

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 מהמוציא לאור ומהגורמים המפורטים להלן.



כל הזכויות שמורות
משרד החינוך

מסת"ב 965-90844-6-3 ISBN

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, בניית 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 בניית פתרון ופתרון 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 ובניית פתרון 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

1..... 9.1 היכרות ראשונית עם המחלקה string

3..... 9.2 תכונות של עצם ותכונת האורך של מחרוזת

5..... 9.3 ביצוע פעולות על מחרוזות

9..... שרשור מחרוזות

10..... ומה בפנים? פעולות המסתכלות אל תוך הקנקן

12..... 9.4 הוראת השמה במחרוזות

20..... סיכום

21..... רשימת פעולות על מחרוזות

פרק 10 – מערכים..... 25

25..... 10.1 מערך ואיברי מערך

41..... 10.2 חריגה מגבולות מערך

43..... 10.3 קשרים בין מערכים

43..... עיבוד מערכים במקביל ובקצב התקדמות זהה

44..... עיבוד מערכים במקביל בקצב התקדמות לא זהה

46..... 10.4 מערך מונים

50..... 10.5 מערך צוברים

50..... 10.6 יעילות מקום

56..... שאלות נוספות

57..... סיכום

58..... סיכום מרכיבי שפת C# שנלמדו בפרק 10

59..... תבניות – פרק 10

מערך מונים, מערך צוברים, חישוב שכית, הזזה מעגלית בסדרה, הזזה של תת-סדרה,

היפוך סדר הערכים בסדרה

פרק 11 – מחלקות ועצמים: הרחבה והעמקה..... 63

63..... 11.1 מחלקה - הגדרה ושימוש

75..... 11.2 פעולות גישה

85..... 11.3 תכונות מורכבות

110..... סיכום

111..... סיכום מרכיבי שפת C# שנלמדו בפרק 11

פרק 12 – תבניות אלגוריתמיות..... 115

115..... 12.1 מערך דו-ממדי

131..... 12.2 חיפוש בינרי

136..... 12.3 מיונים

136	מיון בחירה
139	מיון הכנסה
146	מיון בועות
147	12.4 מיזוג
150	סיכום
151	סיכום מרכיבי שפת C# שנלמדו בפרק 12
151	שאלות נוספות
155	פרק 13 – פתרון בעיות
181	שאלות נוספות
188	סיכום
189	אינדקס

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, תוו ובויליאני. 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 אחולו. 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 מחרוזות 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 <code>string</code> .
--

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";
```

קצ'ה 1

מטרת הבעיה ופתרונה: הצגת קלט של מחרוזות.

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 מהקלט name-
2. הציג כקלט "Your name is:"
3. הציג כקלט name

יישום האלגוריתם

עד כה ראינו כיצד לקלוט ערכים מטיפוסים פשוטים – שלמים, ממשיים ותווים. קליטת מחרוזות 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.WriteLine("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 מִפְנֵה `name` אל אותו שטח זיכרון שהוקצה על ידי פעולת הקלט.

סוף פתרון קציה 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 **כתכונה** 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.

קציה 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. קיבלת מחרוזת str-2
2. השם את אורכה של המחרוזת len-2 str
3. אם len זוגי
- 3.1 השם letter-2 את התו הנמצא במקום 0 str-2
4. אחרת
- 4.1 השם letter-2 את התו הנמצא במקום 1 len-2 str
5. הצג כפול 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 לערך **התכונה** כותבים כך:

שם התכונה.שם העצם

התכונה Length מוגדרת לעצמים מהמחלקה מחרוזות. התכונה שומרת את אורך המחרוזות. 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 מחרוזות. אחת הפעולות המוגדרות עבור מחרוזות היא הפעולה CompareTo המשווה מחרוזות למחרוזת נוספת. אם s1 הוא עצם

מהמחלקה מחרוזות, ו-s2 הוא עצם נוסף מהמחלקה מחרוזות, אז הביטוי `s1.CompareTo(s2)` מפעיל את הפעולה `CompareTo` המשווה את `s1` למחרוזת נוספת (`s2`) שהתקבלה כפרמטר.

הפעולה `CompareTo` מחזירה את תוצאת השוואה 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 מחרוזות. בהמשך הפרק נכיר פעולות נוספות של המחלקה מחרוזות.

פצ'ה 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. אגור counter 1-2
2. קלט מחרוזת name-2
3. קלט מחרוזת str-2
4. כל עוד $name > str$ (בסדר אלפבטי):
 - 4.1. הגדל את ערכו של counter 1-2
 - 4.2. קלט מחרוזת str-2
5. אם $name = str$
 - 5.1. הצג כפוף "המחפש <name> התקבל"
6. אחרת
 - 6.1. הצג כפוף "המחפש <name> לא התקבל"
7. הצג את ערכו של counter

יישום האלגוריתם

? כיצד נבדוק אם שם המועמד גדול על-פי סדר מילוני מהמחרוזות הנקלטות?

כאמור, הפעולה `CompareTo` בודקת את היחס המילוני שבין שתי מחרוזות. אם שם המועמד `name` גדול מהמחרוזות הנקלטות `str`, הפעולה `name.CompareTo(str)` תחזיר ערך שלם חיובי, ואילו אם שם המועמד קטן מהמחרוזות הנקלטות, פעולה זו תחזיר ערך שלם שלילי. אם המחרוזות שוות הפעולה תחזיר 0.

? כיצד נבדוק אם המחרוזות 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 המחרוזות שעליה היא מופעלת למחרוזות המתקבלת כפרמטר. למשל, הביטוי `name.Equals(str)` יחזיר `true` אם המחרוזות `str` שווה למחרוזת `name`, אחרת יחזיר `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 ממוקד
Console.WriteLine("Enter the candidate name: ");
name = Console.ReadLine();
// לולאת זקיף
Console.WriteLine("Enter first winner name: ");
str = Console.ReadLine();
// כל עוד לא עברנו את שם המועמד
while (name.CompareTo(str) > 0)
{
    counter++;
    Console.WriteLine("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):

(פרמטרים) שם הפעולה. שם העצם

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: apple today good הפלט המתאים הוא: apple good today.

שרשור מחרוזות

פעולה נוספת ושימושית מאוד על מחרוזות היא פעולת השרשור. פעולה זו מקבלת שתי מחרוזות ויוצרת מחרוזת חדשה, המורכבת מהמחרוזות 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 מחרוזות 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: מחרוזת2 + מחרוזת1

פעולת השרשור יוצרת מחרוזת חדשה ומקצה עבורה מקום. היא אינה משפיעה על המחרוזות המקוריות.

כאשר משרשרים למחרוזת 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 מחרוזות מגדירה פעולות שונות המאפשרות להסתכל פנימה לתוך המחרוזות ולהתייחס לתווים המרכיבים אותה. פעולות כאלו מסייעות מאוד בפתרון בעיות שונות, כפי שמדגימה הבעיה הבאה:

תצ'ה 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 e 8. 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 טיפוס 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 בחשפט
Console.WriteLine("There are {0} words", numOfWords);
} // Main
} // class HowManyWords

```

סוף פתרון קציה 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*? יהיה הפלט AABBB3BB??.

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 ויצג 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 המחרוזת. 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.

הצ'יה 5

מטרת הבעיה הבאה: הצגת השמות מחרוזות למחרוזות 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@ii.IL`, שהיא כתובת ישראלית. לעומתן, המחרוזות הבאות אינן כתובות חוקיות: `ח@ח.ע` (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 ("דגלי"), 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. אגור את ערך isLegal ב-true
2. קלוט כגורב str
3. צור מגורב חדשה, זהה ל-str ובה כל האותיות החדולות מוגולפות בקטנות והשם אורה ב-lowerStr
4. אס האור הראשונה היא לא אור אנלית
 - 4.1 שנה את ערך isLegal ל-false
5. אס הגו '.' אור הגו '@' אינן מופיעים במגורב
 - 5.1 שנה את ערך isLegal ל-false
6. אגור
 - 6.1 אס הגו '.' מופיע לפני הגו '@' אור מיד אגורו אור בסוף המגורב
 - 6.1.1 שנה את ערך isLegal ל-false
 7. אס ערכו של isLegal שווה ל-true
 - 7.1 הצג כפלט הודעה כי הכגורב גוקי
 - 7.2 אס שולש הגוריס האגורונס הס המגורב ".il"
 - 7.2.1 הצג הודעה כי הכגורב ישראלית
 - 7.3 אגור
 - 7.3.1 הצג הודעה כי הכגורב אינן ישראלית
8. אגור
 - 8.1 הצג כפלט הודעה כי הכגורב אינן גוקי

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 הזזה 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, הזזה 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.WriteLine("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 תרגיל 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#.

תרגיל 6

מטרת הבעיה ופתרונה: הדגמת בניית מחרוזות 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 החדשה מושמת ב-`newStr`. כלומר במקום ש-`newStr` יפנה 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 ההשמה של מחרוזות אינה שונה מפעולת ההשמה 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 שאותה 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'; // 'c' ב-'b' או 'B' החלפת
            else
                newStr = newStr + str[i]; // המקורי התו
        } // for
        // פלט
        Console.WriteLine("The new string is: {0}", newStr);
    } // Main
} // class ReplaceCForB
```

סוף פתרון קציה 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.סיומת המדינה.שם החברה

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# מחרוזות 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 מחרוזות 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 ב**סימון 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 טיפוס 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 ב**סימון 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 בכתבייתה אין שימוש בסימון הנקודה. סימן פעולת השרשור הוא הסימן `+`.

בפרקים הבאים נלמד להגדיר מחלקות בעצמנו, ולא רק להשתמש במחלקות קיימות.

רשימת פעולות על מחרוזות

דוגמאות		טיפוס הערך המוחזר	תיאור הפעולה	הפעולה
הערך המוחזר	הפעולה			
3	<code>s1.IndexOf("pl")</code> כאשר ב- <code>s1</code> נמצאת המחרוזת "people"	שלם	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.	<code>IndexOf(string s)</code>
2	<code>s1.IndexOf('o')</code> כאשר ב- <code>s1</code> נמצאת המחרוזת "people"			<code>IndexOf(char c)</code>

false	s1.Equals(s2) כאשר ב-s1 נמצאת המחרוזת "love" וב-s2 נמצאת המחרוזת "Love"	בוליאני	פעולה המקבלת מחרוזות, ומחזירה true אם המחרוזות שעליה הופעלה הפעולה והמחרוזות שהתקבלה שוות זו לזו בדיוק. אחרת, היא מחזירה false. פעולה זו זהה לפעולת השוואה ==	Equals (string s)
true	s1.Equals(s2) כאשר ב-s1 נמצאת המחרוזת "love" וב-s2 נמצאת המחרוזת "love"			
מספר שלילי כלשהו	s1.CompareTo(s2) כאשר ב-s1 נמצאת המחרוזת "aa" וב-s2 נמצאת המחרוזת "ab"	שלם	פעולה המקבלת מחרוזות, ומשווה אותה למחרוזת שעליה הופעלה הפעולה. אם הן שוות זו לזו מוחזר הערך אפס. אם המחרוזות שעליה מופעלת הפעולה קודמות למחרוזות שהתקבלה בסדר מילוני, יוחזר מספר שלם שלילי. אם המחרוזות שעליה מופעלת הפעולה, מופיעה אחרי המחרוזות שהתקבלה בסדר מילוני, יוחזר מספר שלם חיובי.	CompareTo (string s)
המחרוזת החדשה "peace"	s1.ToLower() כאשר ב-s1 נמצאת המחרוזת "Peace"	מחרוזת	פעולה שיוצרת מחרוזת זהה למחרוזת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות קטנות. הפעולה מחזירה את המחרוזת החדשה.	ToLower ()
המחרוזת החדשה "PEACE"	s1.ToUpper() כאשר ב-s1 נמצאת המחרוזת "Peace"	מחרוזת	פעולה שיוצרת מחרוזת זהה למחרוזת שעליה היא מופעלת, ובה כל האותיות מוחלפות באותיות גדולות. הפעולה מחזירה את המחרוזת החדשה.	ToUpper ()
המחרוזת החדשה "Bye"	s1.Substring(4) כאשר ב-s1 נמצאת המחרוזת "GoodBye"	מחרוזת	פעולה שיוצרת מחרוזת זהה לתת-מחרוזת המתחילה מהמקום ה-k של המחרוזת שעליה הפעולה מופעלת ועד סופה. הפעולה מחזירה את המחרוזת החדשה.	Substring (int k)

<p>המחרוזת החדשה "Bye Is"</p>	<p><code>s1.Substring(4, 6)</code> כאשר ב-s1 נמצאת המחרוזת "GoodBye Israel"</p>	<p>מחרוזת</p>	<p>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.</p>	<p><code>Substring(int k, int s)</code></p>
---------------------------------------	---	---------------	--	---

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 פתרון 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

הצ'יף 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:

- קליטת נתוני הקלט וצבירתם
- 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?

- קליטת נתוני הקלט, שמירתם וצבירתם
- חלוקת הסכום המצטבר במספר ערכי הקלט
- 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 המציון 0, כך: `scores[0]`. האיבר השני נמצא במיקום 1 מתחילת המערך ולכן פנייה אליו תיעשה באמצעות המציון 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:

<code>scores[0]</code>	<code>scores[1]</code>	<code>scores[19]</code>	<code>scores[39]</code>
	

האלגוריתם

1. אגף אל `sumOfScores` מ-0
2. אגף אל `belowAverageCounter` מ-0
3. עבור כל `i` שלם מ-0 עד מספר הערכים הנקלטים פגום 1 ב-33:
 - 3.1 קלוט אל זמן ההקפה הבא והשם באיבר ה-`i` במערך `scores`
 - 3.2 הוסף לערך השמור ב-`sumOfScores` אל זמן ההקפה הנקלט
4. גוף אל הערך השמור ב-`sumOfScores` במספר ערכי הקלט והשם `averageScore` מ-2
5. עבור כל `i` שלם מ-0 עד מספר הערכים הנקלטים פגום 1 ב-33:
 - 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 כקלט אל 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 תלויה 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.WriteLine("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 4.50 4.10 ... 4.20
נניח שסכום זמני ההקפות הוא 168.8.

מספר 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 parti cipants...

שימו : המעקב אחר הערכים השמורים באיבר של המערך זהה למעקב אחר הערכים השמורים במשתנה.

סוף פתרון בציה 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 במציינ (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:

```
;שם המערך [טיפוס
```

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 [מספר הערכים] טיפוס;
```

כזכור, ניתן לצרף את ההצהרה וההקצאה בהוראה אחת:

```
[מספר הערכים] טיפוס new = שם המערך [טיפוס
```

וניתן גם לבצען בנפרד:

```
;שם המערך [טיפוס
```

```
[מספר הערכים] טיפוס new = שם המערך;
```

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:

	<code>arr[0]</code>	<code>arr[1]</code>	<code>arr[2]</code>	<code>arr[3]</code>	<code>arr[4]</code>
<code>arr</code>					

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];
```

	<code>arr[0]</code>	<code>arr[1]</code>	<code>arr[2]</code>	<code>arr[3]</code>	<code>arr[4]</code>
<code>arr</code>	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: 4.20 4.80 4.40. 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: 1 2 3.
מה יהיה פלט קטע התוכנית עבור הקלט הנתון?

שאלה 10.5

נתונה הצהרת המערך הבאה:

```
int[] temp = new int[10];
```

- כתבו לולאה להשמת הערך 0 בכל אחד מאיברי המערך.
- כתבו לולאה לקליטת עשרה נתוני קלט בעשרת איברי המערך.
- כתבו לולאה להכפלה ב-2 של ערכו של כל איבר במערך.
- 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 המצוין של איבר, כפי שמראה הדוגמה הבאה.

נניח שנתון המערך arr הבא:

	arr[0]	arr[1]	arr[2]
arr	-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.WriteLine("{0} {1}", i, arr[i])	0	-1	2	1	0 -1
3	i = i + 1	1	-1	2	1	
4	Console.WriteLine("{0} {1}", i, arr[i])	1	-1	2	1	1 2
5	arr[i] = arr[i] + 1	1	-1	3	1	
6	Console.WriteLine("{0} {1}", i, arr[i])	1	-1	3	1	1 3

בקטע התוכנית הזה תפקידו של המשתנה i הוא להיות מצוין 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 הוראות ההשמה שבשורה 5 והוראות 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 לוועזיות
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.

קצ'ה 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 הגדולים מהממוצע. (שלא כמו בבעיה 1, שם היה עלינו למנות את מספר הנתונים הקטנים 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.WriteLine("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.

בואנו לכתוב הוראה לביצוע-חוזר המבצעת סריקה ועיבוד של איברי מערך, נתאים את סוג ההוראה להגדרת הסריקה שצריכה להתבצע.

אם סיום הסריקה תלוי בקיום תנאי, נשתמש בביצוע-חוזר-בתנאי (המיושמת בלולאת **while**).

פירוק הבעיה לתת-משימות

הנה חלוקה ראשונית לתת-משימות עבור פתרון הבעיה:

- קליטה ושמירה של מצב הלוח ושל מקום השחקן על הלוח
- 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 פלט:**

1.1 קאוט אם מצב המשבצת ה-i של הלוח

2. קאוט אם מיקום השחקן

3. **עבור כל i שלם מ-1 עד 6:**

3.1 אם המשבצת באינדקס i ממיקום השחקן היא חיה

3.1.1 הצג אם 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 במציון 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 $pawn + i$ כמציון. הביטוי $pawn + i$ 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: 5 7.

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 שווים 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" 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 בניית 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++)
    compare = (m1[i] == m2[i]);

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++)
    if (m1[i] == m2[i])
        compare = true;

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:

- הביאו דוגמה לערכים של איברים בשני המערכים שקטע התוכנית משיג את מטרתו עבורם.
- הביאו דוגמה לערכים של איברים בשני המערכים שקטע התוכנית אינו משיג את מטרתו עבורם, והסבירו מדוע זה קורה.
- תקנו את קטע התוכנית כך שישגי את מטרתו.

שאלה 10.31

כיתה י' 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

קצ'ה 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 5 1 -1 3 4 -2.

המערך numbers ישמור בתוכו 5 מספרים שלמים. בעקבות ההעתיקה 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: המציינן של המערך numbers יסרוק את איברי המערך בזה אחר זה. המציינן של המערך positive והמציינן של המערך negative יצביעו בכל רגע על המקום הפנוי הבא במערך המתאים להם, כדי שהעתקת האיבר הבא תתבצע למקום הנכון.

המציינן 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. קאוט ערך גיוכי שלם $size-2$
2. עכור כל i שלם $0 \leq i < size-1$ כצב:
 - 2.1. קאוט נאון $numbers[i]$
 3. אגא א $indexPos$ $0-2$
 4. אגא א $indexNeg$ $0-2$
 5. עכור כל $indexNum$ שלם $0 \leq indexNum < size-1$ כצב:
 - 5.1. אס $numbers[indexNum] > 0$
 - 5.1.1. העגק אג ערכו של $numbers[indexNum]$ $positive[indexPos]$
 - 5.1.2. העצא אג ערכו של $indexPos$ $1-2$
 - 5.2. אגא
 - 5.2.1. העגק אג ערכו של $numbers[indexNum]$ $negative[indexNeg]$
 - 5.2.2. העצא אג ערכו של $indexNeg$ $1-2$

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.

הצ'יה 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 מנייה 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 ייצג 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 הממוצעים ב-numOfSingers
2. קלוט מספר מועמד ב-vote
3. כל עוד $vote \neq -1$:
 - 3.1 העלה ב-1 את המונה (במערך) של הממוצע שמספרו vote
 - 3.2 קלוט מספר מועמד ב-vote
 4. אגף את min בערך המונה של הממוצע הראשון
 5. אגף את minSinger ב-1
 6. עד כן i שלם מ-2 עד numOfSingers:
 - 6.1 אם ערך המונה של הממוצע מספר i קטן מ-min:
 - 6.1.1 השם ב-min את ערך המונה של הממוצע שמספרו i
 - 6.1.2 השם את i ב-minSinger
 7. העז 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" והצבות 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.WriteLine("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.WriteLine("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.WriteLine("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 בציה 5

להעמקה בתבנית מערך מונים פנו לסעיף התבניות המופיע בסוף הפרק.

שאלה 10.35

עקבו אחר התוכנית IsraeliIdol בעזרת טבלת מעקב, עבור הקלט 1- 2 4 4 2 1 1 3 2 4 (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 הווידאו 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 שכיה 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 בפתרון נחוץ מערך ובין בעיות אשר ניתן לפתור גם ללא מערך, ונבין מדוע לא כדאי להשתמש במערך אם אין בו צורך.

קצ'ה 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 הצפייה, 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. קאונט את מספר התוכניות numOfPrograms -2
2. אגור את ערכו של above20Counter ל-0
3. עבור כל i שלם מ-0 עד numOfPrograms -232:
 - 3.1 קאונט את אורך הצפייה הכולל rate -2
 - 3.2 אם $20 < \text{rate}$
 - 3.2.1 הגדל את ערכו של above20Counter ב-1
 - 3.3 הגדל את ערכו של above20Counter ב-1

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: 5 41 30 29 15 3

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.WriteLine("Enter number of programs: ");
        numOfPrograms = int.Parse(Console.ReadLine());
        rating = new double[numOfPrograms];
```

```

for (i = 0; i < numOfPrograms; i++)
{
    Console.WriteLine("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 והיפוך 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:

- מה מספר השחקן או השחקנים שצברו את מספר הנקודות הגדול ביותר?
- כמה נקודות צברה הקבוצה במשך כל המשחק?
- מה סוג הקליעה הנפוץ ביותר (קליעות שזיכו בנקודה אחת, ב-2 נקודות או ב-3 נקודות)?
- איזה שחקן או שחקנים לא צברו שום נקודה במשך כל המשחק?

לדוגמה: הקלט (משמאל לימין) 0 3 2 5 1 3 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 להמשיך החיפוש. כמו כן יותם אומר לחניכיו את מספר החדר הראשון שיש להתחיל ממנו את החיפוש. 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 שתחיל 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 שפתרון 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:

שם המערך [] טיפוס;

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:

[מספר הערכים] טיפוס **new**;

ניתן לצרף את ההצהרה ואת ההקצאה להוראה אחת:

[מספר הערכים] טיפוס **new** = שם המערך [] טיפוס;

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) יאוּתחלו ב-0, איברי מערך בוליאני יאוּתחלו ב-**false**, ואיברי מערך תווי יאוּתחלו 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 שבפתרון 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. **אגף** `אגף איברי המערך` `countElements` `0-2`

2. **השם** `אגף המערך` `הכא` `2-3` `element`

3. **כא** `2-3` `לא` `אגפים` `האגף` `toEnd` `2-3`

3.1 **הכא** `אגף` `countElements[whichCounter]` `1-2`

3.2 **השם** `אגף המערך` `הכא` `2-3` `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. אגף את ערכי המערך הצוברים

2. כצע limit פסמים:

2.1. השם את הערך הכא בסדרה element-

2.2. השם את הערך לצבייה 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 limit אלגוריתם:

1. אגף את ערכי המערך המונים

2. כצע limit פסמים:

2.1. השם את הערך הכא בסדרה element-

2.2. הגדל את countElements[whichCount] ב-1

3. אגף את max בערכו של countElements[0]

4. עבור כל i שלם בהגומס מ-1 עד אורך המערך countElements פגומ ו כצע:

4.1. אם $\text{countElements}[i] > \text{max}$

4.1.1. השם ב-max את הערך של $\text{countElements}[i]$

5. עבור כל i שלם בהגומס מ-0 עד אורך המערך countElements פגומ ו כצע:

5.1. אסערכו של $\text{countElements}[i]$ 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 :

- השם temp-2 אג elements[0]
- עבור כל i שלם בגומס 0 עד length-2 כזש :
 - השם temp-2 elements[i] אג הערך של elements[i+1]
 - השם temp-2 elements[length-1] אג 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 :

- השם temp-2 אג elements[length-1]
- עבור כל i שלם בגומס 1 עד length-1 (כסדר יורד) כזש :
 - השם temp-2 elements[i] אג הערך של elements[i-1]
 - השם temp-2 elements[0] אג 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 ($0 \leq k < \text{length}-1$)
 מטרה : הזזה שמאלה של התת-סדרה הנמצאת במקומות k+1..length-1 למקומות k..length-2 אלגוריתם :

- עבור כל i שלם בגומס k-M עד length-2 כזש :
 - השם temp-2 elements[i] אג הערך של elements[i+1]

שם התבנית : הזזה של תת-סדרה ימינה
 נקודת מוצא : סדרת ערכים במערך elements באורך length, מקום k במערך ($0 < k \leq \text{length}-1$)
 מטרה : הזזה ימינה של התת-סדרה הנמצאת במקומות 0..k-1 למקומות k..length-1 אלגוריתם :

- עבור כל i שלם בגומס k-M עד length-2 כזש :
 - השם temp-2 elements[i] אג 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 limit-2 אגמנת החלוקה של length פריטים לשתי קבוצות

2. עבור כל i שלם בגומא מ-0 עד limit-1:

2.1. החלף את הערכים של elements[i] ושל elements[length-i-1]

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 **מעריך מונים**. התבנית של **מעריך מונים**, כפי שמשתמשת 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. כצט 58 צעמים:

1.1 קאול ספריה 2-digit

1.2 הציל אל `countDigitsArr[digit]` 1-2

1. קאול אל מספר המאמציים 2-numOfSingers

2. קאול מספר מושמצי 2-vote

3. כל עוצ 1-vote \neq כצט:

3.1 העלה 1-2 אל המולה של המאמצי מספר vote

3.2 קאול מספר מושמצי 2-vote

בשני קטעי האלגוריתמים האלה ניתן לזהות תבנית **מנייה**, אלא שהיא מופעלת על כמה מונים הפועלים במקביל, ושמורים במערך אחד.

נציג את מאפייני התבנית **מעריך 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. אגף את איברי המערך countElements 0-2

2. השם את הערך הבא בסדרה element-2

3. כן עד לא מקיים הנאי toEnd כ3ע:

3.1 הגדל את countElements[whichCounter] ב-1

3.2 השם את הערך הבא בסדרה element-2

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.**
 - 1 - element (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:

<code>arr[0]</code>	<code>arr[1]</code>	<code>arr[2]</code>	<code>arr[3]</code>	<code>arr[4]</code>	<code>arr[5]</code>	<code>arr[6]</code>	<code>arr[7]</code>
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` למציין 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. כִּצְע 247 עֲצָמִים:

1.1. קְלוּט מִסְפְּרֵי כִּיָּה classNum-2 כִּסְפֵּי שֶׁהוֹווּ גִּמְמִיז 2-
sumStudent

1.2. הַצֵּף אֶל sumClasses[classNum-1] 2-sumStudent

2. אגף sumTotal - 0

3. עבור כל i שלם בגומ 0 עד 6323 :

3.1. הצג כפוף: הסכום המצטבר לכיתה $i+1$ הוא: $\text{sumClasses}[i]$

3.2. הצג sumTotal - $\text{sumClasses}[i]$

4. הצג כפוף: הסכום המצטבר לשכבה הוא sumTotal

1. קוף מספר מלקה - departmentNum וסכום הקנה של לקו - price

2. כל departmentNum שונה מ- 0 עד 323 :

2.1. הצג $\text{sumDepartments}[\text{departmentNum}-1]$ - price

2.2. קוף מספר מלקה - departmentNum וסכום הקנה של לקו - price

3. עבור כל i שלם בגומ 0 עד 423 :

3.1. הצג כפוף: הפדיון היומי למלקה $i+1$ הוא: $\text{sumDepartments}[i]$

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. אגור את ערכי מערך הצוברים

2. בצע limit פעמים:

2.1. השם את הערך הבא בסדרה element-

2.2. השם את הערך לצבירה value-

2.3. הוסף- sumElements[whichSum] את ערכו 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 תלוי 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. אגף את איברי המערך sumClasses ל-0

2. עבור כל i שלם בגוון 0 עד 247 עשה:

2.1 קאוט מספר כינה ב-classNum וסכום כסף שהוויא גלמיז ב-

sumStudent

2.2 הגדל את sumClasses[classNum-1] ב-sumStudent

3. אגף את sumTotal ל-0

4. עבור כל i שלם בגוון 0 עד 6 עשה:

4.1 הגדל כפוט: הסכום המצטבר לכיתה i+1 ה/א: sumClasses[i]

4.2 הגדל את sumTotal ב-sumClasses[i]

5. הגדל כפוט: הסכום המצטבר לשכבה ה/א sumTotal

6. אגף את max ל-sumClasses[0]

7. עבור כל i שלם בגוון 0 עד 6 עשה:

7.1 אם $\text{sumClasses}[i] > \text{max}$

7.1.1 השם ב-max את הערך של sumClasses[i]

8. עבור כל i שלם בגוון 0 עד 6 עשה:

8.1 אם ערכו של sumClasses[i] שווה ל-max

8.1.1 הגדל כפוט: כינה מספר i+1 זוכה ביום טיול אגב

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

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 מציני 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 במערך **מונים**.

הנה האלגוריתם המלא :

1. **כצט 58 פעמים:**

1.1. **קלט ספרה 2-digit**

1.2. **הציא אר [digit] countDigitsArr 1-2**

2. **השם 2-max אר [0] countDigitsArr**

3. **עבור כל i שלם בגומ 0 עד 9 צצט:**

3.1. **אר [i] countDigitsArr צצא מ-max**

3.1.1. **השם 2-max אר הערך של [i] countDigitsArr**

4. **עבור כל i שלם בגומ 0 עד 9 צצט:**

4.1. **אר [i] countDigitsArr צצא מ-max**

4.1.1. **הציא 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. אגורא את ערכי המערך המולנס

2. בצע limit פעמים:

2.1. השם את הערך הבא בסדרה element-

2.2. הגדל את sumElements[whichCount] ב-1

3. אגורא את max בערכו של countElements[0]

4. עבור כל i שלם בגווס מ-1 עד אורך המערך countElements פגור ו בצע:

4.1. אם countElements[i] > max

4.1.1. השם ב-max את הערך של countElements[i]

5. עבור כל i שלם בגווס מ-1 עד אורך המערך countElements פגור ו בצע:

5.1. אסערכו של countElements[i] שווה ל-max

5.1.1. הצג את i כפלט

יישום ב-C#:

```
for (i = 1; i <= limit; i++)
{
    element = 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 temp-2 אג elements[0]

2. עבור i כל השם בגומ 0 עד length-2 בצע:

2.1. השם elements[i]-2 אג הערך elements[i+1]

3. השם elements[length-1]-2 אג הערך 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 temp אג elements[length-1]

2. עבור כל i אגם בגומס length-1 עד 1 (כסדר יורד) בצע:

2.1 השם elements[i] אג הערך א elements[i-1]

3. השם elements[0] אג הערך א 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 אגם בגומס 1 עד 6 בצע:

1.1 קאוט מספר אג elements[i]

2. קאוט מספר אג ג'ובי num

3. בצע num פעמים:

3.1 הזזה מעגלית שמאלה במערך numbers

א. נקלטו הערכים הבאים למערך numbers: 21 34 9 78 6 4

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 השתמשותם עבור כתיבת האלגוריתם **היעיל**.
- ד. ישמו את האלגוריתם **היעיל** 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 $k+1$ ימינה כדי לפנות מקום להכנסה של איבר חדש לסדרה במקום k . במקרה של "צמצום לשמאל" מתבצעת פעולת "הוצאה" של ערך מהמקום k לפני ההזזה, ובמקרה של "ריווח ימינה" תתבצע פעולת "הכנסה" של ערך חדש למקום 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 ($0 \leq k < \text{length}-1$)

מטרה: הזזה שמאלה של התת-סדרה הנמצאת במקומות `k+1..length-1` למקומות `k..length-2`

אלגוריתם:

1. עבור i שלם בגוון k עד $\text{length}-2$ בצורה:

1.1 השם ב-`elements[i]` אל 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 ($0 < k \leq \text{length}-1$)

מטרה: הזזה ימינה של התת-סדרה הנמצאת במקומות $k-1 \dots 0$ למקומות $k \dots 1$

אלגוריתם:

1. עבור כל i אלמנט באורך $k-1$ (בסדר יורד):

1.1. העם ב- $\text{elements}[i]$ אל הערך ב- $\text{elements}[i+1]$

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 הראינו 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 limit-2 מגמת החלוקה של length פריטים לשתי קבוצות

2. עבור כל i שלם בגומל מ-0 עד limit-1 כ-3:

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.

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

קצ'ה 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, שתיים 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 10:45 או 12:12. 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 שרשרת 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 לתכונות השעות ולתכונות 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 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 בכוורת הפעולה (במקרה זה `string`). במקרה של פעולות שלא מחזירות ערך כמו הפעולה `SetTime` אין צורך במשפט `return`, והחזרה 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.WriteLine("Enter the hour (0..23): ");  
        h = int.Parse(Console.ReadLine());  
        Console.WriteLine("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 :

```
(רשימת פרמטרים) שם-הפעולה טיפוס-הערך-המוחזר public
{
    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 מופְּרָים 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.

מצי' 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 אורכייהם ורוחבייהם 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. כצעד 20 פסחים:

- 1.1 קלוט את אורך המלבן length
- 1.2 קלוט את רוחב המלבן width
- 1.3 צור עצם מסוג Rectangle בשם rec ואגור 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

הצ'י 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 ועדכון
 - 2.2 קליטת אורך אוזניו של ארני ועדכון, חזרה על כך כל עוד העדכון לא בוצע (פעולת העדכון החזירה **false**)
 - 2.3 קליטת משקלו של ברני ועדכון
 - 2.4 קליטת אורך אוזניו של ברני ועדכון, 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

rabbit1 – עצם מטיפוס המחלקה ארנבון, מייצג את ארני.
rabbit2 – 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.WriteLine("Enter Arni's weight: ");
            currentWeight = int.Parse(Console.ReadLine());
            rabbit1.SetWeight(currentWeight);
            Console.WriteLine("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.WriteLine("Enter Arni's ears length: ");
                currentEarsLength = int.Parse(Console.ReadLine());
            }
        }
    }
}
```

```

    {
        Console.WriteLine("ReEnter Arni's ears length:");
        currentEarsLength = int.Parse(Console.ReadLine());
    }
    Console.WriteLine("Enter Barni's weight: ");
    currentWeight = int.Parse(Console.ReadLine());
    rabbit2.SetWeight(currentWeight);
    Console.WriteLine("Enter Barni's ears length:");
    currentEarsLength = int.Parse(Console.ReadLine());
    // עוד העדכון לא בוצע, נבקש להכניס שוב את האורך
    while (!rabbit2.SetEarsLength(currentEarsLength))
    {
        Console.WriteLine("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 יעצם 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 10 4 (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 שיוספר 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.

תשובה 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 "שחקן" 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 ערך שחקן 1 לא ניצח ואם שחקן 2 לא ניצח בצע:
 - 1.1 אם לשחקן 2 יצא דאבל
 - 1.1.1 שחקן 1 – שחקן כפול
 - 1.2 אגרג
 - 1.2.1 שחקן 1 - שחקן
 - 1.3 אם לשחקן 1 יצא דאבל
 - 1.3.1 שחקן 2 – שחקן כפול
 - 1.4 אגרג
 - 1.4.1 שחקן 2 – שחקן
 2. אם שחקן 1 ניצח ואם שחקן 2 ניצח
 - 2.1 הצג כפול "גיקו"
 3. אגרג אם שחקן 1 ניצח
 - 3.1 הצג כפול "שחקן 1 ניצח"
 4. אגרג
 - 4.1 הצג כפול "שחקן 2 ניצח"

מימוש

```
/*
המחלקה הראשית הממשת 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 קציה 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 צייפס, 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 מאוזנות 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

קציה 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 ינחש 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 תגדיר 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 שנגדיר יהיו שונים זה מזה. לצורך כך, נעזר במערך בוליאני שישמור עבור כל מספר מ-1 עד 100 אם כבר הגדירו 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 שמגדילים 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 המחזווה 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 המחזווה 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 ערך מטיפוס "סופס ניחושים" ואגרו את הפרס
2. כל עוד לא נגשו כל המספרים וגם מונה הניחושים קטן מ-10 בצע:
 - 2.1 הגדל את מונה הניחושים ב-1
 - 2.2 קוט מהמשתף את הניחוש והשם ב-guess
 - 2.3 אם הניחוש מופיע בטופס
 - 2.3.1 הצג כפאט "ניחוש נכון"
 - 2.4 אגרו
 - 2.4.1 הצג כפאט "ניחוש שגוי"
 3. אם כל המספרים נגשו
 - 3.1 הצג כפאט: ניצחג: זכית ב>הפרס<
 4. אגרו
 - 4.1 הצג כפאט: הפסדג: לא זכית ב>הפרס<

מימוש

```
/*  
    המחלקה הראשית המממשת 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.WriteLine("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.

חזרה 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 שגובה קפיצתם זהה לזה שלו. נרצה להשוות כל זוג קופצים פעם אחת בלבד. אחרי שהשוונו 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 n ב-2:
 1.1 עבור כל j מ- $i+1$ עד מספר הקופצים n ב-2:
 1.1.1 אם גובה הקפיצה של הקופץ ה- i במעקף שווה לגובה הקפיצה של הקופץ ה- j במעקף
 1.1.1.1 הצג את שם הקופץ ה- i והקופץ ה- j במעקף ואם 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 תגדיר 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.

קצ'ה 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 (רשימת פרמטרים) שם-הפעולה טיפוס-הערך-המוחזר
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 תציה 7

שימו: הפעולות במחלקת המחשבון מוגדרות כפעולות סטטיות, לכן השתמשנו בהן **ללא** יצירת עצם מסוג מחשבון. זימון פעולות סטטיות נעשה ישירות דרך שם המחלקה:

(פרמטרים) שם הפעולה. שם המחלקה

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. 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:

- 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.
- האות השכיחה ביותר – פעולה המקבלת מחרוזת ומחזירה את האות השכיחה ביותר במחרוזת. אם יש כמה כאלה תוחזר הראשונה מביניהן.

קצ'ה 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

```

/*
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 טיפוסים 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.

קצ'ה 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);
    }
    //חזר 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);
//2 נוכל להעביר 100 שקלים מחשבון 1 לחשבון
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 "הדיגר" 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 _____

קצ"ה 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 בחודש, 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;
    //פעולה 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 מוכר מפני שינויים מבחוץ. כל שינוי שיתבצע על המערך המוחזר לא ישפיע על נתוני המכירות בעצם מוכר.

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 תציה 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 **כפרטיות (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 **כציבוריות (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:

שם-התכונה טיפוס-התכונה הרשאת-הגישה

```
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:

(רשימת פרמטרים) שם-הפעולה טיפוס-הערך-המוחזר הרשאת-הגישה

```
{
    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 `return` 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:

(רשימת פרמטרים) שם-המחלקה הרשאת-הגישה

```
{
}
```

- ◆ 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:

(פרמטרים) שם הפעולה.שם המחלקה

פרק 12 – תבניות אלגוריתמיות

12.1 מערך דו-ממדי

מערך דו-ממדי מייצג מבנה מתמטי הנקרא מטריצה, ובוודאי מוכר לכם מתשביצים, 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 ב**שני** 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. i שלם בין 1 ל-N $0 \leq i < N$
1.1 j שלם בין 1 ל-M $0 \leq j < M$
1.1.1 קלוט ערך והשם אל הערך במערך בשורה ה- i /מסלול 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` מציינ 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 הדין 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 עמודה, מספר שורה]`
- ♦ 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.

קצ'ה 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), ויצג 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 :

מבחן מספר 1
מבחן מספר 7

	↓						↓
1 תלמיד מספר →	85	70					98
2 תלמיד מספר →	62	93					

25 תלמיד מספר →							

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 פגום 1 כ32:
2.1 הוסף לסכום הציונים את הערך הנמצא במערך הציונים
בשורה studentNumber-1 ובמחלקה i
3. גשב את ממוצע התלמיד כסכום הציונים גלקי 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 פגום 1 כ32:
2.1 הוסף לסכום הציונים את הערך הנמצא במערך הציונים בשורה i
ובמחלקה testNumber-1
3. גשב את ממוצע המבחן כסכום הציונים גלקי 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 ועדכוןם.
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:
- פעולה בונה המקבלת את ממדי המערך הדו-ממדי ומקצה עבורו מקום.
 - פעולת גישה לאתחול תא במערך הדו-ממדי. הפעולה מקבלת מספר שורה, מספר עמודה ומספר שלם ומעדכנת את התא המתאים.
 - פעולה המקבלת מספר שורה ומחזירה את סכום איברי השורה.
 - פעולה המציגה את ערכי האיברים הנמצאים בשורות **הזוגיות**.
 - 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:

0	1	2	3
10	11	12	13
20	21	22	23

- 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:
- פעולה בונה המקבלת את ממדי המערך הדו-ממדי ומקצה עבורו מקום.
 - 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:

- פעולה בונה המקבלת את ממדי המטוס (מספר שורות ומספר מושבים בכל שורה). הפעולה תקצה את מערך המושבים ותאתחל את כל המושבים כך שיהיו פנויים.
 - 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: 'a'-ch. חשבו מדוע? (ראו פרק 4).

- 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:

- למי מהשניים כמות הניצחונות הגדולה ביותר.
- 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.
- 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 משני:

		X
	X	
X		

אלכסון ראשי:

X		
	X	
		X

חשבו: מה מאפיין את כל התאים באלכסון הראשי? ובאלכסון המשני? לצורך פתרון הבעיה הבאה נזדקק לאפיון זה.

קצ'ה 2

מטרת הבעיה ופתרונה: הצגת שימוש באלכסונים במטריצה ריבועית.

פתחו אלגוריתם שיקלוט ערכים ממשיים למטריצה ריבועית בגודל $n \times n$. האלגוריתם יחשב ויצג 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 המתחלת 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]`, חֲבְרו את האינדקסים, מה קיבלתם?

במטריצה ריבועית בגודל $n \times n$, המציינים של כל תא באיברי האלכסון הראשי שווים.
 במטריצה ריבועית בגודל $n \times n$, סכום המציינים של כל תא באיברי האלכסון המשני שווה ל-1-n.

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;
    }
}
}

```

מימוש הפעולה הראשית

```

/*
    המחלקה הראשית המשתמשת במחלקת 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 5x5 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[i,2]);
 }
 ב. `for(i = 0; i < mat.GetLength(0); i++)`
 {
 Console.Write(m[2,i]);
 }
 ג. `for(i = 0; i < mat.GetLength(0); i++)`
 {
 Console.Write(m[i,i]);
 }
 ד. `for(i = 0; i < mat.GetLength(0); i++)`
 {
 Console.Write(m[5,5]);
 }

שאלה 12.7

פתחו וישמו אלגוריתם הבודק לגבי מטריצה ריבועית אם היא:

- "שוות שורות" – מטריצה "שוות שורות" היא מטריצה שסכום כל שורה בה שווה.
- "שוות עמודות" – מטריצה "שוות עמודות" היא מטריצה שסכום כל עמודה בה שווה.
- "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 המציניים את מיקום המטבע, תפעיל את פעולת מהלך המשחק, תסכם את הניקוד ולבסוף תציג את התוצאה.

הצ'יה 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 ויצג 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 אינה סדרה גשכולית נחזיר "שקר"*

2. *נחזיר "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;  
    }  
}
```

```
} // class Serial
```

שימו ♥ :

2. בדיקה אם המטריצה "סדרתית" באמצעות פעולה מתאימה.

סוף פתרון בעיה 3

שאלה 12.9

שתטפל בפעולות הקולנוע:

 $\{$

```

        _____;
    }
    // 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 כיסא פנוי, מספר שורה] [מספר כיסא פנוי, מספר שורה]
    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 מְחִיָּה: 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 מציניים 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 במערך 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.	כאשר יש איברים במערך	23
1.1	אם key שווה לאיבר האמצעי אז key במערך	
1.2	אחרת אם האיבר האמצעי גדול מ- key אז	
1.2.1	המשך את החיפוש במחצית הגדולה של המערך	
1.3	אחרת	
1.3.1	המשך את החיפוש במחצית הקטנה 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 key (50) 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 שיצינו 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.

מהו התנאי לעצירת החיפוש? 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.

כיצד נדע שאי אפשר להקטין עוד את קטע המערך? 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 ממיון 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 פעמים בחצי עד שמגיעים ל-1 (8, 4, 2 ו-1).

קצ'ה 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

```
/*
    המחלקה DrivingTest
*/
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;
            }
        }
        return false;
    }
}
//class DrivingTest
```

סוף פתרון מציה 4

שאלה 12.13

לפניכם קטע קוד שמבצע חיפוש בינרי במערך arr :
הניחו כי המשתנים low ו-high מאותחלים בגבולות המערך 0 ו-N.

```
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 **המציניים** 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": 15, 20, 29, 40, 66, 97. 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 ממיון 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 n אורך הסדרה
השם במקום ה-i את הערך הקטן ביותר מבין הערכים שבהם לא טיפאנו

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.

ענה אפשר להשלים את האלגוריתם למיון המערך A :

עבור i מ-0 עד $A.Length-2$ כ-2:

מצא את הערך הקטן ביותר בקטע $A[i]..A[A.Length-1]$

ושמור את המצויין $iMin$ כ-2

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.

תשובה 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 תמייין 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

```
/*
    המחלקה Library
*/
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 המצוין 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 $i = N-1$ הלולאה הפנימית מתבצעת $N-(N-1) = 1$ פעמים

לפי הנוסחה לסכום סדרה חשבונית נקבל שמספר הפעמים שהלולאה מתבצעת הוא:

$$1 + 2 + \dots + (N-1) = \frac{(N-1) \cdot N}{2}$$

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 להוזז? האיברים שצריכים להוזז 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 i את מספר האיברים הממוינים במערך פגום את
2. כל עוד האיבר במקום ה- i גדול מ- element i צמצם
 - 2.1. הצב את האיבר במקום i ימין
 - 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 $i \geq 0$ האיבר במקום ה- i גדול מ- element i צמצם

תשובה 6

מטרת הבעיה ופתרונה: הצגת מחלקה ובה הכנסת איבר למערך ממוין.

המורה להיסטוריה צופיה, החליטה לשמור את רשימת תלמידיה בצורה ממוינת לפי ציונים 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

```
/*  
    המחלקה Student  
*/  
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.

```

/*
    המחלקה Teacher
*/
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--;
        }
        students[i+1] = new Student(stud.GetName(),
                                    stud.GetHistoryGrade());
        numOfStudents++;
    }
}
} // class Teacher

```

סוף פתרון תרגיל 6

שאלה 12.28

- א. כמה דוגמאות מייצגות שונות של המשתנים `numOfStudents`, `stud` ו-`students` בדאי 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-חזור 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:

<code>A[0]</code>	<code>A[1]</code>	<code>A[2]</code>	<code>A[3]</code>	<code>A[4]</code>	<code>A[5]</code>	<code>A[6]</code>
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.

<code>A[0]</code>	<code>A[1]</code>	<code>A[2]</code>	<code>A[3]</code>	<code>A[4]</code>	<code>A[5]</code>	<code>A[6]</code>
35	40	15	38	42	17	39

<code>A[0]</code>	<code>A[1]</code>	<code>A[2]</code>	<code>A[3]</code>	<code>A[4]</code>	<code>A[5]</code>	<code>A[6]</code>
15	35	40	38	42	17	39

<code>A[0]</code>	<code>A[1]</code>	<code>A[2]</code>	<code>A[3]</code>	<code>A[4]</code>	<code>A[5]</code>	<code>A[6]</code>
15	35	38	40	42	17	39

<code>A[0]</code>	<code>A[1]</code>	<code>A[2]</code>	<code>A[3]</code>	<code>A[4]</code>	<code>A[5]</code>	<code>A[6]</code>
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.

קצ'ה 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 ממיון".

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 ממיון ביחס לעצמו בלבד).
- ◆ **`InsertElement`** – פעולה פרטית המקבלת איבר להכנסה, ומכניסה את האיבר בחלק הממוין של המערך, כך שהמערך יישאר ממיון. 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

```
/*
    המחלקה SortedArray
*/
public class SortedArray
{
    private int[] numbers;
    private int sorted;
    // פעולה בונה
    public SortedArray(int[] a)
    {
        numbers = a;
        sorted = 1;
    }
    // הכנס למערך ממיון, 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 מציה 7

שאלה 12.30

תארו מה קורה במיון הכנסה כאשר מנסים למיין מערך ממוין.

שאלה 12.31

אפינו 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 ממוין 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 במיין הכנסה, ללא הכנסת מספרים חוזרים. הוסיפו פעולה המחזירה את המערך הממוין.

מיין בועות

מיין בועות הוא מיין 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:

for (int i = 0; i < A.Length - 1; i++)

for (int j = 0; j < A.Length - 1 - i; j++)

if (A[j] > A[j+1])

swap(A[j], A[j+1])

מיין בועות נקרא כך כיוון שבכל שלב "מבעבעים" את האיבר הגדול למקומו.

שאלה 12.35

לפניכם קטע תוכנית הממיין בשיטת "מיין בועות" את המערך A ובו מספרים שלמים. השלימו את החלקים החסרים במיין: (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[_____] = temp;
        }
    }
}
```

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 המצוין iA ואת המערך B באמצעות המצוין 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.

חשוב: 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:	A[0]	A[1]	A[2]	A[3]					
	F	G	K	R					
	iA								
B:	B[0]	B[1]	B[2]	B[3]	B[4]				
	J	L	M	T	X				
	iB								
C:	C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
	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					
	B[0]	B[1]	B[2]	B[3]	B[4]				
	J	L	M	T	X				
	C[0]	C[1]	C[2]	C[3]	C[4]	C[5]	C[6]	C[7]	C[8]
	F								

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 ערוץ נשארו איברים ב-A וגם ב-B בצע
1.1 אם האיבר ב-A קטן מהאיבר ב-B אז
1.1.1 העסק את האיבר מ-A ל-C
1.1.2 הגדל את המצוין של A ב-1
1.2 אחר
1.2.1 העסק את האיבר מ-B ל-C
1.2.2 הגדל את המצוין של B ב-1
1.3 הגדל את המצוין של C ב-1
2. כל ערוץ נשארו איברים ב-A בצע
2.1 העסק את האיבר מ-A ל-C
2.2 הגדל את המצוין של A ב-1
2.3 הגדל את המצוין של C ב-1
3. כל ערוץ נשארו איברים ב-B בצע
3.1 העסק את האיבר מ-B ל-C
3.2 הגדל את המצוין של B ב-1
3.3 הגדל את המצוין של C ב-1

3.1 העתק את האיבר מ-B ל-C

3.2 העדל את המצוין של B ב-1

3.3 העדל את המצוין 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-sortedB. 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 הממוזג את שני המערכים כך שהמערך הממוזג יהיה ממוין בסדר עולה.

סיכום

בסעיף 12.1 הוצגו בעיות שפתרון 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 מצוין את מספר השורה והשני מצוין 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 **למיון 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 הממויג את האיבר הקטן ביותר מבין האיברים שבמערך הקלט. בסוף התהליך יכול להישאר "זנב" באחד ממערכי הקלט, יש להעתיק אותו למערך הממויג.

סיכום מרכיבי שפת C# שנלמדו בפרק 12

- ♦ **הצהרה על מערך דו-ממדי בשפת C#** נכתבת כך:


```
שם המערך [ , ] טיפוס
```

 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**:


```
[מספר עמודות, מספר שורות] טיפוס new = שם המערך
```

 ניתן לשלב את ההצהרה ואת ההקצאה בהוראה אחת:


```
[מספר עמודות, מספר שורות] טיפוס new = שם המערך [ , ] טיפוס
```
- ♦ **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 0 (ניתן להניח שיש לפחות 0 אחד).

2. כתבו תוכנית המממשת את המשחק "חם קר". שחקן המשחק נגד התוכנית בוחר מספר שלם K 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 ממיון 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 ממיון 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 "ממיון מעגלית"; כלומר מאיבר מסוים בו שאינו בהכרח האיבר הראשון, הוא ממיון 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 ממיון 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 ממיון) 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 ממיון 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 ממיון) 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".

צינו 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 ושאינו ערכים זהים בתוך מערך B , אך ייתכן שיש ערכים הקיימים גם ב- A וגם ב- B . שנו את פעולת ה**מיוזג** 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 ה**ממוזג** קיים לפחות נציג אחד מכל אחד ממערכי המקור. קטע התוכנית יציג הודעה מתאימה.

3. נתונים שני מערכים ממוינים. כתבו קטע תוכנית המייצר את המערך ה**ממוזג** 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 ה**ארוכה 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 ממיוזג 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.

הצ'יה 1 מכירות 2004 תשס"ד, false 10

מטרת הבעיה ופתרונה: הצגת חיפוש במערך דו-ממדי.

נתון מערך דו-ממדי בגודל $N \times N$ 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 2×2 . לדוגמה, לפניכם מערך דו-ממדי בגודל 5×5 . 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:

- פעולה למציאת "רביעייה k". הפעולה תקבל את המספר k ותחזיר true אם קיימת "רביעייה k" במערך ו-false אחרת.
- 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.

בעיה זו נדרשנו להגדיר מחלקה המכילה מערך דו-ממדי כתכונה ושתי פעולות הפעולות על המערך. בשלב ראשון, נגדיר בצורה מדויקת כיצד נייצג את התכונות וכיצד נממש כל פעולה. לאחר מכן, נממש את המחלקה בשפת C#.

הגדרת המחלקה רביעייה K

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 מימין, 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))
    {
        // מצאנו רביעייה k
    }
}
```

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

```
/*
    המחלקה רביעייה k
*/
public class QuadrupletK
{
    // הגדרת התכונות
    private int[,] matrix; // המערך
    private const int MAX_NUM = 30;
    // פעולה בונה
    public QuadrupletK(int[,] mat)
    {
        matrix = mat;
    } // QuadrupletK
    // בדיקה אם קיימת במערך רביעייה k
    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. לדוגמה: 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 3×3 . 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".

דוגמה: במערך בגודל 5×4 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 מצוין שורה, והמספר השני מצוין 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.

פרק 2 ההצגות והפונקציות 2004 תשס"ד, פרק 9

מטרת הפרק והפונקציות: הצגת בעיה מורכבת שלפתרון 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.

בעיה 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 אותם 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), ותאחז 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. חשוב, מה היה יכול לקרות אילו החזרנו פשוט את המערך passengers באופן הבא:

```
return passengers;
```

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.WriteLine("Enter passenger name, type 'End' to " +
                           "finish: ");
        passengerName = Console.ReadLine();
        while (passengerName != "End")
        {
            Console.WriteLine("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
    // חיון 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.

תרגיל 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.WriteLine("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; // אִפּוּס המונה maxCount
    }
    if (grade == max)
        maxCount++;
}
Console.WriteLine("{0} students received the maximum grade",
    maxCount);
```

סוף פתרון בעיה 3

בעיה 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.WriteLine("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?

הצ'יה 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 יכול 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, ומעריך 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 1-1 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 יכול 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 "סכום 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.

תצורה 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 כצ'ה 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.

כצ'ה 6

מטרת הבעיה ופתרונה: יחסים בין מערכים ושימוש בפעולות עזר פרטית.

פתחו אלגוריתם שיגדיר מערך דו-ממדי של מספרים שלמים בגודל $N \times M$ (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 תרגיל 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 אותה, 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#.

בצ'יה 7 מבטאות 2007 תשס"ז, סאלה 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 שערך 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 שערך 318 – מייצג את מספר המקומות בחניון.

◆ **HOURLY_RATE** – קבוע שלם שערך 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.WriteLine("Enter 1 for entry, 2 for exit, -1 to end: ");
        action = int.Parse(Console.ReadLine());
        while (action != -1)
        {
            if (action == 1) // כניסה לחניון
            {
                Console.WriteLine("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.WriteLine("Enter the time - " +
                                   "a number between 7-23: ");
                hour = int.Parse(Console.ReadLine());
                Console.WriteLine("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 תרגיל 7

תרגיל 8 מבטאות 2007 תשס"ז, ט"ז

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.

3	7	2	0	1
---	---	---	---	---

דוגמה עבור k=2 ומערך a בגודל 5:

2	0	1	0	0
---	---	---	---	---

ג. לאחר הפעלת הפעולה Shift המערך שיתקבל יראה כך:

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 אות **הבאה** וכל ספרה הופכת להיות הספרה **הקודמת**. ההצפנה פועלת באופן מעגלי כך שהאות הבאה אחרי אות 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 ש"ח, 5 ש"ח, ו-1 ש"ח. 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.

דוגמה: במערך בגודל 4×5 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" ←

← "איבר מוקף"

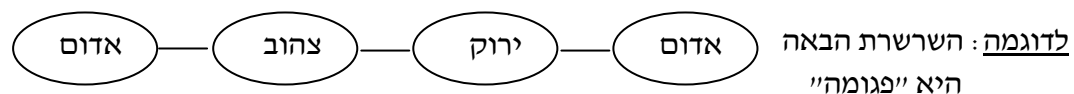
א. כתבו את המחלקה "שרשרת במערך" המכילה מערך דו-ממדי כתכונה, ובה פעולה בונה המקבלת מערך דו-ממדי ומאתחלת את התכונה. ניתן להניח שהמערך המתקבל כפרמטר תקין, כלומר מכיל רק את הערכים 0 ו-1, ושכל שורה או עמודה יש לכל היותר שרשרת אחת.

ב. כתבו פעולה המקבלת שני מספרים שלמים המציינים מיקום של איבר במערך (אינדקסים), המספר הראשון מציינ שורה והמספר השני מציינ 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".



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: 3 3 1 1 2 2 1 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 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 ותמייך אותם לפי שמותיהם. אין צורך לקלוט את שמות היוצרים ואת שמות הדיסקים שלהם. ניתן להניח שמערך היוצרים מאותחל בעצמים מסוג יוצר.

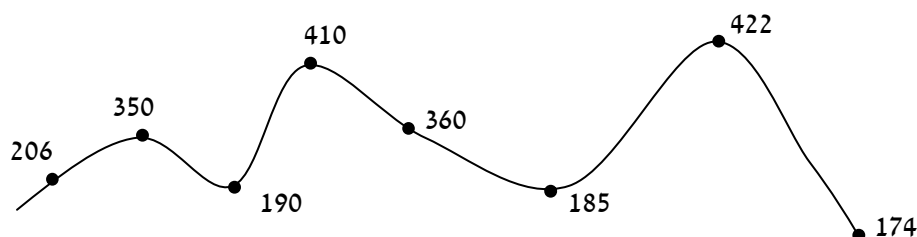
10. (הרחבת שאלה 13.6)

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 1

מדריך מעבדה לסביבת העבודה

Visual C# Express

כתבה: יעל בילצ'יק (סופרין)

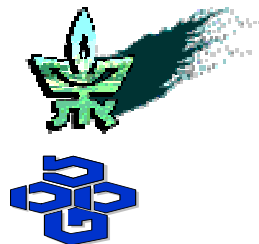
מהדורת עיצוב

תשס"ו 2006

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



אין לשכפל, להעתיק, לצלם, לתרגם או לאחסן במאגר מידע כל חלק שהוא מחומר הלימוד של ספר זה. שימוש מסחרי מכל סוג שהוא בחומר הכלול בספר זה אסור בהחלט, אלא ברשות מפורשת בכתב מהגורמים המפורטים להלן.



כל הזכויות שמורות

אוניברסיטת תל-אביב ומשרד החינוך

תוכן העניינים

5	פתיחת סביבת העבודה.....
5	יצירת פרויקט.....
6	חלונות סביבת העבודה.....
7	כתיבת תוכנית ראשונה.....
8	הידור תוכנית.....
9	הרצת תוכנית.....
10	שמירת תוכנית.....
11	יצירת פרויקט נוסף.....
11	הקלדת קלט.....
12	ניפוי שגיאות.....
15	עבודה מתקדמת עם מנפה השגיאות.....
16	טבלת מקשי קיצור שימושיים לפעולות בסביבת העבודה:.....

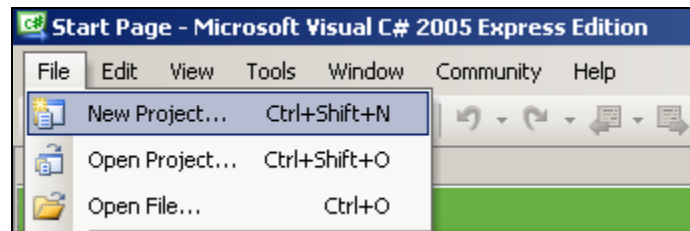
סביבת העבודה Visual C# Express מאפשרת לנו לכתוב תוכניות בשפת C#, החל מתוכניות פשוטות למדי בעלות שורות קוד בודדות, ועד מערכות מסחריות מורכבות בעלות אלפי שורות קוד. במדריך זה נלמד כיצד להשתמש בסביבה זו באופן הנוח ביותר לצרכינו כמתכנתים מתחילים.

פתיחת סביבת העבודה

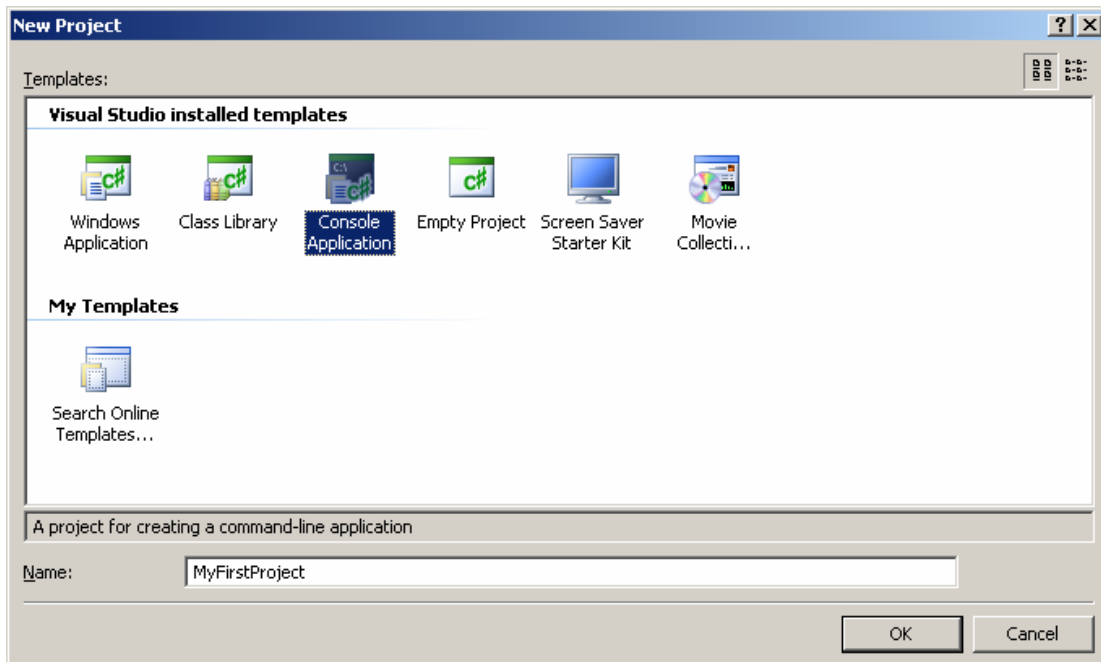
כדי לפתוח את סביבת העבודה נבחר את התוכנה Microsoft Visual C# 2005 Express Edition, דרך תפריט ההתחלה של "windows". יתקבל מסך הפתיחה של סביבת העבודה.

יצירת פרויקט

כל תוכנית בשפת C# נמצאת בתוך פרויקט נפרד. לכן, כדי לכתוב תוכנית חדשה עלינו לפתוח פרויקט חדש. נבחר בתפריט העליון את File, ושם נבחר ב-New Project.



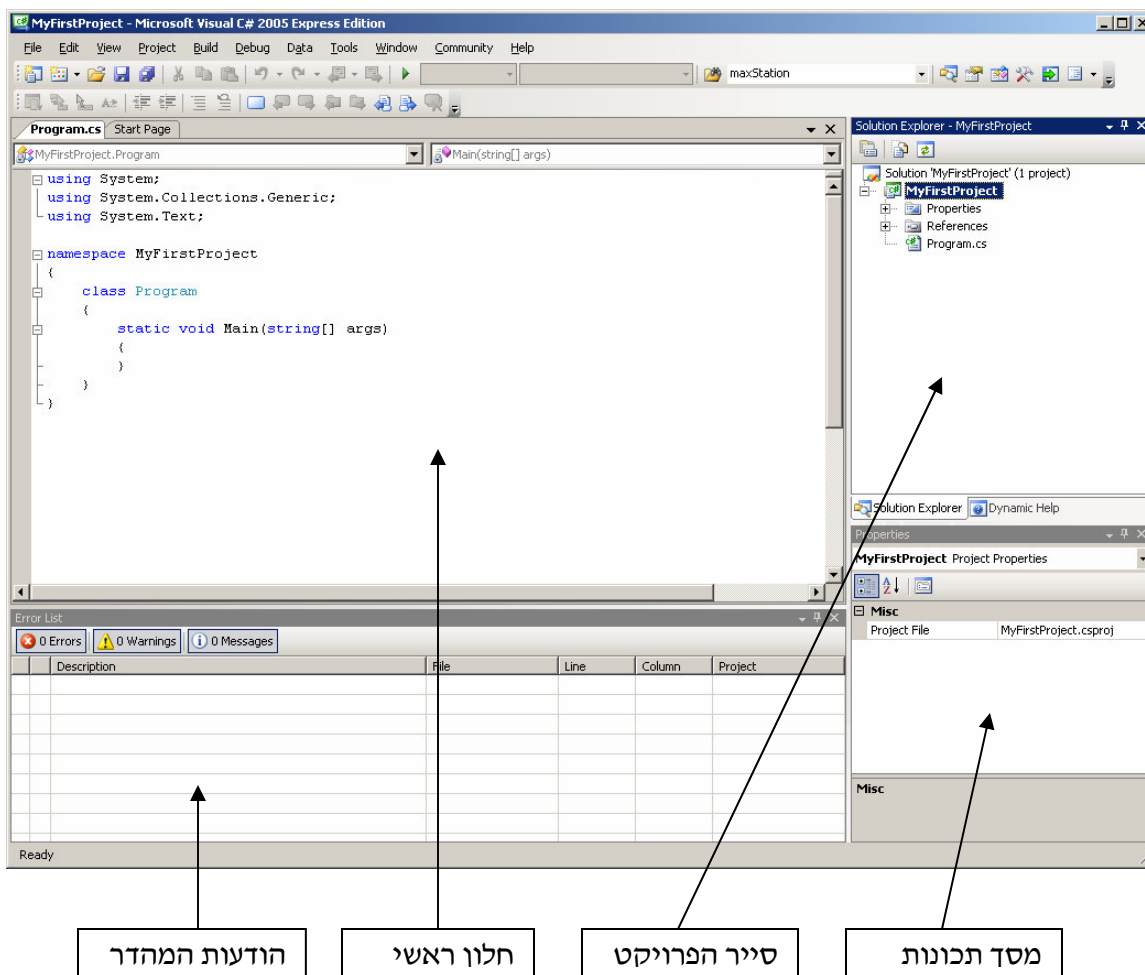
יפתח חלון המציע לנו סוגי פרויקטים שונים. נבחר ב-Console Application. בשורת השם נכתוב את השם שנקרא לפרויקט. בדוגמה זו בחרנו בשם "MyFirstProject".



לאחר הלחיצה על "OK" יתקבל מסך סביבת העבודה, כאשר בחלון המרכזי כבר מופיע בסיס של תוכנית ריקה בשפת C#.

חלונות סביבת העבודה

נכיר את החלונות השונים בסביבת העבודה:



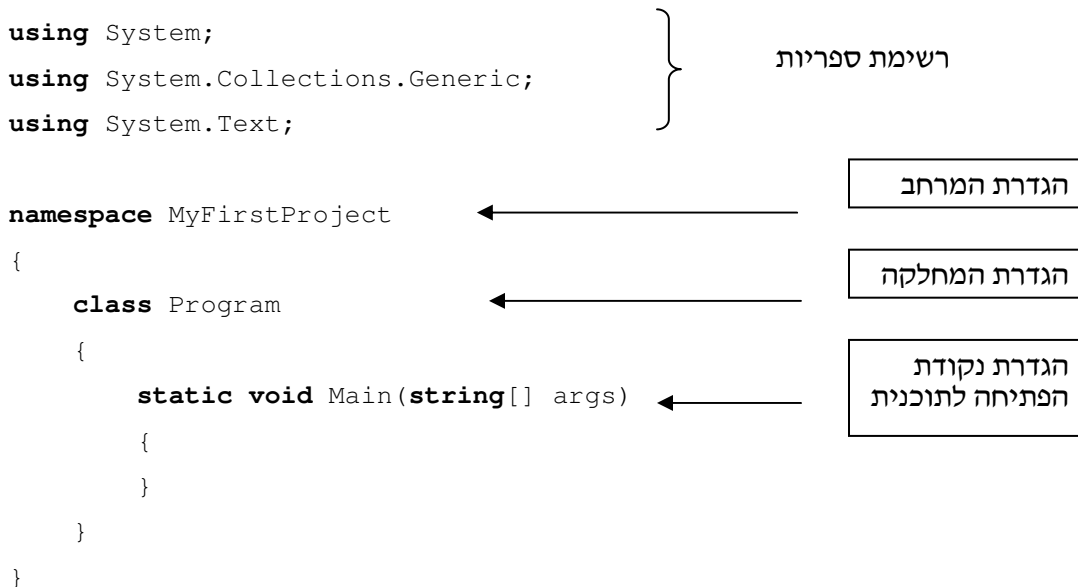
החלון הראשי ישרמש אותנו לכתיבת קוד התוכנית

בחלון **הודעות המהדר** נצפה בשגיאות והערות המהדר (קומפילר) לאחר כל הידור (קומפילציה).

בחלון **סייר הפרויקט** נשתמש על מנת לעבור בנוחות בין חלקי הפרויקט השונים. ניתן לראות כי הפרויקט שפתחנו קיים בתוך Solution. כאשר פתחנו פרויקט חדש נפתח עבורו Solution חדש. משמעות המילה solution היא פתרון. בסביבת העבודה, Solution שומר בתוכו פרויקט אחד או יותר, ומקובל שיהיו אלה פרויקטים שיש ביניהם קשר, והם מהווים במובן מסוים פתרון לבעיה מורכבת אחת. בהמשך נלמד כיצד ניתן ליצור כמה פרויקטים בתוך Solution אחד.

חלון מסך התכונות משמש בעיקר עבור תוכניות בהן נעשה שימוש בעזרים גרפיים ולכן לא נשתמש בו.

ניתן לסגור כל חלון אם אין בו שימוש, ולפתוח אותו שוב בעזרת התפריט View.
נתבונן בקוד (כלומר, רצף הוראות בשפת התכנות) אשר מופיע באופן אוטומטי במסך התוכנית עם פתיחת פרויקט חדש:



לא כל ההגדרות הכרחיות לצרכינו, לכן נוכל למחוק את ההגדרות שלא נזדקק להן ולהישאר עם השלד המוכר לנו:

```
using System;

class Program
{
    static void Main()
    {
    }
}
```

כתיבת תוכנית ראשונה

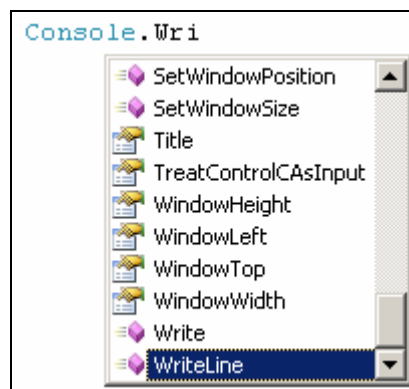
כעת אנו יכולים להקליד את התוכנית הראשונה, שתציג למסך את הפלט "Hello world".
נשנה את שם המחלקה לשם שנרצה לקרוא לתוכנית (אין חובה לשנות את השם, אך רצוי תמיד לתת לתוכניות שמות משמעותיים, שמביעים את תפקידן), ונוסיף את פקודת ההדפסה בתוך תחום ה-
:Main

```
using System;

class HelloWorld
{
    static void Main()
    {
        Console.WriteLine("Hello world");
    }
}
```

שימו ♥:

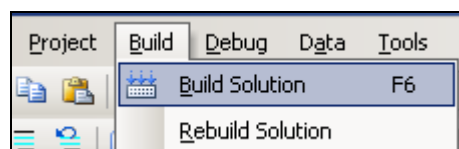
- ישנם צבעים שונים בקוד הכתוב: מילים שמורות נכתבות בכחול, מחרוזות באדום, שמות מחלקות בטורקיז.
- מייד לאחר כתיבת שם המחלקה Console נפתח חלון המציג את כל התכונות והפעולות של המחלקה בהן נוכל להשתמש. לפקודת הדפסה נבחר את הפעולה WriteLine.



- תוכלו לעבור ולבדוק תכונות ופעולות נוספות השייכות למחלקה Console.
- למשל, ההוראה `Console.ForegroundColor = ConsoleColor.Red;` תשנה את צבע הפלט לאדום.
- כדי לקבל הנחיות על כל תכונה ופעולה אפשר לעמוד עם הסמן על התכונה או הפעולה המבוקשת ולהקיש על F1.

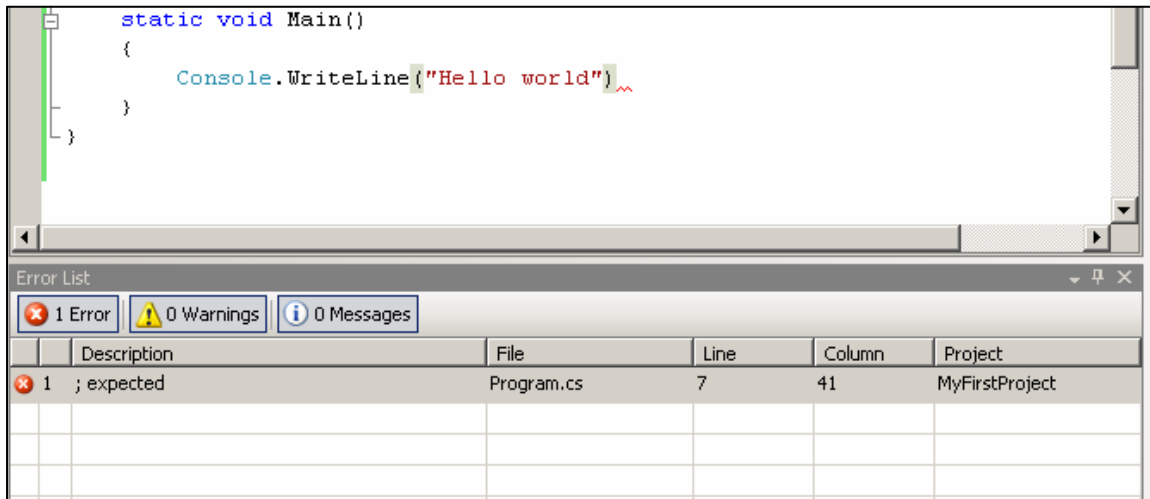
הידור תוכנית

אחרי סיום הקלדת התוכנית, עלינו להדר (לקמפל) אותה כדי לבדוק שאין בה שגיאות תחביריות, ולהכין אותה לריצה. שלב זה יתבצע על ידי בחירת Build בתפריט, ואז בחירת Build Solution, או על ידי הקשה על המקש F6.



אם אין כלל שגיאות תחביר בתוכנית, נקבל את ההודעה Build succeeded בצד השמאלי התחתון של המסך. אם קיימות שגיאות תחביר בתוכנית נקבל הודעות מתאימות בחלון הודעות המהדר. לחיצה כפולה על ההודעה תקפיץ את הסמן למקום בו ארעה השגיאה.

למשל, השגיאה הבאה נגרמה כיוון שלא נכתב הסימן ; בסוף משפט:



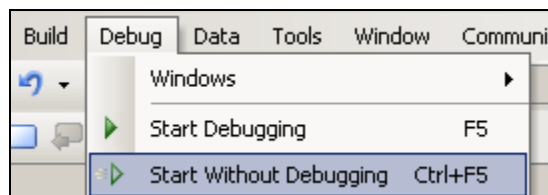
לא נוכל להריץ תוכנית לפני שנתקן את כל שגיאות התחביר. לאחר שנתקן את השגיאות, נהדר שוב את התוכנית, וכך נחזור על התהליך עד אשר לא יופיעו שגיאות בחלון הודעות המהדר, ונקבל את ההודעה Build succeeded.

שימו ♥: ייתכן שלאחר הידור של תוכנית נקבל **הערות** הרצה. ההערות מסומנות בסימן צהוב (בעוד השגיאות מסומנות באדום). ניתן להריץ תוכנית שהתקבלו עבורה הערות, אך יש לתת את הדעת על הערות אלה, משום שייתכן שהן מצביעות על טעות או על בעיה אפשרית אחרת בתוכנית.

הרצת תוכנית

לאחר שהתוכנית עברה בהצלחה את שלב ההידור, היא מוכנה להרצה.

כדי להריץ את התוכנית נבחר בתפריט Debug את Start Without Debugging או נקיש על המקשים Ctrl+F5 או F5 בזמנית.



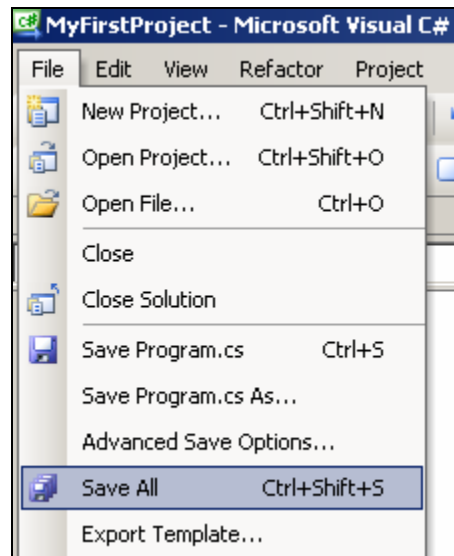
כתוצאה מכך, יפתח חלון ריצת התוכנית, ייכתב בשורה נפרדת המשפט "Hello world", וריצת התוכנית תסתיים. בסיום ריצת התוכנית חלון ההרצה ייסגר רק לאחר הקשה על מקש כלשהו.

```
Hello world
Press any key to continue . . . _
```

שימו ♥: אם תנסו להריץ תוכנית אשר לא עברה הידור בהצלחה, או תוכנית שעברה הידור אך לאחר מכן עברה שינוי כלשהו ויש להדרה שוב, יתבצע הידור באופן אוטומטי.

שמירת תוכנית

לאחר שסיימנו לכתוב את התוכנית, להדר אותה (תוך תיקון שגיאות תחביר, במידת הצורך), להריץ אותה, ולבדוק שאין שגיאות לוגיות בתוכנית, נשמור את קובץ התוכנית לשימוש עתידי. בתפריט File בחרו את Save All או הקישו על המקשים Ctrl, Shift ו-S בו זמנית.



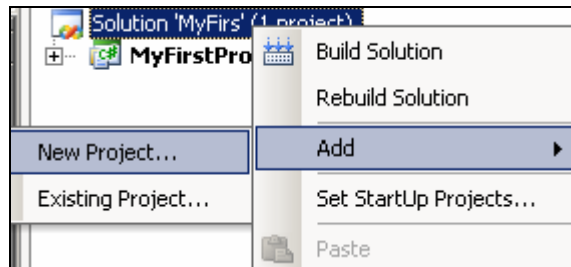
יפתח חלון בו יופיע שם הפרויקט, הכתובת על גבי הדיסק הקשיח בה יישמר הפרויקט, ושם ה-Solution.

כאשר נרצה בעתיד לפתוח תוכנית שמורה נפתח את הפרויקט שלה על ידי Open project שבתפריט File, ונבחר את ה-Solution הנדרש לפי שמו בסיומת .sln.

יצירת פרויקט נוסף

כעת ברצוננו לכתוב תוכנית נוספת, הדורשת גם הזנת קלט. לפנינו שתי אפשרויות: לסגור את ה-Solution הקיים על ידי Close Solution שבתפריט File, ולפתוח פרויקט חדש כפי שעשינו בתחילה.

אפשרות שנייה היא להוסיף פרויקט ל-Solution קיים. לשם כך ניגש עם העכבר למילה Solution שבסייר הפרויקט, נלחץ על המקש הימני של העכבר ונבחר ב-Add. מהתפריט שיפתח נבחר את NewProject.



כעת ייפתח לנו חלון פרויקט חדש, כפי שקרה כאשר פתחנו Solution חדש.

שימו לב שלאחר הוספת הפרויקט קיימים שני פרויקטים תחת אותו ה-Solution, אך רק אחד מהם פעיל – זה המסומן בהדגשה. ניתן להחליף ולסמן פרויקט שונה כפעיל על ידי בחירתו עם העכבר, הקשה על מקש ימני ובחירת Set as StartUp Project.

כאשר אנו מוסיפים פרויקט ל-Solution, המכיל כבר פרויקטים אחרים, הרי שבסופו של דבר הפרויקטים יישמרו יחדיו, תחת אותו Solution (ואיתם גם התוכנית שכתבתם בכל אחד מהם). לכן, אם אתם כותבים כמה תוכניות בעלות אופי דומה שברצונכם לשמור יחדיו (למשל, כמו כמה תרגילים עבור עבודה אחת להגשה באותו נושא) מומלץ לשמור אותן בפרויקטים נפרדים תחת אותו ה-Solution. אך אם הינכם כותבים כמה תוכניות שאין כל קשר ביניהן, מומלץ לשמור כל אחת מהן ב-Solution נפרד עם שם משמעותי שיקל עליכם את הזיהוי של התוכנית כאשר תחפשו אותה בקבצים השמורים.

הקלדת קלט

כידוע, זהבה גרמה נזק רב לשלושת הדובים, ולכן החליטה לפצותם בסכום כסף. את הסכום תחלק שווה בשווה בין כל השלושה. הדוב הקטן החליט שאת סכום הכסף שקיבל יחלק לארבעה חסכוניות נפרדים, ואת השארית יבזבז על צנצנת דבש איכותית. עלינו לכתוב תוכנית שתקבל כקלט את סכום הכסף שהקצתה זהבה לתשלום הפיצויים, ותציג כפלט את סכום הכסף שיוכל הדוב הקטן לבזבז על צנצנת דבש איכותית.

הנה תוכנית שנכתבה לצורך פתרון הבעיה, כפי שהוקלדה בסביבת העבודה:

```
using System;

class Bears
{
    static void Main()
    {
        int money, smallBearSum, sumForHoney;
        Console.Write("Insert the sum of money Zeahva has: ");
        money = int.Parse(Console.ReadLine());
        smallBearSum = money / 3;
        sumForHoney = smallBearSum / 4;
        Console.WriteLine("The bear junior will spend {0} shekels for
honey", sumForHoney);
    }
}
```

הקלידו גם אתם את התוכנית הזאת, והדרו אותה.

לאחר שהתוכנית עברה בהצלחה את תהליך ההידור, נריץ אותה.

בתחילה יירשם במסך ריצת התוכנית משפט הפלט

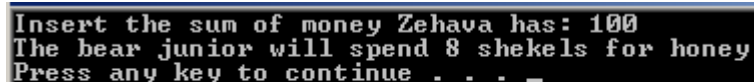
"Insert the sum of money Zeahva has: "

ולאחר מכן התוכנית תעצור פעולתה ותמתין לקלט של מספר שלם. עלינו להקליד בהמשך למשפט זה מספר שלם כלשהו ואחריו להקיש על המקש Enter. רק אז התוכנית תמשיך את ריצתה, תציג את הפלט המחושב ותסיים את פעולתה.

שימו ♥: אם התוכנית ממתינה לקבל מספר שלם, אך אנו נקליד קלט שאינו מספר שלם, יגרום הדבר לשגיאת ריצה. ייפתח מסך בו תופיע השאלה: האם ברצונכם לנפות את השגיאות באמצעות כלי לניפוי שגיאות? לחצו על "No". אז תופיע הודעת השגיאה על גבי מסך התוכנית, והתוכנית תפסיק ריצתה.

ניפוי שגיאות

נתבונן במסך ריצת התוכנית לאחר ריצתה, כאשר הקלט הוא 100:



```
Insert the sum of money Zehava has: 100
The bear junior will spend 8 shekels for honey
Press any key to continue . . . _
```

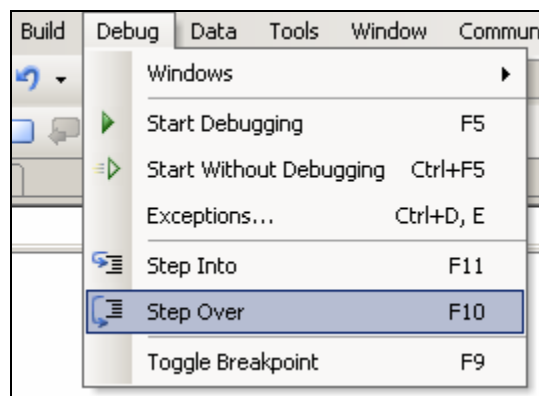
נבחן את התוצאה שהתקבלה:

אם זהבה הקצתה 100 ₪ לפיצויים, הרי כל דוב יקבל 33 ₪. הדוב הצעיר יחלק את הסכום שקיבל, 33 ₪, ל-4 קבוצות ובשארית שיקבל יקנה צנצנת דבש איכותית. שארית החילוק של 33 ב-4 היא 1, לכן הפלט צריך להיות 1. אם נתבונן במסך הפלט נראה כי הפלט הוא 8. מכאן, שיש בתוכנית שכתבנו שגיאה לוגית, שגיאת חישוב במהלך התוכנית.

בתוכנית קצרה כגון זו שכתבנו ניתן להתבונן בתוכנית ולמצוא את השגיאה בקלות יחסית. אך כאשר התוכנית גדולה ומורכבת יותר ניפוי השגיאות הופך למשימה קשה הרבה יותר. לשם כך קיים כלי המאפשר לנו לנפות שגיאות ביתר קלות, ה-Debugger.

בעזרת ה-Debugger ניתן להריץ את התוכנית באופן מבוקר, שורה אחר שורה, בכל שורה נצפה בערך המשתנים ונבדוק כי ערכם תואם לערך הצפוי.

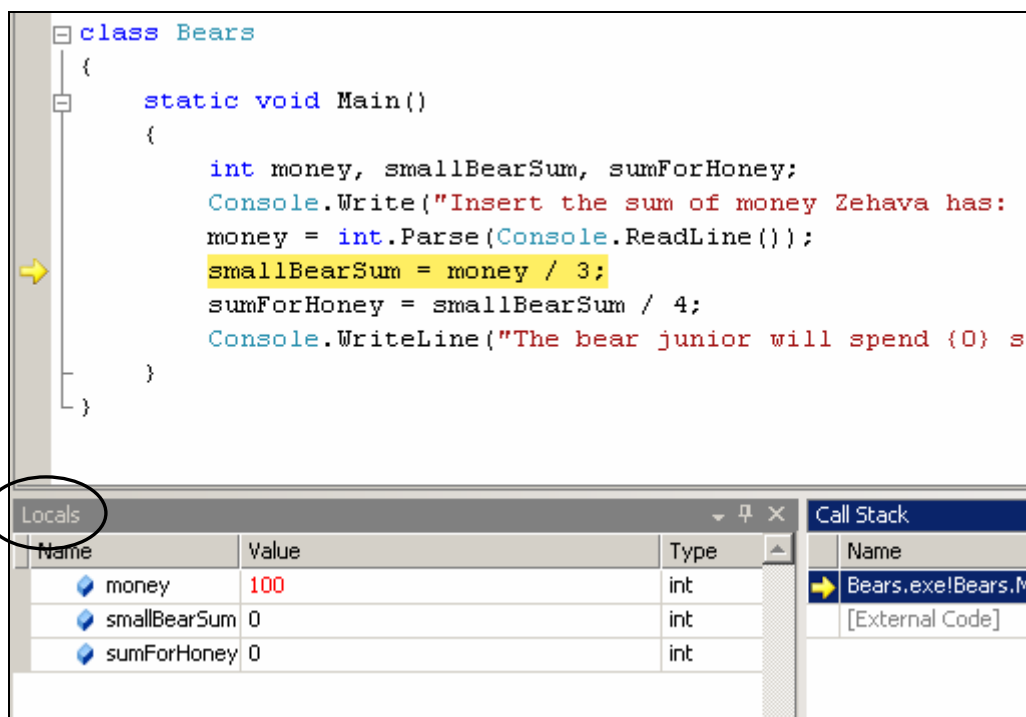
לתחילת הרצה מבוקרת של התוכנית בחרו מהתפריט Debug את Step Over או הקישו על F10.



ריצת התוכנית תחל והשורה הראשונה של התוכנית תיצבע בצהוב. כעת, כל הקשה על F10 תגרום להתקדמות התוכנית לשורה הבאה. לחצו על F10 עד אשר תתבצע שורת הפלט הראשונה וראו כי אכן נרשמה על מסך ריצת התוכנית שורת הפלט. לחיצה נוספת על F10 תגרום להמשך ריצת התוכנית אל השורה הבאה, שורת הקלט. מכאן תוכלו להמשיך את ריצת התוכנית רק לאחר שתזינו את הקלט המבוקש. הקלידו את המספר 100 במסך ריצת התוכנית והקישו Enter. כעת חזרה השליטה לתוכנית.

נתבונן במסך "Locals" הנפתח בצד שמאלי תחתון של המסך:

במסך "Locals" נוכל לצפות בערכי המשתנים בעת ריצת התוכנית. שימו לב לרגע המתואר בצילום: על פי השורה הצבועה בצהוב ניתן לדעת כי כבר נקרא הקלט 100 ממסך ריצת התוכנית, והושם במשתנה money. נתבונן במשתנה זה במסך "Locals" ונבחין כי אכן המשתנה קיבל את הערך 100. הערך צבוע בצבע אדום מכיוון ששורת התוכנית האחרונה שהתבצעה גרמה לשינוי ערך המשתנה.



השורה הבאה בתוכנית אמורה לחשב את ערך המשתנה `smallBearSum`. לפני שנמשיך את ריצת התוכנית נחשוב מה ערכו של משתנה זה אמור להיות. כפי שחישבנו קודם, כל דוב יקבל 33 ₪ ולכן זה צריך להיות ערכו החדש של המשתנה. נמשיך את ריצת התוכנית על ידי הקשה נוספת על המקש F10. התבוננו במסך "Locals" וראו כי המשתנה `smallBearSum` שינה כעת את ערכו מ-0 ל-33, כצפוי. אם כך, עד עתה התוכנית עבדה באופן תקין.

השורה הבאה אמורה לחשב את ערך המשתנה `sumForHoney`. שוב, נחשוב מה אמור להיות ערכו של משתנה זה. כפי שחישבנו קודם, שארית החלוקה של 33 ב-4 היא 1, ולכן זו התוצאה המבוקשת. הקשה נוספת על המקש F10 תמשיך את ריצת התוכנית לשורה הבאה, וערכו של המשתנה `sumForHoney` משתנה לערך 8, ולא לערך הצפוי. אם כך, בשורה זו ישנה שגיאה.

נתבונן בשורת הקוד הבעייתית:

```
sumForHoney = smallBearSum / 4;
```

כוונתנו הייתה לחשב את **שארית החילוק** של `smallBearSum` ב-4, ובמקום זאת חישבנו את **תוצאת החילוק** של `smallBearSum` ב-4. שגיאה זו אירעה כיוון שהשתמשנו בסימן / במקום בסימן %.

נתקן את השגיאה בקוד התוכנית כך שהשורה תיראה כעת כך:

```
sumForHoney = smallBearSum % 4;
```

לאחר שתיקנו את השגיאה, נרצה לבדוק האם כעת התוכנית נכונה ומציגה את הפלט הנכון.

נוכל לבצע זאת באחת משתי הדרכים הבאות:

1. נעצור את ריצת התוכנית (על ידי Stop Debugging שבתפריט Debug, או על ידי הקשה על המקשים shift ו-F5 בו זמנית), ולאחר מכן נריץ את התוכנית שוב, לאחר השינוי, מההתחלה, על ידי ריצה רגילה של התוכנית, או על ידי ריצה תוך כדי ניפוי שגיאות.

2. נגרום לשורה שתיקנו להתבצע שוב: משמאל לשורות הקוד הצבוע בצהוב (השורה הבאה לביצוע) נבחין בחץ צהוב. נגרור את החץ הצהוב חזרה לשורת הקוד המתוקנת על ידי העכבר, כך ששורה זו תתבצע שוב מחדש, ונמשיך את ריצת התוכנית על ידי המקש F10 עד לסיומה.

נחזור על תהליך ניפוי השגיאות עבור כל שורה בתוכנית, ונבצע את המעקב עד אשר נהיה בטוחים כי התוכנית מספקת פלט נכון עבור כל קלט אפשרי.

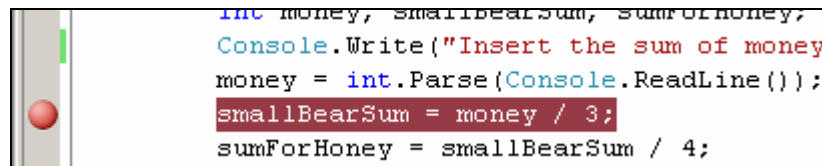
שימו ♥: בשלב זה שימוש במקש F11 יהיה זהה עבורנו לשימוש במקש F10. בהמשך לימודינו, כאשר נרצה להיכנס ולבדוק פעולות בקוד שנכתבו על ידינו, נשתמש במקש F11, ואילו מעבר על פניה מבלי להיכנס ולבדוק אותן יבוצע על ידי המקש F10.

עבודה מתקדמת עם מנפה השגיאות

בהמשך לימודינו נכתוב תוכנית ארוכות, ולא נרצה לעבור על כל שורות הקוד בחיפושנו אחר שגיאה. לכן, אפשרות נוספת לעבודה עם ה-Debugger היא על ידי הרצת התוכנית באופן רציף, עד נקודה מסוימת, בה תעצור התוכנית את ריצתה. לשם כך נשתמש בנקודות עצירה, ה-Breakpoint.

למשל, נניח שאנו משוכנעים כי בתוכנית Bears קליטת נתון הקלט נעשתה כיאות ואין צורך לבדוק זאת. אם כך, אנו יכולים להציב נקודת עצירה בשורה שאחרי קליטת הקלט. לשם הצבת נקודת עצירה בשורה מסוימת נעמוד עם הסמן על השורה המבוקשת, ונבחר את Toggle Breakpoint שבתפריט Debug, או שנקיש על המקש F9. דרך נוספת להצבת נקודת עצירה היא על ידי לחיצה על המקש השמאלי של העכבר בשטח האפור שמשמאל לשורת הקוד המבוקשת.

כתוצאה מכך תופיע נקודה אדומה משמאל לשורת הקוד המבוקשת, והשורה תיצבע באדום.



כעת נריץ את התוכנית, אך לא באופן הרגיל, אלא במצב ניפוי שגיאות, על ידי בחירת Start Debugging בתפריט Debug, או על ידי הקשה על המקש F5.

התוכנית תתחיל את ריצתה, תדפיס את שורת הפלט, תעצור לבקש קלט, נקליד 100 ו-Enter, וכעת, כאשר התוכנית תגיע לשורה המסומנת על ידי נקודת-עצירה, תפסיק את ריצתה ותעצור במצב Debugging המוכר לנו. ממצב זה, כפי שכבר ראינו, ניתן להמשיך להריץ שורה אחר שורה על ידי

המקש F10. אפשרות נוספת היא להוסיף נקודות-עצירה נוספות לאורך התוכנית ולרוץ מאחת לשנייה על ידי המקש F5. בכל הקשה על F5 התוכנית תרוץ מהמקום האחרון בו היא עצרה ועד נקודת-העצירה הבאה. אם לא תמצא נקודת-עצירה נוספת, תרוץ התוכנית עד לסיומה.

טבלת מקשי קיצור שימושיים לפעולות בסביבת העבודה:

מקש קיצור	הסבר	פעולה
F6	הידור התוכנית	Build Solution
Ctrl + F5	הרצת התוכנית	Start Without Debugging
F5	הרצת תוכנית עד נקודת עצירה	Start Debugging
F10	קידום הרצת התוכנית בשורה אחת מבלי להיכנס לפעולות בקוד	Step Over
F11	קידום הרצת התוכנית בשורה אחת כך שניכנס לפעולות בקוד שנכתבו על ידינו	Step Into
F9 או מקש שמאלי של העכבר	הצבה או הסרה של נקודת עצירה על השורה בה נמצא הסמן	Toggle Breakpoint
Ctrl+E, D	הזחת התוכנית (אינדנטציה)	Format Document

אינדקס

הערכים באינדקס שלפניכם מפנים את הקריאה לשני הכרכים של יסודות. הערכים המפנים ליסודות 1 מסומנים באות א.

סימנים

! 196 א

=! 100 א

% 70 א

&& 113 א

/ 70 א

{ } 26 א

|| 118 א

++ 157 א

> 100 א

=> 100 א

= 36 א, 35 א

== 100 א

< 100 א

=< 100 א

≠ 100 א

[] – פנייה לתו במחרוזת 11, 22

א

או 111 א, 116 א, 118 א

אורך

מחרוזת 3, 20

מערך 25, 27

אינדקס של מערך 26, 30

אלגוריתם 13 א

אלכסון משני 123

אלכסון ראשי 123

אקראי

הגרלת מספר 85 א

אתחול 38 א

אתחול מערך 27, 30

ב

בחירה ממצה של דוגמאות קלט 147 א

ביטוי בוליאני 98 א, 114 א, 118 א

ביצוע-חוזר 18 א, 155 א

ביצוע-חוזר-בתנאי 174 א

ביצוע-חוזר מספר פעמים ידוע מראש 156 א

זקיף 174 א, 178 א

לולאה אינסופית 187 א, 189 א
 לולאת for 157 א
 לולאת while 176 א
 קינון 198 א, 200 א
 ביצוע-מותנה 16 א, 95 א, 99 א
 if 99 א
 ביטוי בוליאני 98 א, 114 א, 118 א
 הוראת שרשרת לביצוע-בתנאי 129 א
 קינון של הוראה לביצוע-בתנאי 122 א
 קשרים לוגיים
 או 111 א, 116 א, 118 א
 וגם 111 א, 112 א, 114 א
 לא (not) 196 א
 תנאי מורכב 111 א
 בניית מספר 80 א, 93 א, 211 א

ג

וגם 111 א, 112 א, 114 א

ד

דוגמאות קלט מייצגות 147 א

ה

הגרלת מספר 85 א
 הוראת בחירה (switch) 133 א, 99
 הוראת פלט 25 א
 הוראת קלט 29 א
 הזזה מעגלית 60 א, 62 א
 החזרת ערך 67 א, 66, 69
 המרת אותיות גדולות לקטנות 81 א, 15, 22
 המרת אותיות קטנות לגדולות 22
 המרת טיפוסים 49 א, 75 א, 83 א
 הערות 27 א
 הפניה 3, 94, 105
 הפניה המועברת כפרמטר 110
 הפניה עצמית (this) 72, 74
 הקצאת מקום בזיכרון 1
 הקצאת מקום עבור מערך 26, 30
 הרשאות גישה
 פרטית 64
 ציבורית 64
 השוואה – אופרטורים 100 א
 השוואת מחרוזות 7, 22
 השמה 33 א, 30 א

ז

זיכרון 3 א
זמן ביצוע של אלגוריתם 213 א
זקיף 174 א, 178 א

ח

חומרה 2 א, 8 א
חזקה 67 א
חיפוש בינרי 214 א, 131
חיפוש תו במחרוזת 11
חיפוש תת-מחרוזת 22, 23
חלוקה בשלמים 68 א
שארית 69 א, 70 א
חריגה מגבולות המערך 41

ט

טבלת אמת
או 118 א
וגם 114 א
לא (not) 196 א
טבלת מעקב 39 א
טיפוס 29 א, 46 א
טיפוס בוליאני 191 א
טיפוס המחרוזת 1
טיפוס ממשי 47 א, 48 א
טיפוס שלם 29 א
טיפוס תווי 80 א

י

יחידת עיבוד מרכזית 3 א
יעילות
יעילות זמן 213 א
יעילות מקום 50, 52
יצירת עצם 85 א, 1, 67

ל

לא (not) 196 א
לולאות 18 א, 155 א
ביצוע-חוזר-בתנאי 174 א
ביצוע-חוזר מספר פעמים ידוע מראש 156 א
זקיף 174 א, 178 א
לולאה אינסופית 187 א, 189 א
לולאת for 157 א
לולאת while 176 א
קינון 198 א, 200 א

מ

מהדר 6 א

מונה 162 א, 203 א, 163

מחלקה 26 א, 1, 63

הגדרת מחלקה 63, 68

הפניה עצמית (this) 72, 74

יצירת עצם 85 א, 1, 67

מחלקה מתמטית 67 א

מחלקת שירות 96, 179

עצם 1, 63

פעולה בונה 70, 71, 74

פעולות גישה 75, 79, 84

פעולות סטטיות 96, 179

פעולות של עצם 63, 69

מחרוזת 1

אורך 3

המרת אותיות גדולות לקטנות 81 א, 15, 22

המרת אותיות קטנות לגדולות 22

השוואת מחרוזות 7, 22

השמה במחרוזות 12

חיפוש תו במחרוזת 11

חיפוש תת-מחרוזת 22, 23

מיקום תו במחרוזת 4, 21

פנייה לתו במחרוזת 4

פעולות על מחרוזות 5, 21

שרשור מחרוזות 9

מיון בועות 146, 162

מיון בחירה 136

מיון הכנסה 139, 143

מיזוג 147

מינימום 67 א

מציאת מינימום בסדרה 169 א, 208 א, 47

מציאת מיקום המינימום בסדרה 172 א, 209 א, 47

מיקום תו במחרוזת 4, 21

ממוצע 47 א, 61 א, 26

מספר אקראי 85 א

מספר ממשי 47 א

מספר שלם 29 א

מעבר על זוגות סמוכים בסדרה 126

מערך 25, 26

אורך 27, 30

איברי מערך 27

אינדקס של מערך 26, 30

אתחול מערך 27, 30
 החזרת עותק של מערך 109
 הקצאת מקום עבור מערך 26, 30
 חריגה מגבולות המערך 41
 יחסים בין מערכים 170
 מערך כערך החזרה מפעולה 107
 מערך כתכונה 85, 159
 מערך מונים 46, 59, 166
 מערך צוברים 50, 60
 מערך של עצמים 90, 94, 159
 מציין של מערך 26, 30
 סריקה של מערך 59
 מערך דו-ממדי 115
 אורך 116
 אלכסון משני 123
 אלכסון ראשי 123
 גודל 116
 מערך דו-ממדי ריבועי 123
 סריקת מערך דו-ממדי 115
 מציינים 26, 30
 מקסימום 67 א
 מציאת מיקום המקסימום בסדרה 172 א, 209 א
 מציאת מקסימום בסדרה 143 א, 169 א, 208 א, 163
 משתנים 28 א
 טיפוס בוליאני 191 א
 טיפוס המחרוזת 1
 טיפוס ממשי 47 א, 48 א
 טיפוס שלם 29 א
 טיפוס תווי 80 א
 משתנה מקומי (לוקאלי) 88

נ

ניפוי שגיאות 7 א
 נכונות אלגוריתם 147 א

ס

סריקת מערך דו-ממדי 115
 סריקת מערך חד-ממדי 59

ע

עיגול מספר ממשי 67 א
 עצם 1, 63
 העברת עצם כפרמטר לפעולה 101, 104
 הפניה 3, 94, 105
 הפניה המועברת כפרמטר 110

הפניה עצמית (this) 72, 74
יצירת עצם 85 א, 1, 67
פעולות של עצם 63, 68
תכונות של עצם 63, 69
ערך המוחזר מפעולה 67 א, 66, 69
ערך מוחלט 67 א

פ

פירוק מספר דו ספרתי לספרותיו 77 א, 92 א
פירוק מספר לספרותיו 183 א, 210 א
פיתוח אלגוריתם בשלבים 63 א
פלט 1 א
הוראת פלט 25 א
פלינדרום 95 א
פסאודו-קוד 14 א
פעולות
פעולה בונה 70, 71, 74
פעולה ראשית 26 א
פעולות גישה 75, 79, 84
פעולות סטטיות 96, 179
פעולות על מחרוזות 5, 21
פעולות של עצם 63, 69
פרטי (private) 64
פרמטר 67 א, 5
העברת עצם כפרמטר לפעולה 101, 104
הפניה המועברת כפרמטר 110

צ

צובר 162 א, 206 א
ציבורי (public) 64

ק

קבוע 51 א
קינון
הוראה לביצוע-בתנאי 122 א
הוראה לביצוע-חוזר 198 א, 200 א
קלט 1 א
הוראת קלט 29 א

ש

שארית חלוקה בשלמים 69 א, 70 א
שגיאות
ניפוי שגיאות 7 א
נכונות אלגוריתם 147 א
שורש ריבועי 65 א, 67 א
שילוב תבניות 163

שלילה (not) 196 א

שפת מכונה 5 א

שפת תכנות 5 א

שרשור מחרוזות 9

ת

תבניות 21 א

שילוב 163

תו 80 א

תוכנה 2 א, 5 א, 9 א

תחום 27 א

תיעוד 27 א

תכונות של עצם 63, 68

תנאי 16 א, 95 א, 99 א

if 99 א

ביטוי בוליאני 98 א, 114 א, 118 א

הוראת שרשרת לביצוע-בתנאי 129 א

קינון של הוראה לביצוע-בתנאי 122 א

קשרים לוגיים

או 111 א, 116 א, 118 א

וגם 111 א, 112 א, 114 א

לא (not) 196 א

תנאי מורכב 111 א

תת משימות 64 א

A

⌘ 67 Abs
⌘ 114 , ⌘ 112 , ⌘ 111 and
26 , 25 array
68 , 63 attribute

B

131 , ⌘ 214 binary search
⌘ 193 , ⌘ 191 bool
162 , 146 bubble sort

C

⌘ 83 , ⌘ 76 , ⌘ 49 casting
⌘ 82 , ⌘ 80 char
63 , 1 , ⌘ 26 class
22 , 7 CompareTo
⌘ 6 compiler
⌘ 25 Console
⌘ 51 constant
⌘ 52 const
74 , 71 , 70 constructor
163 , ⌘ 203 , ⌘ 162 counter
⌘ 3 CPU

D

⌘ 7 debug
⌘ 70 division
⌘ 48 double

E

22 , 7 Equals

F

⌘ 157 for

I

⌘ 99 , ⌘ 96 , ⌘ 16 if
⌘ 132 if else if
30 , 26 index
21 , 11 IndexOf
143 , 139 insertion sort
⌘ 29 int

L

30 , 27 Length (array)
3 Length (string)

M

⌘ 26 Main
⌘ 67 Math
⌘ 67 Max
147 merge
69 ,63 method
⌘ 67 Min
⌘ 70 modulus

N

67 ,1 ,⌘ 85 new
⌘ 196 not

O

63 ,1 object
⌘ 118 ,⌘ 126 ,⌘ 111 or

P

⌘ 67 Pow
64 private
64 public

R

⌘ 85 Random
69 ,66 return
⌘ 67 Round

S

136 selection sort
⌘ 67 ,⌘ 65 Sqrt
179 ,96 static method
1 string
23 ,22 Substring
99 ,⌘ 137 switch
⌘ 25 System

T

74 ,72 this
22 ,15 ToLower
22 ToUpper
⌘ 47 ,⌘ 29 type

U

⌘ 25 using

V

void 65, 69

W

while 177 א

Write 29 א

WriteLine 25 א

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

יסיטת 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 מהמוציא לאור 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 ב**מה** ולא ב**איך**. 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 ולתקן. 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

קצ"ה 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 גפריט-בוירה()
2. קאוט אג בויהג המאמאש - choice
3. אא ערכו א choice הא 1
- 3ייר-ריבוע()
4. אא ערכו א choice הא 2
- 3ייר-מלבן()
5. אא ערכו א choice הא 3
- 3ייר-משולש()

התכנית

```
using System;
public class Shapes
{
    --- פעולה המציירת ריבוע בגודל 5x5 ---
    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();
    }
}

//--- 5 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 drawTriangle(), drawRectangle(), drawSquare() 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 יכול: 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)

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: `ShowMenu()`, `Console.WriteLine()`.

את הפעולות אפשר לכתוב במחלקה הראשית לפני הפעולה Main או אחריה.

סוף פתרון כעיה 1

2. פעולה המקבלת פרמטרים ואינה מחזירה דבר

כעיה 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 ---

(שם-פרמטר-2 **טיפוס-פרמטר-2**, שם-פרמטר-1 **טיפוס-פרמטר-1**) שם-הפעולה **static void**

```
{
    // 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 המוזמנת. 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 ישפיעו או ישנו את ערכי הארגומנטים בתכנית המוזמנת.

התכנית המקבלת את גודל הצורה הגיאומטרית

```
using System;
public class Shapes2
{
    //--- טענת כניסה: side אורך צלע הריבוע ---
    //--- טענת יציאה: מצויר 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: מצויר 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: מצויר 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 sidel, side2;

    ShowMenu();
    Console.Write("Type shape number --> ");
    choice = int.Parse(Console.ReadLine());

    if (choice == 1)
    {
        Console.Write("type square side --> ");
        sidel = int.Parse(Console.ReadLine());
        DrawSquare(sidel);
    }

    if (choice == 2)
    {
        Console.Write("type rectangle length --> ");
        sidel = int.Parse(Console.ReadLine());
        Console.Write("type rectangle width --> ");
        side2 = int.Parse(Console.ReadLine());
        DrawRectangle(sidel, side2);
    }

    if (choice == 3)
    {
        Console.Write("type trianble base --> ");
        sidel = int.Parse(Console.ReadLine());
        DrawTriangle(sidel);
    }
}
}

```

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

בציה 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. אפש את הסכום
2. כא עזר סכום המשקלים קטן או שווה ל-100 בצע:
2.1 קלוט מספר גלי ספרתי במשתנה num

פעולה המחזירה ערך:

```
//--- תיעוד הפעולה ---
static (רשימת פרמטרים) שם-הפעולה טיפוס-הערך-המוחזר
{
    // 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.WriteLine("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. קלט מספר בגוף המשתנה num.

2. כן / לא? אם num קטן מ-100 או num גדול מ-999: כן

2.1 קלט מספר בגוף המשתנה num

נכתוב פעולה בשם GetNum שתחזיר מספר תלת-ספרתי כנדרש:

```
static int GetNum()
{
    int num;

    Console.WriteLine("type a positive 3 digits number --> ");
    num = int.Parse(Console.ReadLine());
    while (num < 100 || num > 999)
    {
        Console.WriteLine(" Error ! ");
        Console.WriteLine("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.WriteLine("type a positive 3 digits number --> ");
        num = int.Parse(Console.ReadLine());
        while (num < 100 || num > 999)
        {
            Console.WriteLine(" Error ! ");
            Console.WriteLine("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

קצ'ה 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

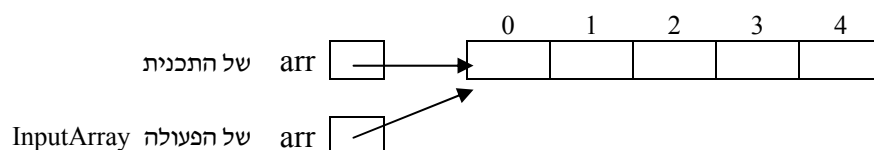
כאשר אנו שולחים מערך כפרמטר לפעולה, אנו בעצם שולחים את ההפניה למערך עצמו, ולכן, כל שינוי במערך בתוך הפעולה משפיע גם על המערך שבתכנית המזמנת את הפעולה.

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.WriteLine("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.WriteLine(" Type array size --> ");
    int n = int.Parse(Console.ReadLine());

    int [] arr = new int[n];

    for (int i = 0 ; i < arr.Length ; i++)
    {
        Console.WriteLine("type number " + i + " --> ");
        arr[i] = int.Parse(Console.ReadLine());
    }

    return arr;
}
```

אפשר להמיר קטע זה
במשפט הזימון:
InputArray(arr);

```
int [] arr = GetArray();
```

משפט הזימון לפעולה, מתוך התכנית, יהיה:

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) הגזר את `arr` כמערך של `n` איברים בגודל 80 וקלוט באיבר מספרים שלמים
- (2) קלוט `k`- מספר שלם בין 0 לגודל המערך (כולל)
- (3) גש `sum1` את סכום `k` האיברים הראשונים במערך
- (4) גש `sum2` את סכום האיברים ממקום `k` ועד סוף המערך
- (5) אם הערך של `sum1` גדול מהערך של `sum2` הדפס את `sum1`

הערה:

- יכולנו להחליף את שורות 3-5 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.WriteLine("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.WriteLine(" type array size --> ");
        int n = int.Parse(Console.ReadLine());

        int [] arr = new int[n];
        for (int i = 0 ; i < arr.Length ; i++)
        {
            Console.WriteLine("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 וכד'.

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.WriteLine("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.WriteLine("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`:

`Array.InputArray(intArr);`

כדי להציג את איברי המערך, נציין שהפעולה `PrintArray` נמצאת במחלקה `Array`. שליחת המערך `intArr` כפרמטר לפעולה, תגרום לבחירת הפעולה המקבלת מערך של `int`.