

נניח שיש לנו צורך בתוכנית לבצע הדפסה של מגן דוד עשוי מכוכביות (לכבוד יום העצמאות) בעזרת פקודות של Console.WriteLine

כולנו מכירים את פקודת ההדפסה ולכן זה תרגיל די פשוט (אבל ארוך).

מה אם היינו צריכים לבצע הדפסה כזו 5 פעמים בתוכנית? אפשר על ידי העתק-הדבק. מה אם היינו צריכים לעשות זאת ב-20 מקומות (העתק-הדבק גורם להגדלה של התוכנית ולכן תופס יותר מקום בזיכרון)

מה אם היינו צריכים כל פעם מגד דוד בגודל אחר?

מה אם היינו צוות של 2, אחד כותב איזה חישוב שהתוכנית מבצעת והשני רוצה לעבוד על הדפסת המגן דוד, אבל היינו רוצים לעבוד במקביל?

ניקח דוגמה מהעולם האמיתי:

2 מתכנתים עובדים על פיתוח מעבד התמלילים וורד. אחד רוצה לעבוד על התפריט של 'קובץ' והשני על תפריט 'עריכה'. בסופו של דבר נרצה ששני הדברים יהיה במעבד (ונרצה ששניהם יעבדו במקביל).

מה אם אחד מהם סיים והשני עדיין לא - איך ניתן למשתמשים את וורד (נניח שנרצה לתת אותו עם התפריט של 'קובץ' אפילו אם 'עריכה' אינו מוכן)

אחד הפתרונות עבור כל הבעיות הנ"ל היא ההמצאה של קטע תוכנית עצמאי שניקרא: פונקציה או פעולה.

מהי פונקציה או פעולה או מתודה? אפשר לומר שזו למעשה מיני-תוכנית.

זהו קטע תוכנית שיש בו פקודות ואפשר לבצע אותו על ידי איזכור שמו.

למעשה מה שכתבנו בתור תוכנית עד היום היו פקודות שרוכזו כולן בבלוק שמתחיל בשם: Main. זוהי למעשה פונקציה מיוחדת שאותה יודע C# לזהות כפונקציה הראשית של התוכנית.

בצורה יותר מדויקת:

1. פונקציה זהו סט פקודות (בלוק בין צמד סוגריים מקושטים) שמתחיל עם כותרת הפונקציה וסוגריים. דוגמה לכותרת:

```
public static int func1(int a, double b)
```

2. הפונקציה יכולה לקבל 'פרמטרים', אבל לא חייבת. הפרמטרים אלה משתנים שניכתבים בתוך הסוגריים של כותרת הפונקציה. בדוגמה, זוהי פונקציה בשם: func1, שמקבלת שני פרמטרים, הראשון מסוג שלם (a) והשני מסוג דאבל (b). לא נתעכב כרגע על המילים: public static, אבל לאחריהן מופיע הסוג: int - זהו סוגו של הערך המוחזר על ידי פונקציה. כלומר, לאחר ביצוע הפונקציה, היא יכולה להחזיר ערך כלשהו (או שלא)

3. הנה עוד דוגמה לכותרת של פונקציה בשם rnd שלא מקבלת שום פרמטר ומחזירה ערך מסוג דאבל:

```
public static double rnd()
```

על מנת לכתוב פונקציה, יש לדעת מה היא מקבלת כפרמטרים ומה היא מחזירה. הפונקציה יכולה לקבל הרבה פרמטרים (או כלום), אבל **יכולה להחזיר עד ערך אחד (או כלום)**. הנה דוגמה לפונקציה בשם: nada שלא מקבלת שום פרמטר ולא מחזירה שום ערך:

```
public static void nada()
```

**מילת המפתח: void**, מסמלת שהפונקציה אינה מחזירה שום ערך. הסוגריים הריקות מראות שאינה מקבלת שום פרמטר.

הפונקציה שעליה דיברנו שמדפיסה מגן-דוד יכולה לקבל פרמטר שזה גודל הצלע שלו, אבל אינה מחזירה ערך, אלא פשוט מדפיסה אותו.

ניכתוב כעת פונקציה בשם: print שלא מקבלת שום פרמטר ולא מחזירה שום ערך:

```
public static void print()
{
    Console.WriteLine("Hello World");
}
```

כעת נשתמש בפונקציה בתוך ה- Main שלנו:

```
public static void Main(string[] args)
{
    print();
}
```

4. יש להבדיל בין הגדרת הפונקציה לבין השימוש בה. השימוש בפונקציה נעשה בעזרת כתיבת שמה + ציון הפרמטרים המתאימים בתוך הסוגריים הצמודות לה. כפי שעשינו בדוגמה, הגדרנו את הפונקציה: print והשתמשנו בה בתוך ה- Main. התוצאה תהיה הדפסת המחרוזת: Hello World

5. אם הפונקציה מקבלת פרמטרים, הם צריכים להיות תואמים במיקום ובסוג בין הקריאה וההגדרה. כעת ניראה דוגמה של פונקציה בשם: square שמקבלת כפרמטר מספר מסוג int ומחזירה ערך מסוג: int

```
public static int square(int num)
{
    return num*num;
}
```

שימו לב שלפני שם הפונקציה (square) כתבנו את סוג הערך המוחזר (int), ובסוגריים הגדרנו את סוג הפרמטר (int). פונקציה זו מקבלת מספר שלם ומחזירה אותו בריבוע. החזר הערך נעשה ע"י פקודת ה- return. אם פונקציה מחזירה ערך זה תמיד יהיה על ידי return. כעת ניראה כיצד להשתמש בפונקציה: square בתוך ה- Main

```
public static void Main(string[] args)
{
    Console.WriteLine("Please enter a number, whose square you'd like to get");
    int input_num=int.Parse(Console.ReadLine());
    Console.WriteLine(square(input_num));
}
```

מה קרה פה? הגדרנו משתנה מסוג int וקראנו לו: input\_num. קראנו לתוכו מספר שהקיש המשתמש לאחר מכן הדפסנו, ובתוך פקודת ההדפסה השתמשנו בפונקציה: square. (זוהי הקריאה לפונקציה)

קראנו לפונקציה ע"י איזכור שמה, עם המשתנה: input\_num כפרמטר. בפונקציה עצמה הגדרנו את הפרמטר בשם: num - ולכן, **הערך של: input\_num הופך להיות הערך של: num** בתוך הפונקציה. הפונקציה תחזיר את המספר כפול עצמו (כלומר בריבוע). אם המשתמש הקיש 9, תהיו סמוכים ובטוחים

6. הפונקציה יכולה להחזיר ערך או לא להחזיר ערך. אם היא מחזירה, סוג הערך המוחזר צריך להיות בכותרת. אם היא אינה מחזירה ערך, בכותרת כערך מוחזר **חייבת להיות המילה: void**

7. אם הפונקציה אינה מחזירה דבר (הוגדרה עם סוג החזר void), אסור שיהיה בה משפט: return

8. משפט: return תמיד מסיים את מהלך הפונקציה. עוד רגע ניראה מה המשמעות של זה. פרמטרים שניתנים בקריאה צריכים להיות תואמים בסוגם לאלה שהוגדרו בפונקציה (על פי סדר הופעתם בסוגריים). מייד ניראה דוגמה:

דוגמה לפונקציה בשם: search שמקבלת שני פרמטרים. הראשון מספר שלם והשני מערך. הפונקציה תחזיר אמת אם המספר השלם נימצא במערך ושקר אם אינו נימצא:

```
public static bool search(int value1, array1[])
{
    for(int i=0; i<array1.Length; ++i)
    {
        if (value1 == array1[i])
            return true;
    }
    return false;
}
```

**הסבר:** המאפיין Length שמוצמד ל- array1 נותן את גודלו, כך שהפונקציה תעבוד עם כל גודל של מערך שנותנים לה.

בודקים בלולאה על ידי השוואת התא התורן במערך עם הערך שניתן כפרמטר לפונקציה. אם אחד הערכים היה שווה - **הפונקציה מסתימת בפקודה: return true** - תזכרו שכל פקודת return מסיימת את ביצוע הפונקציה ומחזירה אותנו למקום שממנו ניקראה הפונקציה.

**מה תפקידה של הפקודה: return false; ?** פשוט מאד, אם הגענו לפקודה זו סימן שהסתיימה הלולאה שמשווה בין value1 ובין כל תא במערך ולא נימצא שוויון כזה (שהרי אם היה, היינו יוצאים מהפונקציה בעזרת הפקודה: return true;)

זהו 'טריק' ידוע בפונקציות שיכולים להיות מספר משפטי return וכל אחד מהם מסיים את הפונקציה, אם המחשב הגיע לביצועו.

דוגמה לשימוש ב- search

```
public static void Main(string[] args)
{
    int val= 123;
    int [] arr= new int[10] {10,4,100,23,2,0,-50,30,11,28};
    if (search(val,arr))
        Console.WriteLine("Value is found in the array!!!");
    else
        Console.WriteLine("Value is NOT ound in the array!!!");
}
```

תנחשו מה תדפיס התוכנית.

שימו לב שאי אפשר לקרוא לפונקציה כך:

```
search(arr, val);
```

01/05/2017

(הסדר של הפרמטרים בקריאה לפונקציה אינו תואם לסידורם בהגדרתה, הראשון צריך להיות השלם  
והשני המערך של שלמים)