הראשון (משמאל) קטן יותר מהערך השני. בבעיה 2 יש להציג כפלט את המכפלות של זוגות מספרים סמוכים שאיבריהם זוגיים. זה הוא, למעשה, תיאור של התבנית ***מעבר על זוגות סמוכים בסדרה***.

נתבונן בשני האלגוריתמים הבאים לפתרון בעיות 1 ו-2:

1. *אתחל את* ordered *ב-*true
2. *אתחל את* howmany *ב-*1
3. *קלוט מספר שלם ב-*beforeLast
4. *כל עוד* howmany < 18 *וגם ערכו של* ordered *הוא* true
   - 4.1. *קלוט מספר שלם ב-*last
   - 4.2. *אם* beforelast ≥ last
     - 4.2.1. *השם ב-*ordered *את הערך* false
   - 4.3. *אחרת*
     - 4.3.1. *השם ב-*beforeLast *את הערך של* last
5. *אם* ordered
   - 5.1. *הצג כפלט את ההודעה "הסדרה מסודרת בסדר עולה ממש"*
6. *אחרת*
   - 6.1. *הצג כפלט "הסדרה אינה מסודרת בסדר עולה ממש"*

---

1. *קלוט מספר שלם ב-*beforeLast
2. *קלוט מספר שלם ב-*last
3. *כל עוד* last<>0 *בצע:*

3.1. *אם* beforeLast *הוא מספר זוגי וגם* last *הוא מספר זוגי*

3.1.1. *הצג כפלט את הערך של הביטוי החשבוני* beforeLast * last

3.2. *השם ב-*beforeLast *את הערך של* last

3.3. *קלוט מספר שלם ב-*last

בפתרון בעיות אלגוריתמיות רבות יש לבצע פעולות על זוגות ערכים הסמוכים בסדרה. עבור סדרת נתוני קלט באורך limit יש בסך הכול limit-1 זוגות ערכים סמוכים. למעשה ניתן להסתכל על כל זוג ערכים סמוכים כאל פריט אחד. הרעיון שבבסיסם של שני האלגוריתמים הוא שבכל שלב המשתנים beforeLast ו-last מכילים איברי זוג סמוך. המעבר לזוג הבא מתבצע כך: beforeLast מקבל את ערכו של last (כלומר, האיבר שקודם היה איבר שני בזוג הוא עכשיו איבר ראשון בזוג הסמוך) וב-last נקלט ערך נוסף. כך עד לסיום הקלט.

לשם המחשה נדגים בתבנית הצגה כפלט של סכום זוגות ערכים סמוכים בסדרה. ניתן כמובן לבצע פעולות אחרות על ערכי כל זוג, כגון מנייה, צבירה ועוד. בהמשך נדגים כמה מהאפשרויות השונות דרך שאלות.

נציג את מאפייני התבנית ***מעבר על זוגות סמוכים בסדרה*** עבור סדרה שאורכה ידוע מראש. אפשר לערוך התאמות למקרה בו אורך סדרת הקלט אינו ידוע מראש, כמו בבעיה 2 לעיל.

שם התבנית: מעבר על זוגות סמוכים בסדרה

נקודת מוצא: אורך סדרת נתוני הקלט limit, ערכי הקלט

מטרה: הצגה כפלט של סכומי כל זוגות הערכים הסמוכים בסדרת הקלט שאורכה limit

אלגוריתם:

1. *קלוט ערך ב-*beforeLast

2. ***בצע*** limit-1 ***פעמים***

2.1. *קלוט ערך ב-*last

2.2. *הצג כפלט את הערך של הביטוי החשבוני* beforeLast + last

2.3. *השם ב-*beforeLast *את הערך של* last

יישום ב-#C:

```
beforeLast = int.Parse(Console.ReadLine());
for (i = 1; i <= limit-1; i++)
{
      last = int.Parse(Console.ReadLine());
      Console.WriteLine(beforeLast + last);
      beforeLast = last;
}
```

שימו ♥ : אנו מניחים שבסדרת הקלט לפחות שני ערכים.

שאלה 42

ישמו את האלגוריתמים של שתי הבעיות הנתונות בשפת #C .

שאלה 43

א. כתבו אלגוריתם, שהקלט שלו הוא 25 תווים והפלט שלו הוא כל הזוגות הסמוכים בסדרת הקלט שמכילים אותיות קטנות ב-abc עוקבות. ישמו את האלגוריתם בשפת #C.

ב. ציינו באילו תבניות נוספות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

שאלה 44

כתבו אלגוריתם שהקלט שלו הוא סדרת מספרים שלמים המסתיימת בזקיף 1-, והפלט הוא כל המספרים בקלט השווים לסכום שני המספרים הקודמים להם בסדרה (שני המספרים הראשונים לא יודפסו).

למשל, עבור סדרת הקלט הבאה (משמאל לימין) : 1- 89 6 5 13 0 13 11 2 9 4 יוצגו כפלט המספרים : 13 13 11.

ישמו את האלגוריתם בשפת #C.

שאלה 45

כתבו תוכנית בשפת #C, שהקלט שלה הוא מספר שלם וחיובי num והפלט שלה הוא הודעה האם ספרות המספר מסודרות בסדר עולה ממש.

ציינו באילו תבניות נוספות השתמשתם בכתיבת התוכנית וכיצד **שילבתם** ביניהן.

תבניות – פרק 10

מערך מונים

נתבונן בבעיה האלגוריתמית הבאה:

כתבו אלגוריתם שהקלט שלו הוא 58 ספרות והפלט שלו הוא מספר הפעמים שהופיעה כל ספרה בקלט.

למעשה, עבור פתרון הבעיה האלגוריתמית אנו צריכים 10 מונים – מונה לכל ספרה. כדי להימנע מסירבול, נגדיר מערך CountDigitsArr בגודל 10. מצייני המערך הם מ-0 ועד 9, ולכן כל תא במערך מייצג את המונה של המציין. כך, למשל, תפקידו של התא countDigitsArr[3] הוא למנות את מספר הפעמים שמופיעה הספרה 3 בקלט. באופן כללי תפקידו של התא countDigitsArr[digit] הוא למנות את מספר הפעמים שמופיעה הספרה digit בקלט. זהו אלגוריתם המתבסס על התבנית ***מערך מונים***. התבנית של ***מערך מונים***, כפי שמשמשת בחלקו הראשון של פתרון בעיה זו, דומה לחלקו הראשון של הפתרון שהוצג בבעיה 5 בפרק 10.

נתבונן בחלק הראשון של כל אחד משני האלגוריתמים הללו, לפתרון הבעיה שלעיל ולפתרון בעיה 5 בפרק 10:

1. *בצע 58 פעמים:*
   1.1. *קלוט ספרה ב-*digit
   1.2. *הגדל את*countDigitsArr[digit] *ב-*1

---

1. *קלוט את מספר המתמודדים ב-*numOfSingers
2. *קלוט מספר מועמד ב-*vote
3. *כל עוד* vote ≠ -1 *בצע:*
   3.1. *הגדל ב-*1 *את המונה של מתמודד מספר* vote
   3.2. *קלוט מספר מועמד ב-*vote

בשני קטעי האלגוריתמים האלה ניתן לזהות תבנית ***מנייה***, אלא שהיא מופעלת על כמה מונים הפועלים במקביל, ושמורים במערך אחד.

נציג את מאפייני התבנית ***מערך מונים***, עבור ביצוע חוזר התלוי בתנאי. ניתן להתאים את מאפייני התבנית למקרים בהם משך הביצוע ידוע מראש, בדומה לקטע האלגוריתם לפתרון הבעיה שהוצגה לעיל.

שם התבנית: מערך מונים

נקודת מוצא: תנאי סיום toEnd, סדרת ערכים, ביטוי חשבוני whichCounter המקשר בין ערך בסדרה למציין של המערך

מטרה: בניית מערך מונים עבור ערכי הסדרה, בעוד משך הבנייה תלוי בביטוי toEnd.

אלגוריתם:

1. אתחל את איברי המערך countElements ב-0
2. השם את הערך הבא בסדרה ב-element
3. **כל עוד לא מתקיים התנאי toEnd בצע:**
   3.1. הגדל את countElements[whichCounter] ב-1
   3.2. השם את הערך הבא בסדרה ב-element

יישום ב-#C:

```
element = הערך הבא בסדרה;
while (!toEnd)
{
    countElements[whichCounter]++;
    element = הערך הבא בסדרה;
}
```

שימו ♥:

- ♦ התבנית כללית ואינה מפרטת מהיכן מגיעים איברי הסדרה. ערכים אלה יכולים, למשל, להתקבל כקלט או מקריאת ערכי מערך אחר.
- ♦ המשתנה element מתייחס לאיבר התורן בסדרה. למרות שאין זה מופיע במפורש באלגוריתם שניתן עבור התבנית וביישום שלו, הרי פעולת המנייה תלויה ב-element: הביטוי whichCounter תלוי ב-element ומקשר בין ערכו לערך של מציין במערך. הנה כמה דוגמאות לביטויים אפשריים עבור whichCounter:
  - element (כלומר, הערך הבא בסדרה משמש כמציין למערך, בדומה לפתרון של הבעיה שהוצגה בתחילת הסעיף).
  - ***ספרת העשרות*** של element.
  - element - 1 (בדומה לביטוי המשמש לגישה למערך בפתרון בעיה 5 בפרק 10).
- ♦ בדומה לתבנית ***מנייה***, גם התבנית של ***מערך מונים*** כוללת אתחול של המונים ל-0. אין לכך התייחסות ביישום, משום שבשפת #C מתבצע אתחול אוטומטי של איברי מערך שלמים ל-0.

שאלה 1

א. ישמו את הפתרון המלא לבעיית מניית מופעי הספרות בשפת #C.

ב. שנו את התוכנית שכתבתם בסעיף א כך שהקלט שלה יהיה 58 מספרים מהתחום 1 עד 10, והפלט שלה יהיה מספר הפעמים שהופיע כל אחד מהמספרים. מהו הביטוי החשבוני בו השתמשתם כדי לקשר בין ערך תורן למציין במערך?

ג. שנו את התוכנית שכתבתם בסעיף א כך שהקלט שלה יהיה 58 מספרים שלמים חיוביים. הפלט שלה יהיה עבור כל אחת מהספרות 0 עד 9 את מספר הפעמים שהופיעה כספרת האחדות של מספר בקלט. מהו הביטוי החשבוני בו השתמשתם כדי לקשר בין ערך תורן למציין במערך?

שאלה 2

נתון המערך `arr` ובו ערכים שלמים ונתון קטע התוכנית הבא:

```
int[] counts = new int[2];
for (i = 0; i < arr.length; i++)
{
    element = arr[i];
    counts[element % 2]++;
}
for (i = 0; i < counts.length; i++)
{
    Console.WriteLine(counts[i]);
}
```

א. מה יוצג כפלט עבור המערך `arr` הבא:

arr[0]	arr[1]	arr[2]	arr[3]	arr[4]	arr[5]	arr[6]	arr[7]
90	68	198	5	11	34	89	6

ב. תנו דוגמה למערך `arr` בגודל 10, שעבורו יוצגו כפלט הערכים: 3 7.

ג. מהי מטרת קטע התוכנית?

ד. הציעו קטע תוכנית השקול לקטע התוכנית הנתון **ללא** שימוש בתבנית ***מערך מונים***. ציינו באילו תבניות השתמשתם עבור כתיבת קטע התוכנית.

שאלה 3

א. כתבו אלגוריתם, שהקלט שלו הוא סדרת תווים המסתיימת בתו '*' והפלט שלו הוא מספר המופעים של כל אחת מאותיות ה-abc (כלומר, אותיות הא"ב האנגלי הקטנות).

ב. מהו הביטוי החשבוני בו השתמשתם עבור התאמת אותיות ה-abc למצייני המערך?

ג. ישמו את האלגוריתם בשפת #C.

מערך צוברים

נתבונן בשתי הבעיות האלגוריתמיות הבאות:

בעיה 1: תלמידי שכבה י' בבית הספר "נוף-ים", המונה 7 כיתות, החליטו על מבצע לסיוע למשפחות נזקקות: כל תלמיד בשכבה יעבוד בעבודות מזדמנות (כגון שטיפת-מכוניות, בייביסיטר וכו'), ואת הסכום שירוויח יעביר לקופת הכיתה. בסוף החודש יעבירו הנציגים מכל אחת מהכיתות דיווח של הסכום המצטבר שנאסף אל נציג השכבה. נציג השכבה יעביר את הסכום הכולל לנציג ועד השכונה. יש לציין כי כל התלמידים בשכבה נרתמו להצלחת המבצע. כתבו אלגוריתם שהקלט שלו הוא 247 זוגות מספרים עבור כל תלמידי השכבה, כאשר המספר הראשון מייצג את מספר הכיתה של התלמיד והמספר השני מייצג את הסכום שהעביר התלמיד לקופת הכיתה. הפלט של האלגוריתם הוא הסכום המצטבר של כל אחת מהכיתות וכן הסכום הכולל שהצטבר בשכבה.

בעיה 2: בחנות הנעליים "נעל לכל" מעוניינים לדעת מהו הפדיון היומי מסך כל המכירות בכל אחת מ-5 המחלקות בחנות (המחלקות ממוספרות מ-1 עד 5). כתבו אלגוריתם שהקלט שלו הוא סדרה של זוגות מספרים, כאשר המספר הראשון בכל זוג מייצג את מספר המחלקה והמספר השני מייצג את המחיר של סכום הקניה של לקוח. סוף הקלט יסומן על ידי זוג שהמספר הראשון בו הוא 0. הפלט של האלגוריתם הוא התפלגות המכירות לפי מחלקות (כלומר, עבור כל מחלקה הפדיון היומי הכולל שלה).

עבור הפתרונות של שתי הבעיות האלגוריתמיות דרושים לנו כמה צוברים. בבעיה 1 אנו צריכים שבעה צוברים עבור הסכומים המצטברים לכל אחת מהכיתות, ובבעיה 2 אנו צריכים חמישה צוברים (צובר סכום מכירות לכל מחלקה). כדי להימנע מסירבול, נגדיר מערך שגודלו כמספר הצוברים הנדרש (7 עבור בעיה 1 ו-5 עבור בעיה 2). בבעיה 1 מצייני המערך מייצגים את מספרי הכיתות, ובבעיה 2 הם מייצגים את מספרי המחלקות. כל תא במערך מייצג צובר המתאים למציין. זהו אלגוריתם המתבסס על התבנית ***מערך צוברים***. הרעיון בבסיסה של התבנית ***מערך צוברים*** דומה לתבנית של מערך המונים. ההבדל הוא באופי הפעולה על כל תא במערך: מנייה במערך המונים לעומת צבירה במערך הצוברים.

נתבונן בשני האלגוריתמים הבאים, הראשון לפתרון בעיה 1 והשני לפתרון בעיה 2:

1. *בצע* 247 *פעמים:*

   1.1. *קלוט מספר כיתה ב-*classNum *וסכום כסף שהרוויח תלמיד ב-* sumStudent

   1.2. *הגדל את*[classNum-1]sumClasses *ב-*sumStudent

2. אתחל את sumTotal ל-0

3. **עבור כל** i שלם בתחום 0 עד 6 **בצע**:

   3.1. הצג כפלט: הסכום המצטבר לכיתה i+1 הוא: sumClasses[i]

   3.2. הגדל את sumTotal ב-sumClasses[i]

4. הצג כפלט: הסכום המצטבר לשכבה הוא sumTotal

---

1. קלוט מספר מחלקה ב-departmentNum וסכום הקניה של לקוח ב-price

2. **כל עוד** departmentNum שונה מ-0 **בצע**:

   2.1. הגדל את sumDepartments[departmentNum-1] ב-price

   2.2. קלוט מספר מחלקה ב-departmentNum וסכום הקניה של לקוח ב-price

3. **עבור כל** i שלם בתחום 0 עד 4 **בצע**:

   3.1. הצג כפלט: הפדיון היומי למחלקה i+1 הוא: sumDepartments[i]

בשני קטעי האלגוריתמים האלה ניתן לזהות תבנית **צבירה**, אלא שהיא מופעלת על כמה צוברים הפועלים במקביל, ושמורים במערך אחד.

נציג את מאפייני התבנית ***מערך צוברים***, עבור סדרת ערכים שאורכה ידוע מראש. ניתן להתאים את מאפייני התבנית למקרים בהם משך הביצוע תלוי בתנאי, בדומה לפתרון בעיה 1. התבנית מתייחסת לצבירה על ידי סכום, אך ניתן להציג מאפיינים של תבנית דומה המתייחסת לצבירה על ידי מכפלה.

שם התבנית: מערך צוברים

נקודת מוצא: אורך סדרת הערכים limit, סדרת ערכים, ביטוי חשבוני whichSum המקשר בין ערך בסדרה למציין של המערך

מטרה: בניית מערך צוברים עבור ערכי הסדרה, שאורכה limit

אלגוריתם:

1. אתחל את ערכי מערך הצוברים
2. בצע limit פעמים:
   2.1. השם את הערך הבא בסדרה ב-element
   2.2. השם את הערך לצבירה ב-value
   2.3. הוסף ל-sumElements[whichSum] את ערכו של value

יישום ב-#C:

```
for (i = 1; i <= limit; i++)
{
    element = הערך הבא בסדרה;
    value = הערך לצבירה;
    sumElements[whichSum] += value;
}
```

שימו ♥ : גם התבנית של ***מערך צוברים***, הוצגה באופן כללי, כך שהיא מתאימה למגוון בעיות:

- ♦ לא נקבעו במפורש הערכים לאתחול. במקרה הפשוט, מאותחלים כל הצוברים ל-0. במקרה כזה, ביישום בשפת #C אין צורך לבצע אתחול מפורש, משום שב-#C מתבצע אתחול אוטומטי של איברי מערך שלמים ל-0.
- ♦ בדומה לתבנית ***מערך מונים***, גם התבנית של ***מערך צוברים*** אינה מפרטת מהיכן מגיעים איברי הסדרה. ערכים אלה יכולים, למשל, להתקבל כקלט או מקריאת ערכי מערך אחר.
- ♦ התבנית של ***מערך צוברים*** גם אינה מפרטת כצד מחושבים הערכים לצבירה. גם ערכים אלה יכולים, למשל, להתקבל כקלט או מקריאת ערכי מערך אחר, וייתכן כי ניתן לחשב, עבור כל איבר element, בעזרת ביטוי חשבוני מסוים התלוי ב-element, את ערך הצבירה המתאים לו value.
- ♦ כמו בתבנית ***מערך מונים***, גם בתבנית של ***מערך צוברים*** המשתנה element מתייחס לאיבר התורן בסדרה, ולמרות שאין זה מופיע במפורש באלגוריתם שניתן עבור התבנית וביישום שלו, הרי פעולת הצבירה תלויה ב-element: הביטוי whichSum תלוי ב-element ומקשר בין ערכו לערך של מציין במערך.

שאלה 4

ישמו את שני האלגוריתמים עבור בעיות 1 ו-2 בשפת #C.

שאלה 5

לאור הצלחת המבצע לסיוע למשפחות נזקקות החליטה הנהלת בית-הספר "נוף ים" להעניק יום טיול לנגב לתלמידי הכיתה שאספה את הסכום המירבי. יש להרחיב את האלגוריתם כך שיוצג כפלט גם מספר הכיתה שזכתה ביום הטיול. אם יש יותר מכיתה אחת שאספה את הסכום המירבי יוענק יום טיול לכולן.

נתון האלגוריתם הבא לפתרון הבעיה:

1. אתחל את איברי המערך sumClasses ל-0
2. **בצע** 247 **פעמים:**
   2.1. קלוט מספר כיתה ב-classNum וסכום כסף שהרוויח תלמיד ב-sumStudent
   2.2. הגדל את sumClasses[classNum-1] ב-sumStudent
3. אתחל את sumTotal ל-0
4. **עבור כל** i שלם בתחום 0 עד 6 **בצע:**
   4.1. הצג כפלט: הסכום המצטבר לכיתה i+1 הוא: sumClasses[i]
   4.2. הגדל את sumTotal ב-sumClasses[i]
5. הצג כפלט: הסכום המצטבר לשכבה הוא sumTotal
6. אתחל את max ל-sumClasses[0]
7. **עבור כל** i שלם בתחום 0 עד 6 **בצע:**
   7.1. **אם** sumClasses[i] > max
      7.1.1. השם ב-max את הערך של sumClasses[i]
8. **עבור כל** i שלם בתחום 0 עד 6 **בצע:**
   8.1. **אם** ערכו של sumClasses[i] שווה ל-max
      8.1.1. הצג כפלט: כיתה מספר i+1 זוכה ביום טיול לנגב

א. ציינו מהן התבניות **המשולבות** באלגוריתם.

ב. ישמו את האלגוריתם בשפת #C (כזכור, אין צורך ליישם את שלב אתחול המערך).

שאלה 6

בחנות הנעליים "נעל לכל" מעוניינים לדעת את הפדיון מסך כל המכירות בכל אחד מימות השבוע.

יש לכתוב אלגוריתם שהקלט שלו הוא סדרות של ספרים ממשיים (המסתיימות בזקיף 0) – סדרה

עבור כל יום מהימים א' עד ו'. המספרים הממשיים מייצגים את סכומי הקניות של הלקוחות. הפלט של האלגוריתם הוא התפלגות המכירות לפי ימים (כלומר, עבור כל יום הפדיון הכולל שלו). נתון האלגוריתם הבא לפתרון הבעיה:

1. אתחל את איברי המערך sumDays ל-0
2. עבור כל day שלם בתחום 0 עד 5 בצע:
   2.1. קלוט סכום קניה ב-price
   2.2. כל עוד ערכו של price שונה מ-0 בצע:
      2.2.1. הגדל את sumDays[day] ב-price
      2.2.2. קלוט סכום קניה ב-price
3. עבור כל day שלם בתחום 0 עד 5 בצע:
   3.1. הצג כפלט: הפדיון היומי עבור יום day+1 הוא sumDays[day]

א. באלגוריתם ישנו שימוש בתבנית ***מערך צוברים***. הציעו אלגוריתם השקול לאלגוריתם הנתון **ללא** שימוש בתבנית ***מערך צוברים***. ציינו באילו תבניות השתמשתם עבור כתיבת האלגוריתם.

ב. ישמו את האלגוריתם הנתון בשפת C# (כזכור, אין צורך ליישם את שלב אתחול המערך).

שאלה 7

א. במוסד לביטוח רפואי בוצע מחקר על צריכת 150 תרופות על ידי החולים המבוטחים. לכל תרופה מספר סידורי בין 1 ל-150. פתחו אלגוריתם, שהקלט שלו הוא סדרת שלשות של ערכים, כאשר כל שלשה מייצגת גיל של מבוטח, מספר סידורי של התרופה שנצרכה והכמות שלה (מספר יחידות). סדרת הקלט מסתיימת עם קליטת הזקיף -1 כגיל המבוטח. הפלט של האלגוריתם הוא קבוצת הגילאים הצורכת כמות תרופות כוללת גדולה ביותר, מבין 4 קבוצות הגילאים הבאות: קבוצת גילאי 0 עד 10, קבוצת גילאי 11 עד 30, קבוצת גילאי 31 עד 50, קבוצת גילאי 51 ומעלה. ייתכן שיותר מקבוצת גיל אחת צורכת את כמות התרופות הכוללת הגדולה ביותר.

שימו לב: לצורך פתרון סעיף זה אין צורך להבחין בין סוגי התרופות השונים.

ב. הרחיבו את האלגוריתם כך שיוצגו כפלט גם מספרי התרופות שלא נצרכו כלל.

ג. ציינו באילו תבניות השתמשתם בכתיבת האלגוריתם וכיצד **שילבתם** ביניהן.

ד. ישמו את האלגוריתם בשפת C#.

חישוב שכיח

נתבונן בבעיה האלגוריתמית הבאה:

כתבו אלגוריתם שהקלט שלו הוא 58 ספרות והפלט שלו הוא הספרה שהופיעה בקלט הכי הרבה פעמים. אם יש יותר מספרה אחת כזאת, יש להציג את כולן.

לצורך פתרון הבעיה צריך קודם כל למנות את מספר המופעים של כל אחת מהספרות. לשם כך, נשתמש בתבנית ***מערך מונים***. לאחר מכן, עלינו לחשב את הערך המקסימלי מבין ערכי המונים. זהו למעשה הערך השכיח, כלומר הערך שהופיע מספר פעמים רב ביותר. לצורך כך נוכל להשתמש בתבנית של ***מציאת מקסימום בסדרה***, כדי למצוא את הערך המקסימלי, ולאחר מכן להשתמש בתבנית של ***מציאת כל הערכים בסדרה המקיימים תנאי***, כאשר התנאי הוא שוויון לערך המקסימלי, תוך הצגה כפלט של מצייני הערכים שנמצאו. יש לשים לב, שעלינו לבצע שינויים קלים בתבניות ***מציאת מקסימום בסדרה*** ו-***מציאת כל הערכים בסדרה המקיימים תנאי*** מאחר שהערכים בסדרה אינם נקראים מהקלט, אלא שמורים ב***מערך מונים***.

הנה האלגוריתם המלא:

1. בצע 58 פעמים:

*1.1. קלוט ספרה ב-*digit

*1.2. הגדל את*countDigitsArr[digit] *ב-*1

*2. השם ב-*max *את*countDigitsArr[0]

3. עבור כל i *שלם בתחום* 0 *עד* 9 *בצע:*

*3.1. אם*countDigitsArr[i] *גדול מ-*max

*3.1.1. השם ב-*max *את הערך של* countDigitsArr[i]

4. עבור כל i *שלם בתחום* 0 *עד* 9 *בצע:*

*4.1. אם*countDigitsArr[i] *שווה ל-*max

4.1.1. הצג כפלט: ספרה i *הופיעה הכי הרבה פעמים*

נציג את מאפייני התבנית ***חישוב שכיח*** עבור ביצוע חוזר שאורכו ידוע מראש. ניתן להתאים את מאפייני התבנית לביצוע חוזר התלוי בתנאי.

שם התבנית: חישוב שכיח

נקודת מוצא: אורך סדרת הערכים limit, סדרת ערכים, ביטוי חשבוני whichSum המקשר בין ערך בסדרה למציין של המערך

מטרה: הצגה כפלט של הערך השכיח או של הערכים השכיחים בסדרת הקלט, שאורכה הוא limit

אלגוריתם:

1. אתחל את ערכי מערך המונים
2. בצע limit פעמים:
   2.1. השם את הערך הבא בסדרה ב-element
   2.2. הגדל את sumElements[whichCount] ב-1
3. אתחל את max בערכו של countElements[0]
4. עבור כל i שלם בתחום מ-1 עד אורך המערך countElements פחות 1 בצע:
   4.1. אם countElements[i] > max
      4.1.1. השם ב-max את הערך של countElements[i]
5. עבור כל i שלם בתחום מ-1 עד אורך המערך countElements פחות 1 בצע:
   5.1 אם ערכו של countElements[i] שווה ל-max
      5.1.1. הצג את i כפלט

יישום ב-#C:

```
for (i = 1; i <= limit; i++)
{
    element = הערך הבא בסדרה;
    sumElements[whichCount]++;
}
max = countElements[0];
for (i = 1; i < countElements.length; i++)
{
    if (countElements[i] > max)
    {
        max = countElements[i];
    }
}
for (i = 0; i < countElements.length; i++)
{
    if (countElements[i] == max)
    {
        Console.WriteLine(counts[i]);
    }
 }
```

שימו ♥ : באלגוריתם של התבנית בחרנו להציג כפלט את הערך השכיח או הערכים השכיחים אבל ניתן לבצע על ערכים אלו פעולות חישוביות שונות כגון מנייה, צבירה וכו'.

שאלה 8

המורה אמיר החליט לבצע כמה עיבודים סטטיסטיים על ציוני 40 תלמידיו ב"יסודות מדעי המחשב". אמיר חילק את ציוני התלמידים, הנעים בין 0 ל-100, ל-10 קבוצות באופן הבא: קבוצת הציונים בין 0 ל-10 (קבוצת ציונים ראשונה), קבוצת הציונים בין 11 ל-20 (קבוצת ציונים שנייה) וכך הלאה עד קבוצת הציונים בין 91 ל-100 (קבוצת ציונים עשירית).

א. כתבו אלגוריתם, שהקלט שלו הוא ציוני 40 התלמידים והפלט שלו הוא מספר התלמידים בכל קבוצת ציונים, וכן קבוצת הציונים השכיחה (תיתכן יותר מקבוצה אחת).

ב. הרחיבו את האלגוריתם שכתבתם בסעיף א כך שיציג כפלט את טווח הציונים, כלומר, ההפרש בין הציון הגבוה ביותר במבחן לבין הציון הנמוך ביותר במבחן. ציינו באילו תבניות נוספות השתמשתם וכיצד **שילבתם** ביניהן.

ג. הרחיבו את האלגוריתם שכתבתם בסעיף ב כך שיציג כפלט הודעה האם הציונים מתפלגים סימטרית, כלומר, מספר התלמידים בקבוצה הראשונה שווה למספר התלמידים בקבוצה העשירית, מספר התלמידים בקבוצה השנייה שווה למספר התלמידים בקבוצה התשיעית וכן הלאה. ציינו באילו תבניות נוספות השתמשתם וכיצד **שילבתם** ביניהן.

ד. הרחיבו את האלגוריתם שכתבתם בסעיף ג כך שאם הציונים מתפלגים סימטרית, על האלגוריתם להציג הודעה האם הציונים מתפלגים סימטרית בצורת פעמון, כלומר, הערכים ב-5 קבוצות הציונים הראשונות מהווים סדרה עולה ממש, והערכים ב-5 קבוצות הציונים האחרונות מהווים סדרה יורדת ממש. ציינו באילו תבניות נוספות השתמשתם וכיצד **שילבתם** ביניהן.

ה. ישמו את האלגוריתם המורחב שכתבתם בסעיף ד בשפת C#.

הזזה מעגלית בסדרה

בפרק 3 הראינו את התבנית ***הזזה מעגלית בסדרה*** עבור סדרה בת שני ערכים. עתה נרחיב את התבנית עבור סדרת ערכים באורך כלשהו. נזכיר כי ישנם שני סוגי הזזות מעגליות: הזזה מעגלית שמאלה והזזה מעגלית ימינה.

נפריד את מאפייני התבנית ***הזזה מעגלית בסדרה*** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית ***הזזה מעגלית שמאלה בסדרה*** ואחר כך נציג את מאפייני התבנית ***הזזה מעגלית ימינה בסדרה***.

שם התבנית: הזזה מעגלית שמאלה בסדרה

נקודת מוצא: סדרת ערכים במערך elements שאורכו length

מטרה: הזזה מעגלית שמאלה של ערכי המערך

אלגוריתם:

1. *השם ב-*temp *את* elements[0]
2. *עבור כל* i *שלם בתחום* 0 עד length-2 ***בצע:***
   2.1. *השם ב-*elements[i] *את הערך של* elements[i+1]
3. *השם ב-*elements[length-1] *את הערך של* temp

יישום ב-#C:

```
temp = elements[0];
for (i = 0; i <= elements.length - 2; i++)
{
    elements[i] = elements[i + 1];
}
elements[elements.length - 1] = temp;
```

שם התבנית: הזזה מעגלית ימינה בסדרה

נקודת מוצא: סדרת ערכים במערך elements שאורכו length

מטרה: הזזה מעגלית ימינה של ערכי המערך

אלגוריתם:

1. *השם ב-*temp *את* elements[length-1]
2. *עבור כל* i *שלם בתחום* length-1 *עד* 1 *(בסדר יורד) בצע:*
   2.1. *השם ב-*elements[i] *את הערך של* elements[i-1]
3. *השם ב-*elements[0] *את הערך של* temp

יישום ב-#C:

```
temp = elements[elements.length - 1];
for (i = elements.length -1; i >= 1; i--)
{
    elements[i] = elements[i - 1];
}
elements[0] = temp;
```

שימו ♥: לשם פשטות, התבנית מניחה שאיברי הסדרה המיועדים להזזה נמצאים במערך elements, האחד אחרי השני. אבל, ייתכן שהסדרה שבה נדרש לבצע הזזה מעגלית היא תת-סדרה (רצופה או לא רצופה) של איברי המערך. שאלה 10 מתייחסת לבעיה כזאת.

שאלה 9

נתון אלגוריתם, שהקלט שלו הוא 6 מספרים שיישמרו במערך numbers, ומספר שלם חיובי num:

1. *עבור כל* i *שלם בתחום* 1 *עד* 6 *בצע:*
   1.1. *קלוט מספר שלם ב-*numbers[i]
2. *קלוט מספר שלם חיובי ב-*num
3. *בצע* num *פעמים:*
   3.1. *הזזה מעגלית שמאלה במערך* numbers

א. נקלטו הערכים הבאים למערך numbers: 4 6 78 9 34 21

1. מה יהיה הפלט של האלגוריתם עבור הקלט 5 ל-num?
2. מה יהיה הפלט של האלגוריתם עבור הקלט 35 ל-num?

ב. מהו מספר הפעמים שמתבצעת פעולת התבנית: ***הזזה מעגלית שמאלה***?

ג. כתבו אלגוריתם **יעיל** יותר, השקול לאלגוריתם הנתון כך שפעולת התבנית: ***הזזה מעגלית שמאלה*** תתבצע מספר קטן יותר של פעמים. ציינו באיזו תבנית השתמשתם עבור כתיבת האלגוריתם ה**יעיל.**

ד. ישמו את האלגוריתם ה**יעיל** בשפת #C.

שאלה 10

א. כתבו אלגוריתם, שהקלט שלו הוא 16 מספרים למערך, והפלט שלו הוא ערכי המערך לאחר הזזה מעגלית שמאלה עבור סדרת המספרים הנמצאים במקומות האי-זוגיים במערך והזזה מעגלית ימינה עבור סדרת המספרים הנמצאים במקומות הזוגיים במערך.

ב. ישמו את האלגוריתם בשפת #C.

הזזה של תת-סדרה

הזזה של תת-סדרה היא תבנית הנחוצה בהקשרים רבים בעיבוד סדרות. ***הזזה של תת-סדרה*** היא תבנית הזזה ליניארית (לא מעגלית) של תת-סדרה של ערכים אל מקום אחד **שמאלה** או אל מקום אחד **ימינה**. הנה שתי דוגמאות בולטות לשימוש בתבנית: הוצאת ערך ממקום k בסדרה ו"צמצומה לשמאל", כלומר, הזזה במקום אחד שמאלה של כל האיברים מהמקום k+1 וימינה; "ריווח הסדרה ימינה" על ידי הזזה של הערכים החל מהמקום ה-k+1 ימינה כדי לפנות מקום להכנסה של איבר חדש לסדרה במקום ה-k. במקרה של "צמצום לשמאל" מתבצעת פעולת "הוצאה" של ערך מהמקום ה-k **לפני** ההזזה, ובמקרה של "ריווח ימינה" תתבצע פעולת "הכנסה" של ערך חדש למקום ה-k **אחרי** ההזזה. באופן דומה ניתן לבצע "צמצום לימין" ו-"ריווח שמאלה".

נפריד את מאפייני התבנית ***הזזה של תת-סדרה*** לשתי תת-תבניות: ראשית נציג את מאפייני התבנית ***הזזה של תת-סדרה שמאלה*** ואחר כך נציג את מאפייני התבנית ***הזזה של תת-סדרה ימינה***.

שם התבנית: הזזה של תת-סדרה שמאלה

נקודת מוצא: סדרת ערכים במערך elements באורך length, מקום k במערך ($0 \le k < length-1$)

מטרה: הזזה שמאלה של התת-סדרה הנמצאת במקומות k+1..length-1 למקומות k..length-2

אלגוריתם:

1. עבור כל שלם i בתחום מ-k עד length-2 בצע:

1.1 השם ב-elements[i] את הערך של elements[i+1]

יישום ב-C#:

```
for (i = k; i <= elements.length - 2; i++)
{
    elements[i] = elements[i + 1];
}
```

שם התבנית: הזזה של תת-סדרה ימינה

נקודת מוצא: סדרת ערכים במערך elements באורך length, מקום k במערך ($0 < k \le length-1$)

מטרה: הזזה ימינה של התת-סדרה הנמצאת במקומות 0 .. k-1 למקומות 1 .. k

אלגוריתם:

1. עבור כל i שלם בתחום מ-k עד 1 (בסדר יורד) בצע:

    1.1. השם ב-elements[i] את הערך של elements[i-1]

יישום ב-#C:

```
for (i = k; i >= 1; i--)
{
    elements[i] = elements[i - 1];
}
```

שאלה 11

ברשימת החולים המוזמנים לרופא מומחה נקבעה לכל מוזמן פגישה, החל מהשעה 16:00 ועד השעה 20:00, כאשר לכל מוזמן מוקדש פרק זמן של חצי שעה.

מקרה בהול גרם להכנסת חולה חדש לרשימה בשעה 18:30, ולכן יש לעדכן אצל המזכירה הרפואית את רשימת המוזמנים, המסודרת לפי סדר פגישתם המיועדת עם הרופא.

א. כתבו אלגוריתם, שהקלט שלו הוא הרשימה התחילית של מספרי הזהות של 9 החולים המוזמנים וכן את מספר הזהות של החולה החדש, והפלט שלו הוא רשימת מספרי הזהות של החולים, על פי הסדר לאחר העדכון, כאשר לכל חולה מוצגת גם שעת הפגישה המיועדת לו עם הרופא.

ב. ישמו את האלגוריתם בשפת #C.

שאלה 12

ב"מכרז המבטיח" מנהלים רישום של הצעות רכישה ל-5 מערכות ישיבה לסלון. המידע נשמר עבור 20 ההצעות הגבוהות ביותר והן מסודרות לפי סדר יורד.

א. כתבו אלגוריתם, שהקלט שלו הוא 20 הסכומים של ההצעות הגבוהות שהתקבלו עד כה, וכן סכום של הצעה חדשה, והפלט שלו הוא 20 ההצעות הגבוהות ביותר לאחר העדכון של ההצעה החדשה.

ב. ציינו באילו תבניות השתמשתם וכיצד **שילבתם** ביניהן?

ג. הרחיבו את האלגוריתם כך שיציג כפלט את הסכומים של הזוכים ב-5 מערכות הישיבה לסלון. ציינו באיזו תבנית נוספת השתמשתם וכיצד **שילבתם** אותה באלגוריתם.

היפוך סדר האיברים בסדרה

בפרק 3 ראינו את התבנית ***היפוך סדר האיברים בסדרה*** עבור סדרה בת שני איברים. עתה נרחיב את התבנית עבור סדרת איברים באורך כלשהו.

לצורך ***היפוך סדר הערכים בסדרה*** אפשר להחליף בין האיבר הראשון בסדרה לבין האיבר האחרון בסדרה, להחליף בין האיבר השני בסדרה לבין האיבר הלפני אחרון בסדרה וכן הלאה.

נציג את מאפייני התבנית ***היפוך סדר האיברים בסדרה:***

שם התבנית: היפוך סדר האיברים בסדרה

נקודת מוצא: סדרת ערכים במערך elements

מטרה: היפוך סדר האיברים במערך, שאורכו length

אלגוריתם:

*1. השם ב-*limit *את מנת החלוקה של* length *פריטים לשתי קבוצות*

2. עבור כל i *שלם בתחום מ-0 עד* limit-1 *בצע:*

2.1. החלף את הערכים של elements[i] *ושל* elements[length-i-1]

שימו ♥ : ההחלפה מתבצעת עד מחצית אורך סדרת האיברים. אם אורכה אינו זוגי, אז האיבר האמצעי אינו מוחלף עם אף איבר, ונשאר במקומו, כפי שאכן צריך להיות.

שאלה 13

א. כתבו אלגוריתם, שהקלט שלו הוא 15 מספרים למערך, המסודרים בסדר עולה, והפלט שלו הוא ערכי המערך לאחר היפוכם.

ב. שנו את האלגוריתם שכתבתם בסעיף א כך שיהפוך את הסדר של 12 האיברים הראשונים במערך. שלושת הערכים הגדולים ביותר במערך יישארו במקומם. הפלט יהיה ערכי המערך לאחר ההיפוך.

ג. ניתן לבצע את האלגוריתמים שבסעיפים א ו-ב **ללא** שימוש בתבנית ***היפוך סדר האיברים בסדרה***. כתבו את האלגוריתמים המתאימים.