פרק 3 – מודל חישוב בסיסי

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

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

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

3.1 צעדים ראשונים: הוראת פלט, הוראת קלט ומשתנים

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

בעיה 1

מטרת הבעיה ופתרונה: הצגת משפט כפלט

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

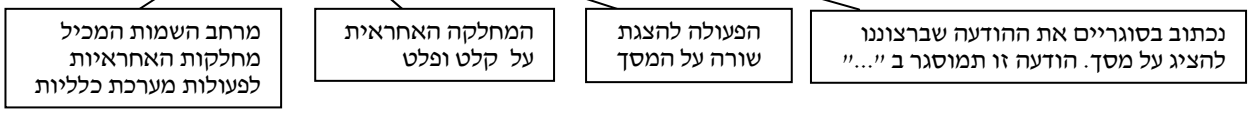
האלגוריתם לפתרון הבעיה יהיה האלגוריתם הפשוט הבא, הכולל הוראת פלט אחת:

1. *הדפס כפלט את המילים* Hello World

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

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

נבחן ממה מורכבת הוראה זו:



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

```
using System;
```

וכעת ניתן לכתוב את הוראת הפלט בצורה מקוצרת כך:

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

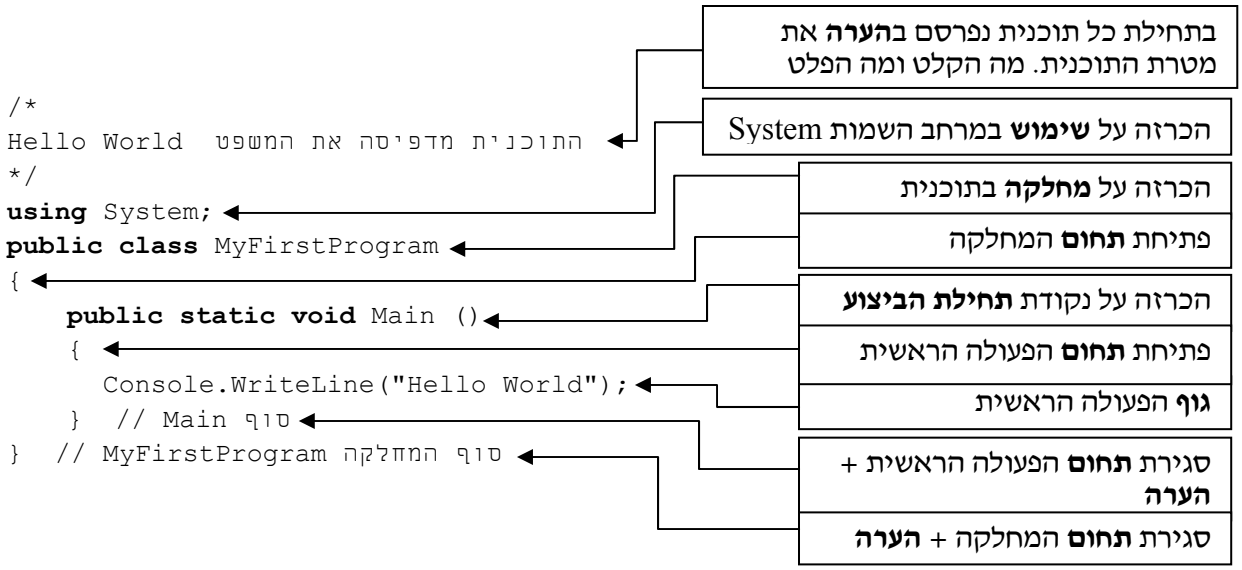
כעת ניצור מהמשפט שכתבנו תוכנית מלאה בשפת C#:

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

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

סוף פתרון בעיה 1

ננסה להבין כמה מהחלקים שהוספנו לתוכנית ונוסיף הערות להבהרה:



מחלקה – class

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

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

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

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

Main

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

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

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

בדוגמה זו, שם המחלקה הראשית הוא MyFirstProgram.

גוף הפעולה הראשית

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

כל הוראה נכתבת בשורה נפרדת המסתיימת בסימן ;.

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

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

תחום

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

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

הערה

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

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

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

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

שאלה 3.1

פתחו אלגוריתם המציג על המסך את שמכם, וישמו אותו בשפת C#.
למשל, אם השם הוא יאיר, הפלט יהיה: Yair

שאלה 3.2

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

למשל אם השם הוא יאיר הפלט יהיה:

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

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

כזכור, המחשב קולט נתונים באמצעות אמצעי קלט. אמצעי קלט נפוץ הוא לוח המקשים – המקלדת.
בפתרון הבעיה הבאה, נראה אלגוריתם הקולט נתונים ומציגם כפלט.

בעיה 2

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

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

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

חלוקה לתת-משימות

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

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

? היכן ישמור המחשב את הנתונים הנקלטים?

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

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

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

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

האלגוריתם לפתרון הבעיה ישתמש בשני המשתנים שבחרנו:

1. *קלוט שני מספרים שלמים במשתנים* num1 ו-num2

2. הדפס כפלט את ההודעה: "שני המספרים שקלטו הם:"
3. הדפס כפלט בשורה נפרדת את ערך המשתנה num1 ואת ערך המשתנה num2

יישום האלגוריתם

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

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

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

```
int num1;
```

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

למשל נוכל להצהיר על שני המשתנים כך:

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

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

הוראה 1 (קלט)

כדי ליישם את הוראה 1 עלינו ללמוד כיצד לקלוט.

קליטת מספר שלם נעשית באמצעות ההוראה:

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

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

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

לכן, את הוראה 1 ניישם במשפטי הקלט הבאים:

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

הוראה 2 (פלט ערכי המשתנים)

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

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

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

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

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

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

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

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

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

נעקוב אחר הרצת התוכנית `ReadWrite` עבור קלט כלשהו.

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

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

המחשב יציג כפלט:

```
Enter first number:
```

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

```
Enter first number: 10
```

המחשב יציג כפלט:

```
Enter second number:
```

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

```
Enter second number:  20
```

המחשב יציג כפלט:

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

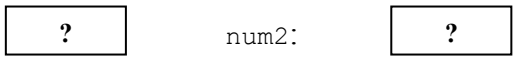
סוף התרגיל בעיה 2

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

המשפט הראשון שמתייחס למשתני התוכנית הוא משפט ההצהרה על המשתנים:

```
int num1, num2;
```

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

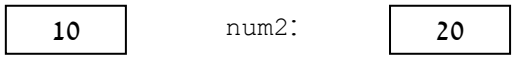


המשפטים הבאים בתוכנית הם משפטי הקלט (בצירוף הנחיות):

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

תוצאת הביצוע של משפט קלט היא שמירת ערך במשתנה.

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



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

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

אין לפעולת הפלט כל השפעה על ערכם של המשתנים.

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

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

שאלה 3.3

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

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

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

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

שאלה 3.4

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

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

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

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

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

שאלה 3.5

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

שאלה 3.6

נתונה התוכנית הבאה:

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

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

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

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

שאלה 3.7

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

שאלה 3.8

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

8 3 3 6 6 6

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

3.2 הוראת הֲשָׁמָה

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

בעיה 3

מטרת הבעיה ופתרונה: הצגת הוראת השמה.

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

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

בדיקת דוגמאות

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

שאלה 3.9

ציינו את הפלט עבור כל אחד מהקלטים הבאים:

א. 10 5

ב. 3 12

חלוקה לתת-משימות

מהן התת-משימות של האלגוריתם?

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

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

בחירת משתנים

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

בסך הכול בחרנו ארבעה משתנים מטיפוס שלם:

length – ישמור את אורך המלבן.
width – ישמור את רוחב המלבן.
area – ישמור את שטח המלבן.
perimeter – ישמור את היקף המלבן.

האלגוריתם

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

*קלוט אורך ורוחב של מלבן ב-*length *וב-*width

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

חישוב שטח מתקבל באמצעות הביטוי: length * width
חישוב היקף מתקבל באמצעות הביטוי: 2 * (width + length)
שימו ♥: ב-C# מציינים פעולת כפל באמצעות התו *

❓ היכן נשמור את תוצאות החישובים?

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

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

האלגוריתם לפתרון הבעיה יכלול שתי הוראות השמה ויהיה:

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

יישום האלגוריתם

כעת ניגש ליישום האלגוריתם בתוכנית C#.

יישום הוראות ההשמה יהיה על ידי שני המשפטים הבאים:

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

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

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

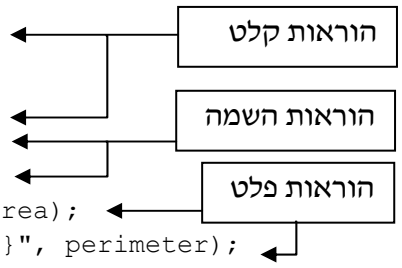
ביצוע המשפט השני, יביא לחישוב הסכום של ערכו של `width` בערכו של `length` ולהכפלתו ב-2, והשמת התוצאה במשתנה `perimeter`.

הנה יישום האלגוריתם כולו בשפת C#:

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

נשלים את משפטי התוכנית לתוכנית מלאה:

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



מעקב

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

בתחילת ההרצה ערכי כל המשתנים אינם ידועים.

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

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

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

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

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

בעקבות ביצוע שני משפטי הפלט האחרונים יתקבל הפלט:

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

סוף פתרון בעיה 3

בפתרון בעיה 3 הכרנו את פעולת ההשמה:

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

דוגמאות:

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

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

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

שאלה 3.10

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

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

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

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

שאלה 3.11

כתבו משפטי השמה לביצוע הפעולות הבאות:

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

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

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

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

שאלה 3.12

נתון קטע תוכנית ובו המשפטים הבאים:

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

תנו שתי דוגמאות קלט שונות שהפלט עבורן הוא 5.

שאלה 3.13

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

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

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

שאלה 3.14

נתונים משפטי התוכנית הבאים:

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

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

נניח שהקלט במהלך ההרצה הוא: 3 2 3.

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

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

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

שאלה 3.15

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

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

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

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

מתבצעת פעולת השמה של המספר הנקלט לתוך המשתנה length.

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

אתחול של משתנה משמעותו מתן ערך התחלתי למשתנה.

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

שפת C# מאפשרת לנו לשלב הצהרה והשמה באופן הזה:

```
int x = 3;
```

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

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

גם הוראה כזאת ניתן לכתוב בשפת C#:

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

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

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

בהמשך הפרק נשתמש לעתים במונחים ערך התחלתי ומצב התחלתי:

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

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

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

3.3 טבלת מעקב

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

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

תרגיל 4

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

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

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

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

השתמשו באלגוריתם במשתנה אחד בלבד!

בדיקת דוגמאות

שאלה 3.16

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

חלוקה לתת-משימות

נחלק לתת-משימות באופן הבא:

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

בחירת משתנים

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

האלגוריתם

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

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

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

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

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

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

הנה האלגוריתם לפתרון הבעיה:

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

יישום האלגוריתם

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

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

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

מעקב

נעקוב אחר מהלך ביצוע התוכנית עבור דוגמת הקלט 5:

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

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

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

הדו-שיח בין המחשב למשתמש יהיה:

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

סוף פתרון בעיה 4

בפתרון בעיה 4 ראינו נקודה חשובה לגבי הוראות השמה:

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

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

שאלה 3.17

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

א. `a = 1;`

ב. `a = a + 1;`

ג. `a = 2 * a + 3;`

ד. `a = 2 * a + (a - 3);`

ה. `a = b;`

ו. `a = a * b;`

ז. `a = a + a * b;`

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

שאלה 3.18

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

א. הכפלת ערכו של המשתנה `a` ב-2.

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

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

שאלה 3.19

נתון קטע התוכנית הבא:

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

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

שאלה 3.20

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

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

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

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

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

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

מבנה אפשרי לטבלת מעקב:

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

♦ שורות הטבלה:

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

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

שאלה 3.21

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

שאלה 3.22

נתון קטע התוכנית הבא:

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

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

מלאו את טבלת המעקב עבור קטע התוכנית הנתון:

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

שאלה 3.23

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

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

```
6.  sum = sum + sum;
7.  sum = sum + sum;
8.  Console.WriteLine(sum);
```

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

שאלה 3.24

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

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

3.4 החלפה בין ערכי משתנים

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

בעיה 5

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

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

הנה הצעה לפתרון:

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

ואחרי יישום כקטע תוכנית מחשב:

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

האם קטע זה משיג את המטרה?

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

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

לא השגנו את המטרה!

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

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

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

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

האלגוריתם לפתרון הבעיה יהיה:

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

ואחרי יישומו, נקבל את קטע התוכנית הבא:

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

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

סוף פתרון בעיה 5

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

משתנה עזר הוא משתנה שנועד לסייע בביצוע משימה כלשהי.

שאלה 3.25

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

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

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

שאלה 3.26

נתון קטע התוכנית הבא:

```
1. Console.Write("Enter number: ");
2. a = int.Parse(Console.ReadLine());
3. Console.Write("Enter number: ");
4. b = int.Parse(Console.ReadLine());
5. Console.Write("Enter number: ");
6. c = int.Parse(Console.ReadLine());
7. temp = a;
8. a = b;
9. b = c;
10.c = temp;
11.Console.WriteLine(a);
12.Console.WriteLine(b);
13.Console.WriteLine(c);
```

א. בנו טבלת מעקב אחר מהלך הביצוע של קטע התוכנית עבור הקלט 3 2 1.
ב. הביאו דוגמת קלט שהפלט המתקבל עבורה הוא 3 2 1.
ג. מהי מטרת קטע התוכנית?
ד. האם ניתן להשיג את מטרת קטע התוכנית ללא משפטי השמה כלל? אם כן, כיצד?

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

שאלה 3.27

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

```
1. Console.Write("Enter number: ");
2. a = int.Parse(Console.ReadLine());
3. Console.Write("Enter number: ");
4. b = int.Parse(Console.ReadLine());
5. Console.Write("Enter number: ");
6. c = int.Parse(Console.ReadLine());
7. a = c;
8. c = a;
9. Console.Write(a);
10.Console.Write(b);
11.Console.Write(c);
```

א. הביאו שתי דוגמאות קלט שונות שעבור כל אחת מהן יוצג הפלט הדרוש.
ב. הביאו דוגמת קלט שעבורה **לא** יוצג הפלט הדרוש.
ג. תקנו את הקטע בלי לשנות את משפטי הקלט והפלט, כלומר, השלימו רק את השורות החסרות בקטע התוכנית שלהלן:

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

     השלימו כאן

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

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

3.5 טיפוסים

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

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

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

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

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

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

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

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

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

פעיה 6

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

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

בדיקת דוגמאות

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

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

חלוקה לתת-משימות

נבצע את החלוקה הבאה לתת-משימות:

1. קליטת ארבעה מספרים שלמים.

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

בחירת משתנים

אילו טיפוסים מתאימים לבעיה זו?

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

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

ובסך הכול נקבל:

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

האלגוריתם לפתרון הבעיה יהיה:

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

יישום האלגוריתם

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

```
double average;
```

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

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

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

את הוראה 3 היינו רוצים ליישם במשפט השמה כזה:

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

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

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

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

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

הנה התוכנית השלמה:

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

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

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

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

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

סוף פתרון בעיה 6

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

מרכיב חשוב בהגדרת הייעוד של משתנה הוא הגדרת סוג הערכים שיישמרו במשתנה. סוג ערכים זה נקבע באמצעות טיפוס המשתנה. חשוב להתאים את טיפוס המשתנה לתפקידו.

משתנה שומר ערכים מטיפוס (סוג) אחד בלבד.

יש להצהיר בנפרד על משתנים מטיפוסים שונים.

בשפת C# מצהירים על משתנה מטיפוס שלם באמצעות המילה **int** ועל משתנה מטיפוס ממשי באמצעות המילה **double**.

במשפט השמה ניתן לשים במשתנה מטיפוס שלם רק ערך מטיפוס שלם, ואילו במשתנה מטיפוס ממשי ניתן לשים גם ערך מטיפוס שלם וגם ערך מטיפוס ממשי.

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

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

שאלה 3.28

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

שאלה 3.29

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

למשל, עבור הקלט 2.84 1.6 הפלט הוא: 0.71 0.4

1.11

שימו ♥ לבחירת טיפוסי המשתנים!

שאלה 3.30

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

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

כדאי לדעת – סיבות לאבחנה בין שלם לממשי:

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

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

3.6 קבועים

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

סיכום

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

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

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

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

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

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

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

מתן ערך התחלתי למשתנה נקרא **אתחול**.

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

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

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

פתרון בעיה אלגוריתמית נעשה בשלבים:

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

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

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

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

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

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

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

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

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

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

סיכום מרכיבי שפת C# שנלמדו בפרק 3

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

מבנה תוכנית בשפת C#

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

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

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

הערות

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

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

שמות

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

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

ערכים ונתונים בתוכנית C#

טיפוסים

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

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

משתנים

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

```
int num;
```

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

```
double num1, num2;
```

קבועים

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

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

הוראות ביצוע של תוכנית בשפת C#

קלט

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

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

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

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

פלט

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

למשל:

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

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

```
using System;
```

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

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

השמה

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

```
משתנה = ביטוי;
```

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

שאלות נוספות

שאלות נוספות לסעיף 3.1

1. כתבו תוכנית להדפסת האות L מכוכביות באופן הבא:

```
*
*
* * *
```

2. נתון קטע התוכנית הבא:

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

נניח שנתוני הקלט שהוקלדו הם 10 8:

א. מה יהיו ערכי המשתנים לאחר ביצוע משפטי הקלט?

ב. מה יהיה הפלט?

3. נתון קטע התוכנית הבא:

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

תנו דוגמת קלט שהפלט עבורה הוא:

```
5  9
9  5  9
```

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

שאלות נוספות לסעיף 3.2

1. כתבו משפטי השמה לביצוע הפעולות הבאות:

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

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

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

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

שאלות נוספות לסעיף 3.3

1. נתון משפט ההשמה הבא: `a = a + b + c + d;`

א. כתבו במקום משפט השמה זה סדרה של משפטי השמה אשר הביטוי בצד ימין של כל אחד מהם כולל סימן חיבור אחד בלבד. בסיום ביצוע סדרת המשפטים ערכו של a יהיה זהה לערכו המתקבל לאחר ביצוע המשפט הנתון. בצעו את המשימה ללא הוספת משתנים.

ב. בנו טבלת מעקב אחר מהלך הביצוע של סדרת משפטי ההשמה שכתבתם בסעיף א עבור הערכים ההתחלתיים 1, 5, 2 ו-4 במשתנים a, b, c ו-d בהתאמה.

2. כתבו משפטי השמה לביצוע הפעולות הבאות:

א. הקטנת ערכו של a ב-5.

ב. הגדלת ערכו של b פי 3.

ג. הקטנת ערכו של המשתנה a בערכו של המשתנה b.

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

א.
`n = m + 5;`
`n = n - 2;`
	ב.
	`a = (a + 2)* 3;`
	`a = a * 4 - 9;`

ג.
`v = v + w;`
`v = v * 5;`
	ד.
	`x = x - 2;`
	`x = x - 3;`

שאלות נוספות לסעיף 3.4

1. נתון קטע התוכנית הבא:

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

א. מהו פלט קטע התוכנית עבור הקלט 5 13? היעזרו בטבלת מעקב למציאת הפלט.

ב. נסחו את הבעיה האלגוריתמית שקטע תוכנית זה פותר.

ג. כתבו קטע תוכנית אחר לפתרון הבעיה.

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

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

א. תנו דוגמה של ערכים התחלתיים עבור המשתנים אשר המטרה לא מושגת עבורם.

ב. מה מתבצע בסדרת המשפטים הנתונה? האם היא גורמת ל"אובדן" ערכי משתנים?

ג. כתבו סדרת משפטים שעבורה תושג המטרה.

שאלות נוספות לסעיף 3.5

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

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

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

נבחר את המשתנה הבא מטיפוס שלם:

num – ישמור את מספר המטריות המבוקש

ואת המשתנים הבאים מטיפוס ממשי:

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

פותח אלגוריתם שהוראותיו מיושמות במשפטי התוכנית הבאים:

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

השלימו את משפטי התוכנית.

שאלות מסכמות לפרק 3

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

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

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

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

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

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

4. נתון קטע התוכנית הבא:

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

א. מהו פלט קטע התוכנית עבור הקלט 2 3? היעזרו בטבלת מעקב כדי לענות על השאלה.
ב. תנו דוגמת קלט אשר הפלט עבורה הוא 2 1 3.
ג. נסחו במילים את היחס שמוגדר בקטע התוכנית בין הפלט לקלט.

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

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

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

קטע התוכנית שגוי.

א. הסבירו מדוע הקטע שגוי.

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

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

תבניות – פרק 3

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

החלפת ערכים בין שני משתנים

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

היפוך סדר האיברים בסדרה

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

ממוצע של סדרת מספרים

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

הזזה מעגלית בסידרה

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

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