מבנה התכנית

התכנית הראשונה שלי

מחלקה מגדירה עצם. עצמים מתקשרים ביניהם באמצעות תכנית מַנְחָה. בפרקים הבאים נכיר את מבנה התכנית המנחה ואת הוראות השפה, ובהמשך נלמד לשלב גם עצמים.

לפניך תכנית בשפת C#. התכנית קולטת את גילו של אדם ומדפיסה את גילו בעוד שנה. לבסוף מדפיסה ברכה להמשך יום נעים:

```
1.  /*~~~~~~~~~~~~~~~~~~~~~~~~*
2.   *          תכנית דוגמא          *
3.   *~~~~~~~~~~~~~~~~~~~~~~~~*/
4.  using System;
5.  namespace DefaultNamespace
6.  {
7.      class MainClass
8.      {
9.          public static void Main (string[] args)
10.         {
11.             double age;
12.             string name;
13.             Console.Write ("What is your name? → ");
14.             name = Console.ReadLine ();
15.             Console.Write ("How old are you? → ");
16.             age = double.Parse(Console.ReadLine());
17.             age = age + 1;
18.             Console.WriteLine ("Hello " + name);
19.             Console.WriteLine ("Next year you will be {0}", age);
20.             Console.WriteLine ("Have a nice day   :o)");
21.         }
22.     }
23. }
```

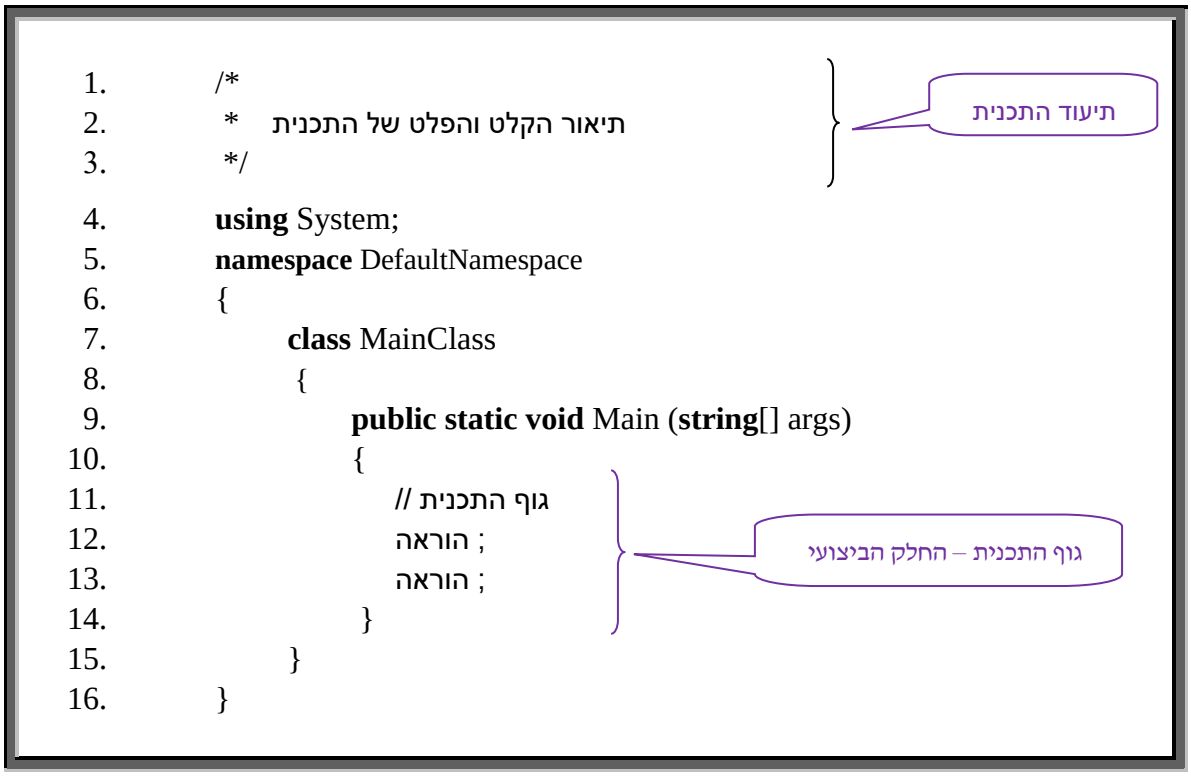
בתכנית מופיעים מאפיינים רבים של שפת C#, כגון שימוש במשתנים, הוראות **קלט**, **עיבוד** הנתונים שנקלטו, הוראות **פלט** והוראת השמה.

יתכן וקוד התכנית נראה לא ברור בשלב זה, אך בהמשך נכיר את כל המאפיינים שהוזכרו ונלמד לכתוב תכניות משלנו.

בסיומו של פרק זה, נהיה מסוגלים לכתוב תכניות כדוגמת תכנית זו.

המבנה הכללי של תכנית בשפת C#

במהלך לימוד פרקי הספר השונים, יובהרו כל מרכיבי התכנית.




```
1.      /*
2.       *   תיאור הקלט והפלט של התכנית
3.       */
4.      using System;
5.      namespace DefaultNamespace
6.      {
7.          class MainClass
8.          {
9.              public static void Main (string[] args)
10.             {
11.                 // גוף התכנית
12.                 הוראה ;
13.                 הוראה ;
14.             }
15.         }
16.     }
```

חלקי התוכנית

שורות 1 – 3: תיעוד התכנית.

שורות אלו הן **הערות**. ההערות אינן הוראות לביצוע. הן משמשות להוספת מידע, הסבר והנחייה לאדם הקורא את התכנית. **המהדר** מתעלם מהן.

מהדר - compiler: תכנית מחשב המתרגמת את התכנית שכתבנו בשפת C# לשפת מכונה - שפה שמהמחשב מסוגל להבין. לכל שפת תכנות יש מהדר משלה.

שורות 4, 5, 6 ו- 16: שימוש ב- System ויצירת מרחב שמות.

שורה 4 מכילה הודעה שהתכנית מתכוונת לעשות שימוש במרחב שמות הקרוי System.

מרחב שמות הוא אוסף של מחלקות אשר נאספו יחד וקיבלו שם משותף. על מנת שנוכל להשתמש בהן, יש לציין שם זה. ניתן להתייחס למחלקות אלה כאל חבילות תכנה המספקות שירותים שונים למתכנת. במקום שהמתכנת יצטרך לכתוב בעצמו את קוד תכנית השרות (הפעולה), יודע המהדר לחפש את הפעולה במרחב השמות.

בתוך מרחב השמות System מוגדרת מחלקה מיוחדת הקרויה Console המספקת לנו שרותי קלט ופלט, ולכן בכל תכנית שנכתוב נצטרך להוסיף את השורה **using System ;** .

בשורה 5 מוגדר מרחב השמות. בתכנית הלימודים שלנו נוכל להתעלם משורה זו, ולוותר עליה, או להשאיר את השם שנקבע אוטומטית על ידי העורך בעת יצירת תכנית חדשה.

אם בחרנו להשאיר את שורת ה- namespace, הרי שהסוגריים המסולסלים שבשורות 6 ו- 16 קובעים **בלוק** (קטע תכנית) שבתוכו מוגדרות הפעולות של מרחב השמות. אם נבחר שלא לרשום את שורה 6, נסיר גם את סימני הסוגריים המסולסלים שבשורות אלו.

שורות 7 , 8 ו- 15: הכרזה על מחלקה.

שפת C# היא שפה מכוונת עצמים. שפות מכוונות עצמים משתמשות במחלקות כדי להכריז

על **עצמים** (**אובייקטים**). התכנית שלנו הינה אובייקט. כדי ליצור אובייקט יש ליצור מחלקה שתגדיר אותו. השם MainClass אינו שם מחייב והוא ניתן לשינוי על ידי המתכנת.

הסוגריים המסולסלים בשורות 8 ו- 15 קובעים את גבולות המחלקה.

שורה 9: כותרת הפעולה הראשית Main ()

המחשב צריך לדעת מהיכן להתחיל את ביצוע התכנית. התכנית ב- C# מתחילה תמיד מהפעולה הראשית Main. פעולה הינה קטע קוד שנועד לבצע משימה מוגדרת.

public static void Main (**string** [] args)

ניתן להשמיט את הקידומת public (זו ברירת המחדל). כמו כן ניתן להשמיט את מה שבתוך הסוגריים (אבל לא את הסוגריים). המינימום הנדרש הוא:

static void Main( )

שים ♥: המילה **Main** מתחילה באות גדולה.

הסוגריים המסולסלים בשורות 10 ו- 14 קובעים את גבולות הפעולה Main.

שורות 10 - 14: החלק הביצועי - התכנית.

החלק הביצועי תחום בתוך סוגריים מסולסלים הפותחים את התכנית בשורה 10 ומסיימים אותה בשורה 14.

הערות ותיעוד מתכנית

על מנת שתכנית תהיה ברורה לקורא התכנית, היא חייבת להיות מוסברת (מתועדת) היטב, מעוצבת וכתובה לפי מוסכמות.

תיעוד (documentation)

נושא התיעוד הינו נושא מרכזי וחשוב בפיתוח תוכנות מחשב. בתהליך פתוח התכנית משתתפים מתכנתים רבים. תיעוד התכנית מסייע למתכנת להבין את הרעיון שעומד מאחורי הקוד שכתב מתכנת אחר, או במקרה של מפתח יחיד, התיעוד מסייע להיזכר בקו המחשבה שליווה אותו במהלך פיתוח התכנית כשירצה לתחזק אותה או לשדרגה בשלב מאוחר יותר.

במשך הזמן התפתחו **מוסכמות** (conventions) של כללים בכתיבת תוכנה. אמנם המחשב אינו זקוק לכללי כתיבה - התוכנית תרוץ גם אם לא נעבוד לפיהם, אך הם בהחלט מחייבים בקהילת המתכנתים ובתי התוכנה. כללים אלו תורמים **לקריאות** (readability) התוכנית ומסייעים בתחזוקה השוטפת שלה. סביבות העבודה החדשות, הנהוגות כיום, כוללות בתוכן הכנה לתעוד בצורות שונות.

הערות

ההערות הינן כלי תחבירי של השפה באמצעותו ירשם התיעוד בתכנית.

הערות הינן קטע בתכנית שאינו חלק מהוראות השפה ונועד להבהיר למתכנתים אחרים את התכנית. להערות סימון מיוחד שנועד להבהיר למהדר (קומפיילר) שלא מדובר בהוראות השפה ויש להתעלם מקטע זה. שים לב שבסביבת העבודה, מסומנות ההערות בצבע מיוחד.

נבחין בין שני סוגים של הערות:

- **הערת קטע**: /* הערת קטע */ הערה היכולה להשתרע על פני מספר שורות.

הערת קטע נפתחת בסימן /* ומסתיימת בסימן */ (ההערה נתחמת בתוך סימני הלוכסן וכוכבית צמודים ללא תו רווח ביניהם). גוף ההערה יכול להיכתב בשפה חופשית, כולל בשפה העברית. שים לב שהעורך מוסיף כוכבית בתחילת כל שורה. כוכבית זו הינה רשות ולא חובה.

- **הערת שורה**: הערה מסוג זה מתחילה בסימן // ומסתיימת בסוף השורה.

תיעוד התוכנית

השורות הראשונות בתוכנית הן כותרת התוכנית. שורות אלו כוללות מידע כללי על התוכנית, הרשום בצורה של הערות. לפי דרישת תכנית הלימודים, חובה לרשום תיעוד בראש כל תכנית. התיעוד יכול להירשם בשפה חופשית המתארת את קלט התכנית ומטרתה או בצורה פורמאלית כמתואר בדוגמא שלהלן. נהוג לרשום את המידע הבא:

- קלט - מאפייני הקלט של התוכנית.
- פלט - פלט של התוכנית.

תיעוד התכנית יסומן כהערת קטע: דוגמא לתעוד:

```
/*
 *                                  קלט: a  צלע הריבוע
 *                                  פלט: s  שטח הריבוע
 */
```

מילים שמורות

בכל שפת תכנות קיימות **מילים שמורות**. מילה שמורה היא מילה שהשימוש בה והמובן שלה שמורים לשימוש בלעדי ע"י השפה ולא ניתן לשנות את משמעותה. מילים שמורות מסומנות בצבע מיוחד בתכנית. בספר זה המילים השמורות מופיעות בקוד התכנית בגופן מודגש בצבע כחול.

עיצוב כללי

- המספור שמשמאל לשורות אינו שייך לתוכנית, אך מסייע להתמצאות בתכנית. מרבית סביבות העבודה מאפשרות להציג מספור זה. הוספת שורות בין שורות התכנית תגרום לעדכון אוטומטי של המספור.
- התוכנית לביצוע תהיה מורכבת מהוראות השפה ותיכתב בתוך הסוגריים המסולסלים תוך שמירה על כללי כתיבה מקובלים כמו **הזחות** (indent) - הקובעות את מיקום תחילת השורה ביחס לשורה שלפניה. חשוב להקפיד על הזחות ברורות.
  בדרך כלל קובע העורך (editor) של סביבת העבודה את ההזחות בצורה אוטומטית.
  הזחות תורמות לקריאות התוכנית.
- יש עורכים הקובעים אוטומטית את מיקומם של הסוגרים המסולסלים של תחילת הבלוק:

```
public class ClassName {          public class ClassName
        …                         {
}                                         …
                                  }
```

  שתי צורות הכתיבה נכונות, ושתיהן תופענה בספר.
- כל הוראה בשפת C# חייבת להסתיים בסימן נקודה-פסיק (**;**).
- **תווי רווח** (whitespace): תווי רווח הינם רווחים בין המילים, טאבים (tab) וירידת שורה (מקש Enter). המהדר מתעלם מתווי רווח. המשמעות - בכל מקום בו מותר תו רווח אחד, מותר לשים הרבה תווי רווח ואף לעבור לשורה חדשה. ריווח התכנית תורם לנוחות הקריאה.
  שים ♥: למרות האמור לעיל, המהדר אינו מתעלם מתווי רווח בתוך מחרוזת טקסט המופיעה בין גרשיים. הקשה על מקש Enter בתוך מחרוזת טקסט יגרור הודעת שגיאה.

- שפת C# הינה שפה **תלוית ראשיות** (case sensitive), כלומר קיימת הבחנה בין אותיות גדולות (upper case) ואותיות קטנות (lower case). המילים Main ו- main נחשבות למילים שונות בשפה. חובה להקפיד על גודל אות מתאים.

משתנים

תכניות פועלות על מידע. המידע עליו פועלת התכנית נמצא בזיכרון המחשב בזמן הפעולה.

משתנה (variable) הוא תא זיכרון אשר ניתן לשמור בו ערך ולקרוא (לאַחזֵר - Retrieve) את הערך השמור בו במהלך ביצוע התכנית. ערך זה נקרא **ערך המשתנה**. הכינוי "משתנה" נובע מכך שבמהלך ביצוע התכנית ניתן לשנות את ערכו שוב ושוב.

בזמן פיתוח אלגוריתם/תכנית בוחר המתכנת במשתנים אשר ישמשו לשמירת נתוני קלט ולשמירת תוצאות חישוב שיתקבלו במהלך ביצוע התכנית. כדי להשתמש במשתנים אלו בתוכנית, יש **להכריז** (declare) על המשתנים איתם נעבוד. פניה למשתנה נעשית באמצעות שם הניתן לו על ידי המתכנת. שם זה הוא **שם המשתנה**.

משתנה יכול **לקבל ערך** באחת משתי הצורות הבאות:

- משפט השמה. המתכנת קובע את ערך המשתנה בזמן כתיבת התכנית.
- משפט קלט. התכנית קולטת ערך מהמקלדת במהלך הביצוע. הקלט מוזן על ידי המשתמש בתכנית.

שימוש במשתנה (בערך המשתנה):

- ניתן לבקש להדפיס את תוכן המשתנה בהוראת פלט.
- ניתן להתייחס לתוכן המשתנה לביצוע פעולות שונות.
- ניתן לשמור במשתנים את התוצאות של חישובים שונים שבוצעו במהלך התכנית.

תכונות כלליות של משתנים

- ניתן להכריז על משתנים בכל מקום בתכנית, ובלבד שנכריז עליהם לפני השימוש הראשון בהם. אף על פי כן, ולמען הסדר הטוב, נשתדל לרכז את כל שורות ההכרזה בתחילת התכנית.
- במשתנה נשמר הערך האחרון שהושם בתוכו. המשתנה אינו "זוכר" מה היה בו לפני כן.
- מתן ערך התחלתי למשתנה נקרא **אתחול**.
- לכל משתנה יש ערך. הערך של משתנה שעדיין לא ביצענו לו השמה אינו ידוע. מנקודת המבט של המתכנת ערך המשתנה אינו מוגדר למרות שתמיד יש בו ערך כלשהו. במהדרים מסוימים, אי אתחול משתנה יגרור הודעת שגיאה.

יש לוודא שכל משתנה מקבל ערך התחלתי לפני שמשתמשים בו.

הכרזה על משתנים בתכנית

ככלל, משתנה חייב להיות מוגדר לפני השימוש הראשוני בו. נהוג להגדיר את משתני הפעולה בתחילת הפעולה לפני ההוראות לביצוע, באופן הבא:

```
1.    /*          (תיעוד התכנית)          */
2.    using System;
3.    class MainClass
4.     {
5.          public static void Main (string [] args)
6.          {
7.               //---  הצהרה על משתני התכנית  ---
8.               טיפוס-משתנה משתנה1 ;
9.               טיפוס-משתנה משתנה2, משתנה3, משתנה4 ;
10.              //--- גוף התכנית – החלק הביצועי ---
11.              הוראה ;
12.              הוראה ;
13.          }
14.   }
```

- חובה להצהיר על כל משתני התכנית.
- מספר שורות ההגדרה אינו מוגבל.
- אין לכלול משתנים מטיפוסים שונים באותה שורת הגדרה.
- נהוג להכריז על משתנים בעלי תפקידים דומים באותה שורה, ולספק תיאור של תפקיד המשתנה כהערה בקצה השורה.
- לכל משתנה יש **טווח הכרה** (scope). משתנה מוכר וקיים בתוך הבלוק שבו הוא נוצר. כלומר - משתנה שהוגדר בפעולה אחת, אינו מוכר בפעולה אחרת.

שם משתנה

שם משתנה חוקי יכול להיות מורכב מסדרה של אותיות באנגלית, ספרות וקו תחתי _ , וחייב להתחיל באות. אסור לכלול תווי רווח וסימנים אחרים בשם המשתנה. שם משתנה אינו יכול להכיל אותיות עבריות.

שפת C# היא שפה תלוית רישיות (case sensitive), כלומר יש חשיבות לגודל אות. מקובל שאם שם המשתנה מורכב ממספר מילים, תתחיל כל מילה, החל מהמילה השניה, באות גדולה. לדוגמא: myNumber , numberOfStudents , newNum. בגלל הרגישות לגודל אות, הרי שהשמות: newnum , NewNum , newNum מתייחסים לשלושה משתנים שונים.

שמות חוקיים: start, old, firstName, number, size37 .

שמות לא חוקיים: family name , name.2 , first-second , -number , a#5 , why? (מדוע ?)

שים ♥: **מילים שמורות אינן יכולות לשמש כשמות של משתנים**.

מוסכם ששמות משתנים יהיו תמיד שמות משמעותיים – המתארים את השימוש במשתנה.

דוגמאות: משתנה האמור להכיל שם של תלמיד ייקרא - **name** או **studentName.**
משתנה האמור להכיל גיל ייקרא – **gil** או **age**.
משתנה האמור להכיל את גודל החדר ייקרא - **size** או **roomSize** .

טיפוסי משתנים

לכל משתנה יש להגדיר את **סוג הנתונים** שלו (data type). סוג הנתונים של המשתנה קובע איזה ערכים יוכל המשתנה להכיל ואלו פעולות ניתן לבצע על המשתנה.

שפת C# מספקת מספר סוגים של **משתנים בסיסיים** (משתנים פרימיטיבים), שבהם ניתן לאחסן מידע מספרי, מידע טקסט (מחרוזת טקסט) או מידע לוגי ("אמת" או "שקר"). כל סוג משתנה הוא בעל דרישות אחסון שונות בזיכרון המחשב ונבדל בסוג המידע שניתן לאחסן בתוכו ובסוג הפעולות אותו ניתן לבצע על מידע זה. לכל משתנה בשפה יש מחלקה המגדירה אותו וקובעת את ה**תכונות** (תחום הערכים שהוא יכול לקבל) וה**פעולות** אותן ניתן לבצע במשתנים אלו.

בתכנית הלימודים **יסודות** נשתמש רק בחלק מטיפוסי הנתונים הקיימים בשפה.

int – משתנה מטיפוס מספרי שלם (integer)

int הוא משתנה מטיפוס מספרי שלם. התופס 4 בתים (32 סיביות) בזיכרון.
תחום הערכים שהוא יכול לקבל הוא מ- -2^{31} ועד $+2^{31}-1$, כלומר מ- -2,147,483,648 ועד +2,147,483,647. ניתן להוסיף סימני +/- לנתון המספרי.

דוגמאות: -2 , 0 , 1996 , -641 , +17 , 9

double – משתנים מטיפוס מספר ממשי (floating point)

לטיפול ואחסון מספרים ממשיים (מספרים בעלי נקודה עשרונית), קיימים שני טיפוסי משתנים עיקריים: float ו- double.

ברירת המחדל של C# הוא double המיוצג על ידי 8 בתים (64 סיביות). מידת הדיוק שלו הוא 15 עד 16 ספרות אחרי הנקודה העשרונית. (אם נרצה להשתמש במספרים מטיפוס float (בעל 7 ספרות דיוק), נוסיף את האות F לקבוע המספרי. לדוגמא: 3.14F ישמר במשתנה מסוג float ולא double לפי ברירת המחדל.)

דוגמאות למספרים ממשיים: 3.2456 , -583.12 , 4.0 , 0.0091

char - משתנה מטיפוס תווי (character)

תווים הם אותיות, ספרות (להבדיל ממספר) וסימני טקסט שונים. תו מיוצג בשפה על ידי תחימתו בין גרש שמאלי לגרש ימני: 'a' , '@' , '3' .

המחשב אינו מזהה תווים. הוא מסוגל לזהות רק מספרים, לכן כל התווים מאוחסנים כערכים מספריים. כדי להבטיח שכל יצרני המחשבים בעולם ישתמשו באותם ערכים, הוגדר תקן שנקרא בשם Unicode שבו כל תו וסימן מיוצגים על ידי מספר שלם המתאים רק לו.

משתנה מסוג char יכול להכיל ערך מספרי המתאים לתו אחד ויחיד. גודל הזיכרון שתופס משתנה זה הוא 2 בתים (16 סיביות) ומכאן שטבלת Unicode מכילה ייצוג של 65,536 ($= 2^{16}$) תווים שונים.

string – מחרוזת (string)

מחרוזת בשפת C# הינה אובייקט מיוחד שנועד לעבודה עם טקסטים. מחרוזת מיוצגת בשפה על ידי תחימתה במירכאות כפולות (גרשיים): "Hello, have a nice day ☺" .
בשלב זה נעשה הכרות ראשונית עם המחרוזת. כדי שנוכל לכתוב תוכניות ברורות יותר. נושא המחרוזות נידון בהרחבה בפרק 12.

bool – משתנה מטיפוס בוליאני

טיפוס הנתונים הבוליאני משמש לבדיקת נכונותם של פסוקים. בעזרתו נוכל לדעת האם משהו מתקיים או לא מתקיים, אמת או שקר, נכון או לא נכון וכדי. משתנה מטיפוס בוליאני תופס בית אחד (8 סיביות) בזיכרון, ויכול לקבל ערך **אמת** - **true** או **שקר** - **false** בלבד.

דוגמא להצהרה על משתנים:

```
1.   /*          (תעוד התכנית)                */
2.   using System;
3.   class MainClass
4.   {
5.       public static void Main (string [] args)
6.       {
7.           int  dd, mm, yy;        // משתני התאריך: יום, חודש, שנה
8.           double  age;            // גיל
9.           string name;            // שם
10.          //--- גוף התכנית – החלק הביצועי ---
11.          הוראה ;
12.          הוראה ;
13.      }
14.  }
```

בדוגמא שלעיל שורה 7 מכריזה שבתוכנית נשתמש במשתנים ששמותיהם dd , mm ו- yy. הקידומת int פרושה שהמשתנים הם מ**טיפוס** (סוג) של **מספר שלם** (integer). בשורה 8 הוצהר על משתנה בשם age מטיפוס double, טיפוס משתנה המכיל **מספר ממשי** - מספר הכולל חלק עשרוני. בשורה 9 הוצהר על משתנה בשם name מטיפוס **מחרוזת** (טיפוס המכיל אותיות ומילים).

קבועים

לעיתים נוצר צורך להשתמש בנתון מסויים שיישאר קבוע בכל התכנית, ולא ניתן יהיה לשנותו. הנתון יכול להיות מחרוזת, מספר שלם או מספר ממשי. הכרזה על קבוע נעשית באמצעות המילה השמורה **const**, בחלק שבו מצהירים על המשתנים. ההכרזה קובעת ערך סופי לקבוע ואין אפשרות לשנותו בהמשך התכנית.

const <טיפוס-נתונים> <שם-הקבוע> = <ערך קבוע> ;

כדי להקל על זיהוי קבועים, נהוג לרשום את שמותיהם באותיות גדולות (upper case). אם שם הקבוע מכיל יותר ממילה אחת, נשתמש בקו תחתי כמפריד בין המילים. לדוגמא: MAX_VALUE.
הרחבה על קבועים ראה בפרק 7 - לולאות.

טיפול במשתנים – הוראות פלט השמה וקלט

כדי ליצור אינטראקציה בין המשתמש בתכנית והתכנית עצמה משתמשים בהוראות קלט ופלט.
בעוד שהוראות הקלט משמשות לקבלה והכנסה של נתונים מן המשתמש לתוך המשתנים, הרי שהוראות הפלט נועדו להציג את תוכן תאי הנתונים.

הוראות הקלט והפלט בשפת C# נעשות דרך המחלקה **Console** המציעה אוסף של פעולות המטפלות בקלט מידע מהמקלדת ופלט לחלון ה- Console של מערכת ההפעלה, הלא הוא המסך.

הפעלת פעולה של מחלקה נעשית בדרך הבאה: **שם-המחלקה.שם-הפעולה( )**
ClassName.MethodName ( )

הוראות פלט

שליחת מידע לאמצעי הפלט הסטנדרטי (המסך):

Console.Write (המידע שיוצג על המסך) ;
Console.WriteLine (המידע שיוצג על המסך) ;

המידע המועבר לפעולה יכול להיות:

- קבוע מספרי: Console.WriteLine (5);
- קבוע מחרוזתי: Console.WriteLine ("Have a nice day");
- תוכן של משתנה: Console.WriteLine (num);
- צירוף של מחרוזות ומשתנים: Console.WriteLine ("There were " + count + " items");

צירוף של פרטי מידע נקרא **שרשור**.

הוראת Console.**Write** מדפיסה את המידע ומשאירה את הסמן במקום בו הסתיימה ההדפסה.
ההוראה Console.**WriteLine** מדפיסה את המידע ומעבירה את הסמן לתחילת השורה הבאה.

מכאן שקטע ההוראות הבא:

```
Console.Write ("My lucky number ");
Console.Write ("is : ");
Console.Write (num);
Console.WriteLine();
```

שקול להוראה:

```
Console.WriteLine ("My lucky number is : " + num);
```

C# מאפשרת להציג את הפלט תוך שימוש **בסמנים** (**Markers**) (האפשרות השלישית מגרסת 2017 והלאה):

Console. Write **(משתנה1 , משתנה0 , "מחרוזת {1} מחרוזת {0} מחרוזת");**
Console. WriteLine **(משתנה1 , משתנה0 , "מחרוזת {1} מחרוזת {0} מחרוזת");**
Console. WriteLine **("מחרוזת {משתנה1} מחרוזת {משתנה0} מחרוזת$");**

דוגמא:
```
Console.WriteLine ("Average grade of {0} and {1} is {2}", grd1, grd2, avg);
Console.WriteLine ($"Average grade of {grd1} and {grd2} is { avg }");
```
השימוש בדרך זו מתאימה לפלט מעוצב כפי שיוסבר בהמשך, בסעיף הדן ב**פורמט הדפסה**.

שורות ריקות בפלט

לעיתים נרצה להציג שורות ריקות במסך הפלט. לשם כך נשתמש בפעולה **Consol.**WriteLine ();

אם נרצה להשאיר שתי שורות ריקות, נוכל לרשום: **Consol.**WriteLine ("\n");

המחרוזת "\n" משמעותה "רד לשורה הבאה" כלומר – השארת שורה ריקה. ההוראה WriteLine תציג על המסך את תוכן המחרוזת \n (שמשמעותה "רד לשורה הבאה") ואחר כך תרד שורה (כי כך נוהגת WriteLine להבדיל מ- Write).

אם נרצה שלוש שורות ריקות, נרשום: **Consol.**WriteLine ("\n\n");
(כל \n יגרום לירידת שורה נוספת).

כשמשתמשים בשיטה (פעולה) **Write()**, נשאר ראש הכתיבה באמצע השורה, במקום בו הסתיימה הכתיבה למסך. הפעלת השיטה **WriteLine()** תגרום להעברת ראש הכתיבה לתחילתה של השורה הבאה. ואילו ההוראה: **Consol.WriteLine ("\n");** תגרום לירידה לשורה הבאה ובנוסף השארת שורה ריקה.

שרשור (צירוף) מחרוזות

שרשור מחרוזות הוא צירוף מחרוזת אחת לסופה של האחרת. (שרשור מהמילה 'שרשרת', צירוף מחרוזות של תווים בזו אחר זו כמו חוליות בשרשרת). פעולת השרשור נעשית באמצעות האופרטור + (חיבור) הפועלת על משתני מחרוזת וקבועים מחרוזתיים.

פעולת השרשור להדפסת מחרוזות:
```
string name = "Dany";
Console.WriteLine ("Hello, " + name);
```

אם נרצה לשרשר מחרוזת וביטוי חשבוני, נתחום את הביטוי החשבוני בסוגריים.

לדוגמא:
```
string str = "number is: ";
int  a = 5;
```

ההוראה	המחרוזת המתקבלת
Console.WriteLine ("a is: " + a + " next " + str + a + 1);	a is: 5 next number is: 51
Console.WriteLine ("a is: " + a + " next " + str + (a+1));	a is: 5 next number is: 6

עברית בסביבת C#

לצערנו, סביבת העבודה של C# אינה תומכת בעברית כראוי. מחרוזות פלט שמופיעות בגוף התכנית בשפה העברית מוצגות כג'יבריש במסך הפלט. לצורך הנוחות והקריאות, מוצגות חלק ממחרוזות הפלט בעברית. תלמיד רשאי לכתוב במחברתו מחרוזות בעברית, ובלבד שיזכור לתרגמן לאנגלית בשלב כתיבת התכנית במחשב.

יוצא מן הכלל הוא הערות ותיעוד, שאינם מופיעים בפלט, ולכן ניתן לתעד את התכנית בעברית.

שים ♥ לגודל האות בשם מחלקה ובשם הפעולה.
(האות הראשונה בתחילת כל מילה הינה אות ראשית)

הוראת השמה

הוראת השמה משמשת את כותב התכנית להכנסת נתונים לתוך המשתנים. השמת נתון למשתנה מוחקת את הערך שהיה שמור במשתנה זה ושומרת את הערך החדש בלבד (ומכאן **משתנה** המשנה את ערכו).

א. השמת קבוע

מתכנת יכול להחליט מראש על ערך קבוע הדרוש לו בשלב מאוחר יותר בתכנית.

```
int       a, b, c ;
double  x, y, z ;
string    greeting ;

a  =     5 ;
b  =     -39 ;
x  =     2.35 ;
greeting  = "hello world !";
```

קרא:

```
<שם-משתנה> מקבל <קבוע> //          ; <קבוע>  =  <שם-תא-מקבל>
```

השמה של קבוע למשתנה שימושית לפעולה של **אתחול בהגדרה**. ניתן לתת למשתנים ערך התחלתי בשלב ההגדרה.

דוגמא:

```
int  a = 5, b = 21 ;
double  x = 7.32 ;
string str = "abc" ;
```

שים ♥: טיפוס הערך המושם בתא חייב להיות תואם את טיפוס הנתונים של התא המקבל.

ב. השמת ערך ממשתנה אחר

ישנם מצבים בהם מעוניין כותב התכנית להעביר ערך ממשתנה אחד למשתנה אחר. למשל, כשהוא מעוניין לשמור, לשימוש בהמשך, ערך הנמצא במשתנה מסויים, כאשר למשתנה זה עומד להיכנס ערך חדש.

```
<שם-תא-נותן>  =  <שם-תא-מקבל> ;
```

```
int  a = 12, b = 5, c ;
double  x = 5.24, y, z ;
```

משתנה מטיפוס ממשי (double) יכול לקבל תוכן של משתנה ממשי או של משתנה שלם.

```
z = x ;
y = b ;
```

משתנה מטיפוס שלם יכול לקבל תוכן של משתנה שלם בלבד.

```
c = a ;
```

ניסיון להשמת ערך ממשי (המכיל נקודה עשרונית), קבוע או מתוך משתנה, בתוך משתנה מטיפוס שלם הינו שגיאה. השמה הפוכה של שלם לתוך ממשי הינה אפשרית.

לדוגמא: השמת x לתוך c יגרור את הודעת השגיאה הבאה:

Cannot implicitly convert type 'double' to 'int'

שפירושה: אין אפשרות להמיר משתנה מטיפוס ממשי לשלם. הרחבה בפרק 4.

שים ♥: מועבר רק עותק מהמשתנה הנותן. ערכו של משתנה זה אינו משתנה.
אם במשתנה המקבל היה ערך כלשהו, ערך זה נמחק בעת ההצבה.

3. השמת תוצאה של חישוב

ברוב המקרים, הערך אותו רוצים להציב במשתנה הוא תוצאה של חישוב.

למשל, כאשר רוצים לחשב שטחו של מלבן, יש לכפול את אורכו ברוחבו. התכנית חייבת אם כן, לכלול פקודה שתורה למחשב להציב במשתנה המיועד לשטח המלבן את מכפלת אורך המלבן ברוחבו.

הערכים הנכללים בביטוי הם קבועים-מספרים או ערכים השמורים במשתנים מספריים. פעולת החישוב היא אחת מפעולות החשבון כפל, חילוק, חיבור ו/או חיסור.

מבנה השורה במקרה זה יהיה כמו במקרים הקודמים, אלא שמימין לסימן ההשמה יופיע ביטוי חשבוני.

```
<שם-משתנה-מקבל> = <ביטוי חשבוני> ;
```

דוגמאות:

```
c = a + 5 ;
y = x – 2*a ;
z = (a – y) / (a * b) ;
```

הביטוי החשבוני הוא החלק שנמצא מימין לסימן ההשמה = .
הביטוי יכול לכלול: קבועים מספריים, משתנים, חישובים מתמטיים ואף ערך מוחזר מפונקציות מתמטיות (כפי שנלמד בפרק 5).

הפעולות החשבוניות:

חיבור	+
חיסור	–
כפל	*
חילוק	/

שים ♥:

- סדר הקדימויות של החישובים המתמטיים הוא סדר הקדימויות המתמטי:
  כפל וחילוק קודמים לחיבור וחיסור.
  סוגריים משנים את סדר החישוב.
- המשתנה לא יכיל את הביטוי החישובי אלא את תוצאת הפעולה, כלומר – ערך מספרי.
- לאחר פעולת ההשמה, המחשב לא "זוכר" את הערך שהיה במשתנה קודם לכן.
- תוצאה של פעולה חישובית שבה מעורב ערך ממשי (קבוע או משתנה) תהיה תמיד ממשית. יש להקפיד שהמשתנה המקבל את התוצאה יהיה ממשי. ראה הרחבה בפרק 4.

דוגמא:

התכנית שלהלן ממירה סכום כסף הנתון בדולרים לערכם בש"ח בהתאם לשער המרה.
לצורך העניין, נבצע המרה של $100 לשקלים, לפי שער המרה של: $1 = 4.5 ₪.

טבלת משתנים:

שם המשתנה	טיפוס המשתנה	תפקיד המשתנה
dollar	int	הסכום בדולרים
yatzig	double	השער היציג
shekel	double	הסכום בשקלים

חשוב: מדוע יש לקבוע את משתנה השקלים כמשתנה מטיפוס ממשי? (מה יקרה אם נקבע אותו להיות משתנה מטיפוס שלם?).

האלגוריתם:

(1) dollar ← 100
(2) yatzig ← 4.5
(3) shekel ← dollar * yatzig
(4) פלט: הערך בשקלים הוא: shekel

התכנית:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~ *
 *      המרה מדולרים לשקלים      *
 *~~~~~~~~~~~~~~~~~~~~~~~~~ */
using System;
class Class1
{
        public static void Main (string [] args)
        {
                int dollar;
                double yatzig, shekel;

                dollar = 100;
                yatzig = 4.5;
                shekel = dollar * yatzig;
                Console.WriteLine ("the value is: " + shekel + " nis.");
        }

}
```

החיסרון של התכנית הוא שתמיד יומר סכום של $100 לפי שער יציג שנקבע ביום מסויים. כדי להמיר סכומים שונים בשער יציג עדכני, נזדקק לפעולת קלט, שתקלוט מן המשתמש את הסכום שברצונו להמיר ואת שער הדולר העדכני.

הוראת קלט

בעזרת הוראה זו ניתן לקלוט, בזמן ריצת התכנית, נתונים מן המשתמש ולשמור אותם במשתנים שהחלטנו עליהם מראש. ההוראה מאפשרת לתכנית לקרוא מן המסך (באמצעות המחלקה Console) מידע שכתב עליו המשתמש, באמצעות המקלדת.

```
משתנה = Console.ReadLine();
```

הפעולה **ReadLine** קוראת טקסט (**תווים** - **chars**) מן המקלדת (עד שנתקלת הסימן של מקש ה-Enter) ומחזירה **מחרוזת** (קטע טקסט). את המחרוזת המוחזרת יש להכניס לתוך משתנה.

קלט למשתנה מספרי

כבר ראינו שלכל אחד מטיפוסי המשתנים יש ייצוג שונה במחשב. ולכן, אם הקלט שלנו הוא מחרוזת יושם הערך המוחזר בתוך משתנה מטיפוס מחרוזת, אבל אם נרצה לקלוט מספר, עלינו **להמיר** (לתרגם) את הקלט מייצוג של טקסט לייצוג של מספר לפי סוג המשתנה.

הפעולה **Parse** המבצעת **המרה** ממחרוזת למספר ושייכת למחלקה של המשתנה המספרי. לדוגמא, אם נרצה להמיר מחרוזת למספר שלם, נשתמש בפעולה: **int.**Parse("מחרוזת המכילה ספרות"); ואם נרצה להמיר מחרוזת לממשי, נשתמש בפעולת ההמרה של double.

נחבר את שני השלבים של קריאת מחרוזת + הפיכה לשלם, ונקבל הוראת קלט ל**מספר שלם**:

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

ואם נרצה לקלוט **מספר ממשי**:

```
double x;
x = double.Parse (Consol.ReadLine( ));
```

קלט למשתנה תווי

```
char ch;
ch = char.Parse (Consol.ReadLine( ));
ch = (char)Console.Read();                    או:
```

הדרך הראשונה לוקחת את מחרוזת הקלט וממירה אותו לתו.
אם נקלט יותר מתו אחד במחרוזת הקלט - באפשרות הראשונה יודיע שגיאה, ובשנייה ייקלט התו הראשון ויתעלם המחשב משאר התווים.
הדרך השנייה מתאימה למקרים בהם יש לבצע פעולה על קבוצה של תווים. בדרך זו ניתן להקליד את רצף התווים בבת אחת, והמחשב ייקח בכל פעם את התו הבא.

שים ♥ :

- קלט חוקי עבור מספר שלם מורכב מספרות וסימני +/- לפני הספרה הראשונה.
  ניסיון להקליד תווים או נקודה עשרונית, יגרור הודעת שגיאה מסוג חריגה **(exception)** כלומר: חריגה מהכללים. ובהמשך הודעה הכוללת את המשפט:
  Input string was not in a correct format שמשמעותה: מחרוזת הקלט שגויה.
- קלט חוקי עבור מספרים שלמים מורכב מספרות ומסימני +/- לפני הספרה הראשונה.
  (אין חובה לשים + לפני מספר חיובי)
- קלט חוקי עבור מספרים ממשיים מורכב מספרות, מנקודה עשרונית, מסימני +/- לפני הספרה הראשונה ואת האות E כמעריך למספר בייצוג מדעי. ניסיון להקליד קלט אחר יניב הודעת שגיאה דומה: הקלט אינו מהטיפוס הדרוש.
- מספר שלם שייקלט למשתנה מטיפוס ממשי, יהפוך להיות מספר ממשי על ידי הוספת 0 אחרי הנקודה העשרונית.
- אם יוקלד יותר מתו אחד למשתנה מטיפוס char, יאוחסן בתא התו הראשון בלבד.

נשנה את התכנית שלנו, כך שייקלט המידע הדרוש מן המשתמש:

האלגוריתם:

(1) קלט: הסכום בדולרים ← dollar
(2) קלט: השער היציג ← yatzig
(3) shekel ← dollar * yatzig
(4) פלט: הערך בשקלים הוא: shekel

התכנית:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~ *
 *      המרה מדולרים לשקלים      *
 *~~~~~~~~~~~~~~~~~~~~~~~~~ */

using System;
class Class1
{
        public static void Main (string [] args)
        {
                int dollar;
                double yatzig, shekel;

                Console.Write("How many dollars you want to convert? → ");
                dollar = int.Parse(Console.ReadLine());

                Console.Write ("What is the rate of the dollar? → ");
                yatzig = double.Parse(Console.ReadLine());

                shekel = dollar * yatzig;

                Console.WriteLine ("the value is: " + shekel + " nis.");
        }
}
```

פורמט הדפסה – שליטה על מראה הפלט

הוראות ההדפסה גורמות להדפסת הנתונים בצורה שנקבעה מראש.

שימוש בסמנים

Console.WriteLine ("first value {0} second value {1} third value {2}", val0, val1, val2);

- הסמן {0} מוחלף על ידי הערך val0. הסמן {1} מוחלף על ידי הערך val1 והסמן {2} על ידי הערך val2.
- הסמן יהיה מספר שלם התחום בסוגרים מסולסלים. ערך הסמן הראשון יהיה תמיד {0}. ערכי כל סמן הבא אחר כך יהיה עוקב לערך הסמן הקודם לו.
  דילוג בסדר המספור הרציף יגרום לשגיאה בתכנית.
- לאחר סגירת המחרוזת יש לרשום את תוכן המשתנים או הקבועים אותם יש להדפיס, כך שכל משתנה או קבוע מתאים לסמן שלו, לפי סדר הופעתם במחרוזת. את הערכים יש להפריד בפסיקים.
- סדר הסמנים במחרוזת אינו חשוב, ובלבד שהוא יתאים לערך שיחליף אותו.
  Console.WriteLine ("The sum of {2} and {1} is {0}", 5+4, 4, 5);
- כל סמן יכול להופיע יותר מפעם אחת.

Console.WriteLine ("{0} is greater then {1}, {1} is greater then {2}.
Therefore {0} is greater then {2}", a, b, c);

שים ♥ : פיצול הוראה לשתי שורות או יותר אינה משפיעה על הביצוע.

- *** החל מ- Visual Studio 2017 ניתן להשתמש במחרוזת הפלט הבאה :
  Console.WriteLine ($"The sum of {a} and {b} is {a+b}");
  שימו לב לסימן ה- $ לפני המחרוזת. המשתנה או הביטוי החישובי שמופיע בתוך הסוגריים המסולסלים יוחלף בערך המשתנה או ערך הביטוי.

פורמט הדפסה לטקסט

- על מנת לשמור על מרווח אחיד בפלט, ניתן לקבוע כמה תווי מסך יוקצו לכל סמן.

Console.WriteLine ("Your name is {0,7} {1,8}", "Burt", "Simpson");

והפלט יהיה : _ _ _ B u r t _ S i m p s o n
(הסימן _ מציין תו אחד במסך)

הערכים שיחליפו את הסמנים ישתרעו על פני 7 ו- 8 תווי מסך בהתאמה, כאשר היישור יהיה לצד ימין. אם נקבע רוחב מרווח קטן ממספר הספרות שבמספר, תתעלם ההוראה מגודל המרווח, והמספר יוצג בשלמותו.

על מנת ליישר את הערכים **לצד שמאל**, נרשום גודל מרווח **שלילי** :

Console.WriteLine ("Your name is {0,-7} {1,-8}", "Burt", "Simpson");

והפלט יהיה : B u r t _ _ _ S i m p s o n _

פורמט הדפסה למספרים

כדי שנוכל להציג מספרים באופן ברור יותר, ניתן להגדיר את **רוחבו** של **שדה ההדפסה** בתוספת דיוק עשרוני למספרים ממשיים, או הפרדת פסיקים לאחר כל 3 ספרות למספרים ארוכים.

רוחב שדה הדפסה: מספר תווי המסך המוקצים להדפסת הערך.

- **הדפסת מספר שלם:**

Console.WriteLine ("Num = {0}", num);

הערך שבמשתנה num יחליף את הסמן {0}. המרחק של הערך מהתו '=' מותנה במספר תווי הרווח שבין התו '=' לסמן. הקפד שיופיע לפחות תו רווח אחד.

- הדפסת מספר שלם תוך ציון רוחב שדה ההדפסה (מספר תווי המסך):

Console.WriteLine ("Num = {0,5}", num);

המספר יודפס ב-5 תווים מיושר לימין. ההוראה לא מקצצת במספר. כלומר, אם מספר הספרות במספר גדול מהרוחב שהוקצה, תתעלם ההוראה מרוחב זה ותציג את המספר בשלמותו.

- קביעת רוחב שדה "שלילי" תגרום ליישור המספר לשמאל.

- **הדפסת מספר ממשי:**

Console.WriteLine ("Average is {0}", avg);

הערך המספרי שב- avg יחליף את הסמן {0}.

- הדפסת מספר ממשי תוך ציון מספר הספרות אחרי הנקודה העשרונית:

Console.WriteLine ("Average is {0:F1}", avg);

ההוראה אינה משפיעה על תוכן המשתנה אלא על אופן התצוגה בלבד.
הספרה 1 קובעת שתצוגת המספר תעוגל לדיוק של 1 ספרות עשרוניות.
ברירת המחדל אם לא צוינה סיפרה היא 2 ספרות עשרוניות.

- הדפסת מספר ממשי בדיוק עשרוני קבוע תוך שימוש ברוחב שדה ההדפסה:

Console.WriteLine ("Average is {0,8:F2}", avg);

תוכן המשתנה avg יוצג ב-8 תווים, מיושר לימין ובדיוק של 2 ספרות אחרי הנקודה העשרונית.
שים ♥: הנקודה העשרונית תופסת מקום אחד.

- הדפסת מספר שלם:

Console.WriteLine ("There is a {0} dollar in the bank", 1000000);

הפלט There is a 1000000 dollar in the bank מוצג בצורת המקשה על קריאת המספר.

- הפרדת המספר בפסיקים:

Console.WriteLine ("There is a {0:N} dollar in the bank", 1000000);

תגרום להצגת הפלט בצורה ברורה יותר: There is a 1,000,000.00 dollar in the bank
הוספת N למספר שלם, תציג את המספר בדיוק של שתי ספרות עשרוניות.

באופן כללי, נשיג שליטה על רוחב שדות ההדפסה בעזרת המבנים הבאים:

עבור **מספרים שלמים** (משתנה או קבוע מספרי):

```
Console.WriteLine (" {0}", משתנה ) ;
Console.WriteLine (" {0, ± n}", משתנה ) ;   // רוחב שדה
Console.WriteLine (" {0:N}", משתנה ) ;      // פסיקים מפרידים
Console.WriteLine (" {0,± n:N}", משתנה ) ;  // רוחב שדה + פסיקים
```

עבור **מספרים ממשיים** (משתנה או קבוע מספרי):

```
Console.WriteLine ("{0}", משתנה ) ;
Console.WriteLine ("{0:F}", משתנה ) ;
Console.WriteLine ("{0:Fn}", משתנה ) ;
Console.WriteLine ("{0,± n:Fn}", משתנה ) ;
```

<u>הערות</u>: שימוש ב- N עבור משתנה ממשי, יוסיף פסיקים מפרידים למספר
אין חשיבות לגודל האות במתגים F ו- N .
n מייצג מספר שלם וחיובי.

תכנית דוגמא המציגה פורמט הדפסה:

```
/*~~~~~~~~~~~~~~~~~~~~~~~~~~~ *
 *        תכנית להצגת פורמט הדפסה    *
 *~~~~~~~~~~~~~~~~~~~~~~~~~~~ */

using System;
class Class1
{
    public static void Main (string [] args)
    {
        double x, y, z;

        Console.Write ("Type 1st number ---> ");
        x = double.Parse(Console.ReadLine());

        Console.Write ("Type 2nd number ---> ");
        y = double.Parse(Console.ReadLine());

        Console.Write ("Type 3rd number ---> ");
        z = double.Parse(Console.ReadLine());

        Console.WriteLine();
        Console.WriteLine ("1st number is:  {0,10:F2}", x);
        Console.WriteLine ("2nd number is:  {0,10:F2}", y);
        Console.WriteLine ("3ed number is:  {0,10:F2}", z);
        Console.WriteLine ("{0,26}" ,"----------") ;
        Console.WriteLine ("the sum is:      {0,10:F2}" , x+y+z ) ;
    }
}
```

/*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ פלט התכנית: ~~~~~~~~~~~~~~

```
Type 1st number ---> 2.5
Type 2nd number ---> 123.026
Type 3ed number ---> 11

1st number is:       2.50
2nd number is:     123.03
3ed number is:      11.00
                 --------
the sum is:        136.53
```

~~~~~~*/

שים ♥: הנקודה העשרונית תופסת מקום אחד מ-10 התווים שהוקצו לרוחב שדה ההדפסה.
~~~~~~

תרגילים

משתנים

1. קבע אלו מבין הביטויים הבאים הוא שם תקין של משתנה:

(1) Y _____ (6) X% _____

(2) 3to2 _____ (7) main _____

(3) x_to_y _____ (8) newNum _____

(4) A4 _____ (9) Abcd _____

(5) x y _____ (10) My-Name _____

2. מדוע רצוי ששמות משתנים בשפה יהיו משמעותיים? _____

_____

3. איך ניתן לשלב הערה בתוכנית? מה חשיבות כתיבת הערות בתכנית? _____

_____

האם ניתן לכתוב הערה בכל מקום בתכנית? (גם באמצע מילה?) _____

_____

4. ברצוננו לשמור במשתנים בתכנית את המידע הבא:

א. מספר התלמידים בכיתה. _____

ב. שם בית הספר. _____

ג. השנה הנוכחית. _____

ד. מספר קטלוגי של פריט. _____

ה. מחיר הפריט. _____

ו. כמות. _____

ז. הרווח לאותו פריט. _____

ח. שם הפריט. _____

קבע עבור כל משתנה את טיפוס המשתנה וכתוב הצהרות מתאימות למשתנים אלו.

מבנה התכנית והוראות פלט

5. בכל אחת מהקביעות הבאות, הקף בעיגול אם **נכון** או **לא נכון**. אם לא נכון, תקן את המשפט כך שיהיה נכון.

נכון / לא נכון א. בכל תכנית חייבת להיות הפעולה Main. _____

נכון / לא נכון ב. כדי לקבל קובץ ריצה, חובה לבצע פעולת הידור על הקובץ. _____

נכון / לא נכון ג. חובה לצרף הוראות הסבר בתחילת כל תכנית מחשב. _____

נכון / לא נכון ד. שם קובץ יכול להיות בעברית. _____

נכון / לא נכון ה. שם קובץ יכול להכיל סימן רווח. _____

נכון / לא נכון ו. חובה לשים סימן נקודה-פסיק (;) בסוף כל הוראת התכנית. _____

נכון / לא נכון ז. תיעוד לתכנית פירושו שמוסיפים מסמכים ותעודות רשמיים לתכנית. _____

נכון / לא נכון ח. הזחה פירושה הכנסת הכתוב כמה תווי רווח מתחילת השורה._____

נכון / לא נכון ט. כשכותבים תכנית, אין חובת הזחה. _____

נכון / לא נכון י. תיעוד קטע יכול להימשך על פני מספר שורות._____

נכון / לא נכון יא. מילה שמורה היא מילה שיש לה משמעות ותפקיד מיוחד בשפה. _____

נכון / לא נכון יב. הדרך הנכונה להשיג הזחה בתכנית היא באמצעות מקש tab._____

נכון / לא נכון יג. כל עוד קיימות שגיאות בתכנית, לא יתקבל קובץ ריצה (בשפת מכונה). _____

נכון / לא נכון יד. חובה שיהיו משתנים בכל תכנית. _____

נכון / לא נכון טו. הוראות הקלט והפלט בתכנית, שייכות למחלקה System _____

6. אחדות מהוראות הפלט שלפניך שגויות. סמן אותן והסבר מהן הטעויות:

(1) Console.WriteLine ("Programming in C#"); _____

(2) Console.WriteLine (hello world!); _____

(3) Console.Write (" "); _____

(4) Console.Write ("Console.WriteLine (hello world !) ");_____

(5) Console.WriteLine (' SHANA TOVA '); _____

(6) Console.WriteLine "we have not yet begun" ; _____

(7) Console.WriteLine ("Have a nice day") ; _____

(8) Console.Write ("what a beautiful morning !") _____

7. בחלק מהתכניות הבאות נפלו שגיאות תחביריות. ציין והסבר את השגיאות:

(1)

```
using System;
namespace firstProgram
{
    class Class1
    {
        static void Main (string [] args)
        {
            Console.WriteLine ("Hello world ! ");
        }
    }
}
```

(2)

```
using System;
namespace firstProgram
{
    public static void Main (string [] args)
    {
        Console.WriteLine ("בוקר טוב עולם! מה שלום כולם");
    }
}
```

(3)

```
using System;
namespace firstProgram
{
    class Class1
    {
        static Main()
            Console.WriteLine ("Learning C# is fun");
    }
}
```

(4)

```
using System;
class Class1
{
    static void Main (string [] args)
    {
        Console.WriteLine ("Have a wonderful day!  ");
    }
}
```

(5)

```
/ *     תכנית מחשב בשפת C#
 *      לכבוד חופשת סוכות.
 * /
using System
namespace firstProgram
{
    class Class1
    {
        public static void Main (string [] args)
        {
            Console.Write ("... נהדרת ויפה ")
            Console.WriteLine ("  שלומית בונה סוכה ")
            Console.WriteLine (" שלומית בונה סוכת שלום")
        }
    }
}
```

כיצד יראה הפלט עבור קטע זה? הסבר!

(6)

```
/ *     תכנית מחשב בשפת C#
 * /
using System;
namespace firstProgram
{
    class Class1
    {
        public static void Main (string [] args)
        {
            console.writeline (" יש לי יום יום חג");
            console. writeline (" יש לי חג יום יום.");
        }
    }
}
```

(7)

```
/ *     תכנית מחשב בשפת C#
 * /
using System;
namespace firstProgram
{
    class Class1
    {
        public static void Main (string [] args)
        {
            writeline (" יש לי יום יום חג");
            writeline (" יש לי חג יום יום.");
        }
    }
}
```

משפטי השמה

8. כתוב משפטי השמה לביצוע הפעולות הבאות:

פעולה	משפט השמה
א. הקטנת ערכו של משתנה a ב- 5.	
ב. הגדלת ערכו של משתנה b פי 3.	
ג. הכפלת ערכו של המשתנה a ב- 2.	
ד. החסרת ערך המשתנה b מהמשתנה a.	
ה. הכפלת ערכו של המשתנה c במשתנה a .	
ו. הוספת 5 למשתנה a .	
ז. איפוס המשתנה x.	
ח. הגדלת ערכו של המשתנה d ב- 8.	
ט. הגדלת ערכו של המשתנה c פי 8.	
י. הגדלת ערכו של המשתנה a במשתנה b.	
יא. הגדלת ערכו של המשתנה h בפעמיים המשתנה s	
יב. הקטנת ערכו של משתנה c פי 10.	
יג. הגדלת ערכו של משתנה d ב- 20%.	
יד. הקטנת ערכו של משתנה f ב- 30%.	

9. כתוב הוראות השמה לביצוע הפעולות הבאות:

פעולה	משפט השמה
(1) שים במשתנה a את הסכום המתקבל מחיבור ערכו של המשתנה a לערכו של המשתנה b.	
(2) שים במשתנה c את ההפרש המתקבל מחיסור מכפלת ערכו של המשתנה d ב- 2 מהערך שבמשתנה c	
(3) שים במשתנה e את הסכום המתקבל מחיבור מחצית ערכו של המשתנה e לחמש פעמים ערכו של המשתנה f.	
(4) שים במשתנה g את ההפרש המתקבל מחיסור 80% מערכו של המשתנה g מפעמיים ערכו של המשתנה h.	

כתיבת תכנית מחשב

הוראות הדפסה

התרגילים 10 עד 16, מומלץ לתכנן תחילה את המטלה המבוקשת על גבי נייר משבצות.

10. א. כתוב תכנית מחשב שתדפיס עבורך כרטיס ביקור הכולל את שמך, כתובתך ומספר הטלפון שלך.
    ב. הוסף הוראות לתכנית כך שכרטיס הביקור יהיה תחום במסגרת.
11. כתוב תכנית מחשב שתקלוט מן המשתמש את שמו (לתוך משתנה מחרוזת) ואת גילו, ותדפיס עבורו כרטיס ביקור.
12. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת על המסך ריבוע מלא מתווי כוכביות. גודל הריבוע לא יהיה קטן מ-7*7.
13. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת על המסך ריבוע חלול בעזרת כוכביות. גודל הריבוע לא יהיה קטן מ-7*7.
14. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת על המסך שני מלבנים, זה לצד זה, בעזרת כוכביות. גודל כל מלבן לא יהיה קטן מ-5 שורות (אורך) ו- 10 עמודות (רוחב).
15. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת את שמך על המסך מתווי כוכביות. ציור השם יתפרס על לפחות מחצית המסך.
16. כתוב תכנית מחשב המשתמשת בהוראות הדפסה, ומציירת ציור כלשהו על המסך, לדוגמא: בית, מכונית וכד'.

קלט, פלט והוראות השמה

17. כתוב תכנית מחשב הקולטת למשתנה מטיפוס int מספר שלם, ומדפיסה אותו באופן הבא:
    בשורה ראשונה יודפס המספר פעם אחת.
    בשורה השנייה יודפס המספר 2 פעמים.
    בשורה השלישית יודפס המספר 3 פעמים.
18. כתוב תוכנית מחשב, בה יוצב במשתנה a המספר 3, ובמשתנה b המספר 5. התכנית תדפיס את הסכום b+a.
19. כתוב תוכנית מחשב, בה יוצב 2 במשתנה a, 3 במשתנה b, ו- 5 במשתנה c. התכנית תדפיס את הערך של $4ac - b$.
20. כתוב תכנית מחשב, לקליטת שני מספרים והדפסת מכפלתם בתוספת לכותרת: "המכפלה:".
21. כתוב תכנית מחשב, לקליטת נתון מספרי למשתנה a והדפסת מספר הגדול ממנו ב- 2.
22. כתוב תכנית מחשב, לקליטת נתון מספרי למשתנה x והדפסת 3 מספרים עוקבים, החל מ- x.
23. כתוב תכנית מחשב, לקליטת נתון מספרי למשתנה x והדפסת 3 מספרים קודמים ל- x (x יהיה המספר האחרון).
24. כתוב תכנית מחשב המציבה את הערך 5 במשתנה num. על התכנית לחשב ולהדפיס את המספרים הבאים, כל הדפסה תעשה בשורה נפרדת (כל ההוראות מתייחסות לערך שבמשתנה num):
    א. מספר הגדול ממנו ב- 5.
    ב. מספר הקטן ממנו ב- 10.
    ג. מספר הגדול ממנו פי 7.
    ד. מספר הקטן ממנו בחצי.

25. כתוב תכנית מחשב המציבה את המספרים 7 ו- 3½ במשתנים. על התכנית לחשב ולהדפיס:
   א. את סכומם.
   ב. את הפרשם.
   ג. את מכפלתם.
   ד. את תוצאת החילוק של הראשון בשני.

26. כתוב תכנית מחשב שתאחסן את הערך 5 במשתנה num1 ואת הערך 8 במשתנה num2, על התכנית לחשב ולהציג כפלט:
   א. מספר הגדול ב- 10 מסכום שני המשתנים.
   ב. מספר הקטן פי 3 מהפרש המשתנים.
   ג. מספר הגדול פי 4 מהמשתנה num1.
   ד. מספר הקטן ב- 25 ממשתנה num2.

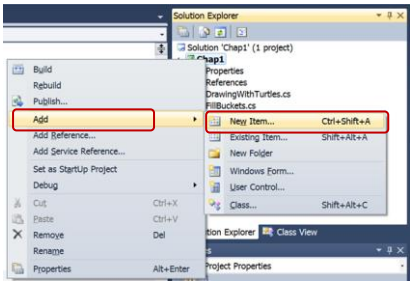
27. כתוב תכנית מחשב הקולטת למשתנה מטיפוס int מספר שלם, ומדפיסה את הפלט הבא:
   בשורה ראשונה יודפס המספר.
   בשורה השנייה יודפס המספר, ואחר כך שני המספרים העוקב לו.
   בשורה השלישית יודפס המספר, מכפלתו ב- 2, מכפלתו ב- 3 .
   להלן דוגמת פלט עבור הקלט 5 :

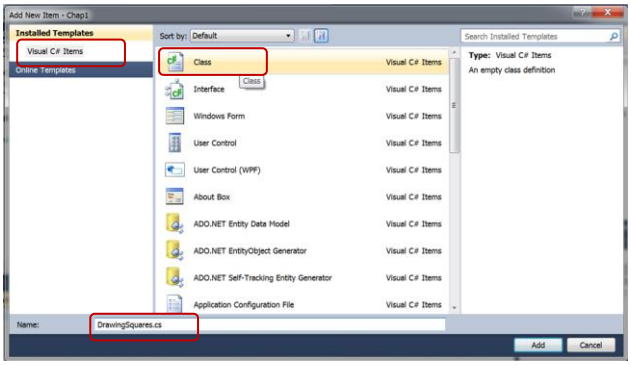
```
5
5 6 7
5 10 15
```

דף עבודה
צבים
שילוב הוראות קלט

פתח את המחלקה DrawingWithTurtles שבפרויקט Chap1 והוסף מחלקה חדשה בשם DrawingSquares (ציור ריבועים) באופן הבא:

I **הוספת מחלקה לפרויקט:**
קליק ימני על שם הפרויקט בחלון Solution שבצד ימין של המסך.
בחירה ב- Add ומתוכו: New Item

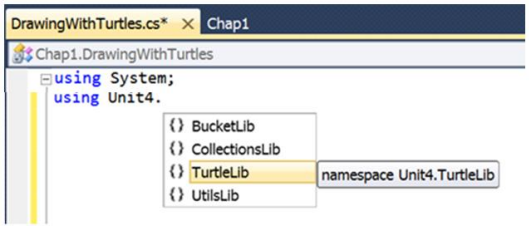




בחלון Add New Item
נבחר ב- class מסוג C#
וניתן שם לתכנית.

II **כתיבת כותרת המחלקה Main:** נרשום במחלקה שיצרנו את כותרת הפעולה Main (ניתן להעתיק שורה זו מתכנית אחרת שנמצאת בפרויקט).

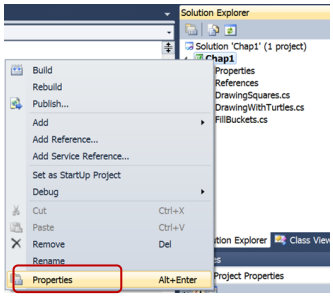
III **שילוב חבילת התוכנה Unit4 המכילה את המחלקות הגרפיות לטיפול בצב:**
נרשום בשורה הראשונה הוראה לקישור ל- Unit4 המגדיר את הצב, באופן הבא:
נרשום את המילים **using** Unit4. לאחר שנרשום נקודה, נשים לב ש-Visual Studio מציע לבחור את החבילה המתאימה מתוך רשימה של חבילות תוכנה.
(אם הרשימה לא מופיעה, נסה ללחוץ על צירוף המקשים <Ctrl> + <מקש הרווח>)



נבחר בחבילת התוכנה: TutrleLib

לתכנית התווספה השורה:

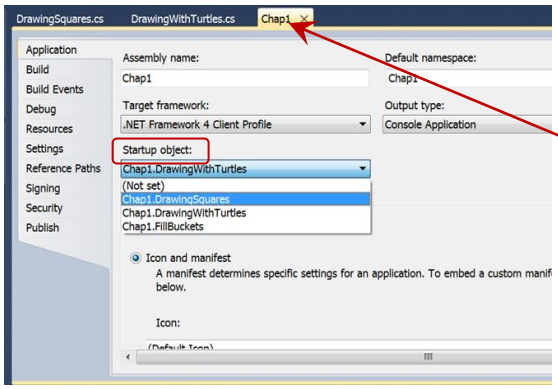
```
using Unit4.TurtleLib;
```



IV **קביעה איזו תכנית תרוץ:**
קליק ימני על שם הפרויקט ובחירה במאפיינים - Properties

מתוך תיבת הבחירה
Startup object נבחר
בתכנית שברצוננו להריץ
(המחלקה : DrawingSquares)

שמירה (^S)
(הכוכבית שליד שם חלון הפרויקט
נעלמה, דבר המעיד שהתכנית
נשמרה).



המשימות

1. בדף העבודה הקודם, ה**מתכנת** הוא זה שקבע את גודל הצעד שיזוז הצב.
עליך להוסיף לתכנית הוראות קלט שיבקשו מה**משתמש** בתכנית את גודל הצעד שיזוז הצב.

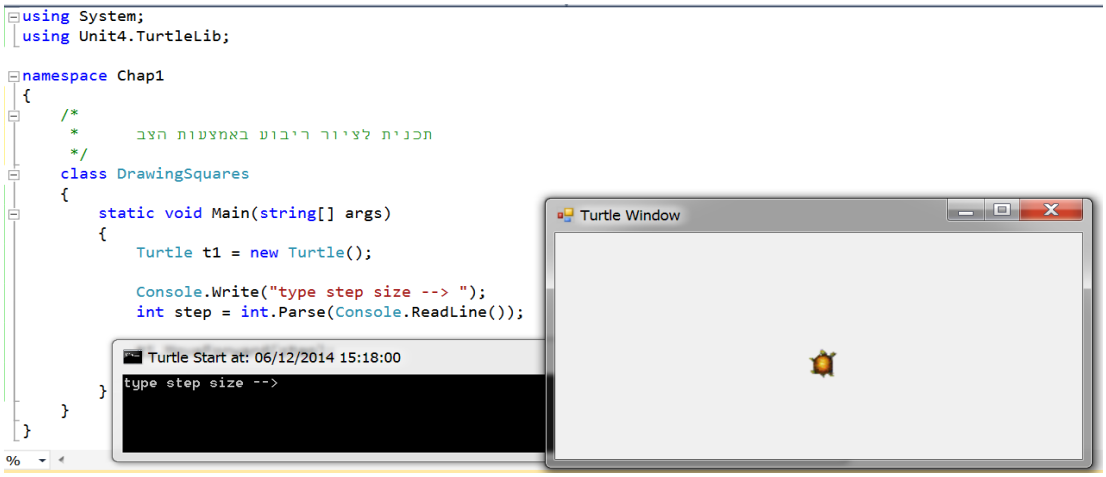
א. צור צב חדש :

```
Turtle t1 = new Turtle();
```

שים לב לגודל אות, לסוגריים ולסימן ה- ;

שם הצב שיצרנו הוא t1. ניתן לבחור שם אחר, ובלבד שיעמוד בכללים של מתן שם למשתנה.

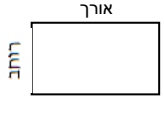
ב. הגדר משתנה מטיפוס מספר שלם (**int**) בשם step, וכתוב הוראת קלט שתקלוט לתוכו את גודל הצעד שיבצע הצב (גודל הצעד = אורך הקו שיצייר הצב).



ג. שנה את ההוראה הגורמת לצב לנוע קדימה, מ- `t1.moveForward (100);`
ל- `t1.moveForward (step);`

2. כתוב שתגרום לצב לצייר ריבוע. על התכנית ליצור צב חדש, לקלוט עבורו את אורך צלע הריבוע למשתנה בשם side, ולצייר ריבוע שאורך צלעו הוא side.

3. כתוב שתגרום לצב לצייר מלבן. על התכנית ליצור צב חדש,
לקלוט עבורו את אורך צלע המלבן למשתנה בשם side,
ולצייר מבלן שאורכו side ורוחבו פי 2 מרוחבו.



4. על מנת לצייר מצולע משוכלל (שאורך כל צלעותיו שוות זו לזו) בעל n צלעות, יש לגרום לצב לפנות בכל פעם 360/n מעלות ימינה (או שמאלה).
לדוגמה: עבור מחומש משוכלל, יש לצייר 5 צלעות, וזווית הפנייה היא: 360/5 = 72
עבור משושה משוכלל, יש לצייר 6 צלעות, וזווית הפנייה היא: 360/6 = 60

כתוב תכנית שתקלוט את מספר הצלעות במצולע למשתנה n, ואת גודל הצלע למשתנה side, ויצייר מצולע בעל n צלעות באורך side כל אחת.

עבודה נעימה !