

# מבוא לתכנות

הרצאה 6 – פונקציות II ומחרוזות

סמסטר 1

# לפי ערך

- הארגומנטים נשלחים לפונקציה לפי ערכם. הכוונה היא שרק הערך משתייך לפרמטר ולא המשתנה עצמו שנשלח במקור. לדוגמה:

```
static void Main(string[] args)
{
    int x=7;
    Console.WriteLine(x); //print 7
    AddOne(x);
    Console.WriteLine(x); //print 7
}

static void AddOne(int num)
{
    Console.WriteLine(num); //print 7
    num++;
    Console.WriteLine(num); //print 8
}
```

# לפי הפניה

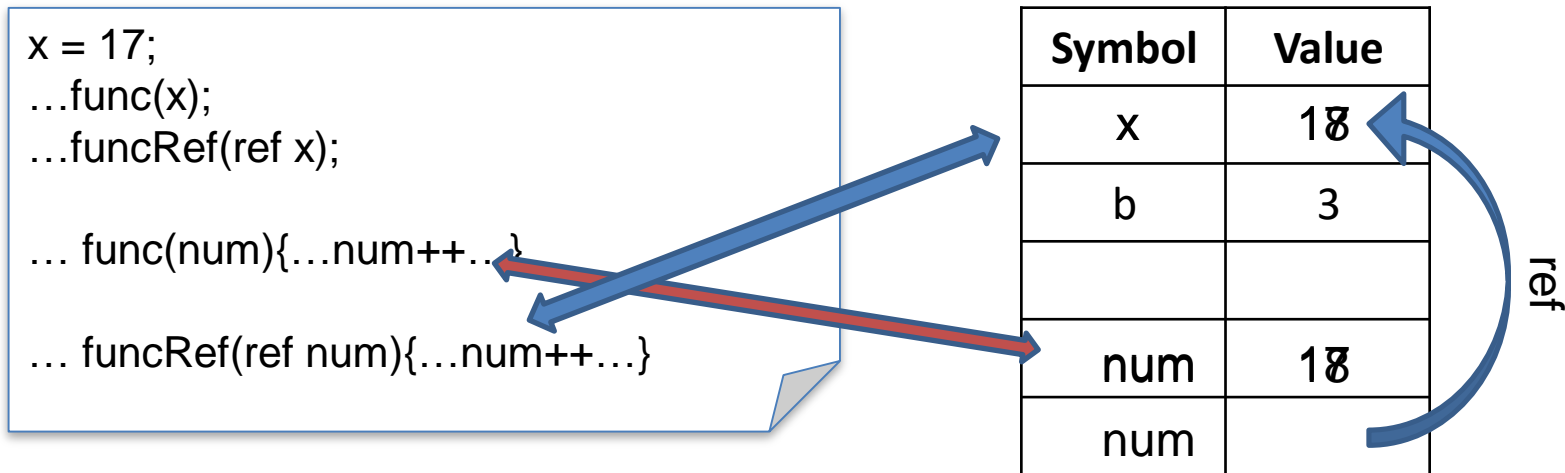
- הארגומנטים נשלחים לפונקציה לפי הפנייתם. כלומר המשתנה עצמו נשלח, ולא רק ערכו. כל שינוי בפונקציה ישפיע על המשתנה המקורי.  
לדוגמה:

```
static void Main(string[] args)
{
    int x=7;
    Console.WriteLine(x); //print 7
    AddOneRef(ref x);
    Console.WriteLine(x); //print 8
}

static void AddOneRef(ref int num)
{
    Console.WriteLine(num); //print 7
    num++;
    Console.WriteLine(num); //print 8
}
```

# ערך אל מול הפניה

- כשאנו נעשה שימוש בהפניה נצטרך להוסיף *ref* לפני הגדרת הארגומנט ולפני הגדרת הפרמטר.
- מאחורי הקלעים:




# out

- אנו יכולים להשתמש במילת המפתח out כדי לקבל ערך מהפונקציה חזרה אל הקורא לה. בדרך זו נוכל להחזיר יותר מערך אחד, בניגוד למילת המפתח return. משתנה out חייב להיות מאותחל בפונקציה עצמה. הערך של המשתנה לא מתקבל מבחוץ. לדוגמה:

```
static void Main(string[] args)
{
    int x=7;
    string name;
    OutFunc(2, out x, out name);
    Console.WriteLine(x + " " + name);
}

static void OutFunc(int x, out int num, out string str)
{
    num = 3;
    num += x;
    str = "gali";
}
```



- מחרוזת היא סדרה של תווים:

h	e	l	l	o	!
---	---	---	---	---	---

- כל תו הוא אות בודדת, ספרה או סמל

# יצירת מחרוזת

- כבר יצרנו מחרוזות:

```
string name = "Jack Robinson"  
string address = "London"  
string phone = "+44 (1) 3344222"
```

- כל אחד מהמשתנים הללו הוא הפנייה למחרוזת.

– כמה תווים כל מחרוזת מכילה?

– מה התוצאה של? `Console.WriteLine(phone)`

# פעולות של מחרוזת

- האופרטור + משמש עבור שרשור 2 מחרוזות:

```
first_name = "Jack"  
last_name = "Robinson"  
full_name = first_name + " " + last_name  
# full_name is "Jack Robinson"
```



# גישתה לתו בודד

- תווים במחרוזת מסודרים ומאונדקסים.

– האינדקס של התו הראשון הוא 0

– האינדקס של תו אחרון במחרוזת באורך  $n$  תווים הוא  $(n-1)$

- כדי לקרוא את התו באינדקס 1, אנו נשתמש ב[] (לקריאה בלבד!)

```
string name = "Jack Robinson";  
char letter = name[1];  
name[1] = 'A'; //ERROR - read only
```

– מהו הערך של letter ?

# פונקציות של מחרוזת

- Length :

– זוהי תכונה ולא פונקציה.

```
name = "Jack Robinson";  
int len = name.Length;  
Console.WriteLine(len); //13
```

- Substring( ) :

```
name = "Jack Robinson";  
string lastName = name.Substring(5);  
Console.WriteLine(lastName); // "Robinson"  
Console.WriteLine(name.Substring(5,3)); // "Rob"
```

- התוצאה של חיתוך מחרוזת הינה מחרוזת

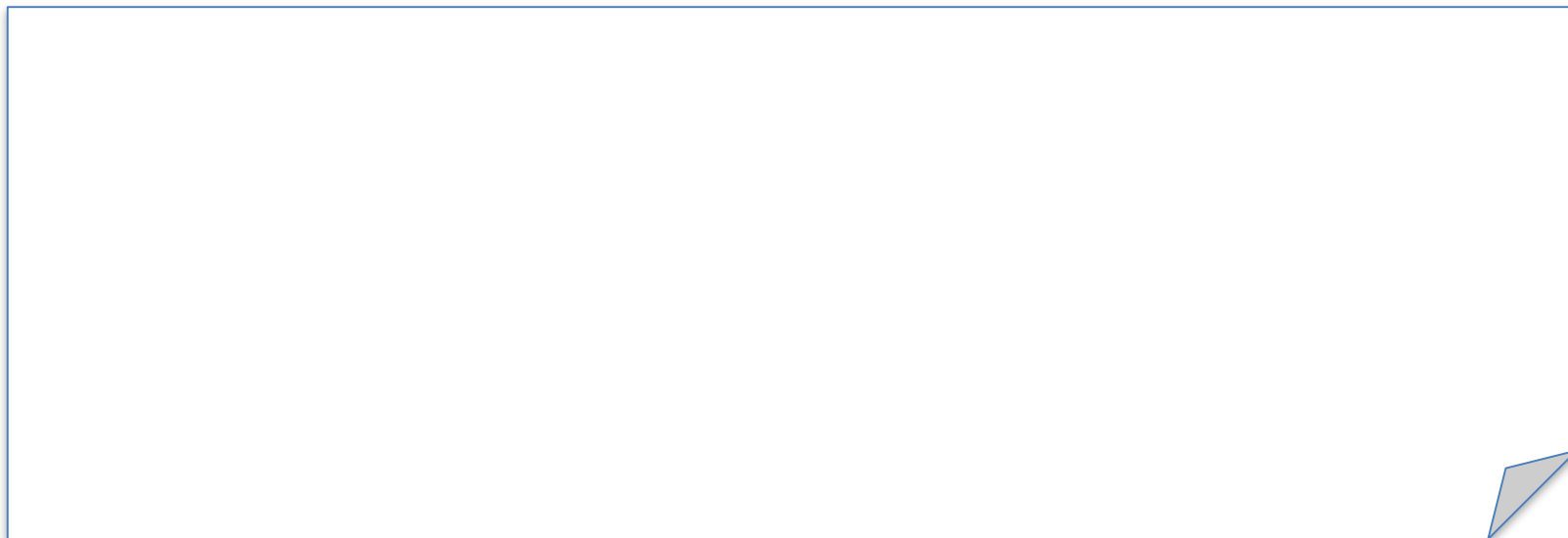
# מעבר על מחרזת

- ניתן לעבור על המחרוזת תו-תו באמצעות אינדקס:

```
name = "Jack Robinson";  
for (int i = 0; i < name.Length; i++)  
{  
    Console.WriteLine(name[i]);  
}
```

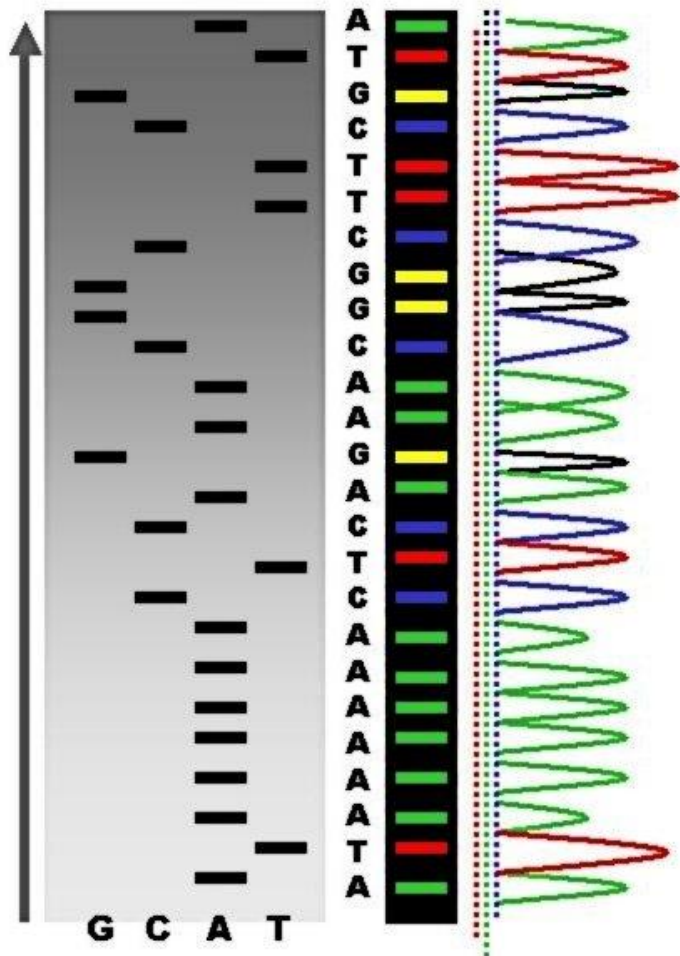
# תרגיל 1

- כתבו את הפונקציה  
`int Count_letter(string sentence, char letter)`
- הפונקציה מקבלת משפט ואות
- היא מחזירה את מספר הפעמים שהאות מופיעה במשפט
- לדוגמה:  
`Count_letter("hello, world", 'l')` תחזיר 3
- ראשית, בואו נגדיר את האלגוריתם במילים.



# דוגמה: ניתוח DNA

- רצפי DNA מוצגים כסדרה של אותיות בלטינית.



- כל אות מייצגת חומר ביולוגי, ורצף של אותם חומרים מייצג גנים ומאפיינים שונים.

- משימה חשובה היא למצוא גנים ודפוסים אחרים ברצף DNA.

## תרגיל 2

- הפונקציה `bool Find_sub_sequence(string sequence, string sub)` מקבלת רצף DNA ואותיות לחיפוש בתוך ה-DNA.
- הפונקציה תחזיר אמת אם האותיות נמצאות ב-DNA ושקר אחרת.
- נתחיל על ידי תיאור האלגוריתם במילים.

# שיטות\פונקציות של מחרוזת

- מחרוזות מציעות מגוון שיטות לעיבוד טקסט
- נלמד חלק מהן בהמשך הקורס
- אתם יכולים ללמוד אותן בעצמכם דרך האינטרנט או בספרים
- למשל: `ToUpper()`

```
name = "Jack Robinson";  
string s = name.ToUpper();  
Console.WriteLine(s);  
// will display JACK ROBINSON on screen
```

`ToUpper()` , כמו כל שיטה של מחרוזת, לא משנה את המחרוזת! (מדוע?)

# עוד שיטות למחרוזות

- עוד שיטה שימושית למחרוזת היא: `IndexOf ( . . . )`

```
name = "Jack Robinson";  
Console.WriteLine(name.IndexOf("ck"));  
// will display 2 on screen
```

- מה אם המחרוזת שמחפשים מופיעה יותר מפעם אחת?

```
name = "The elephant phoned from his cellphone";  
Console.WriteLine(name.IndexOf("phone")); //prints 13  
Console.WriteLine(name.IndexOf("phone", 14)); //prints 33
```

- מחזירה -1 אם לא הצליחה למצוא את המחרוזת



# השוואת מחרוזות

- ניתן להשוות בין מחרוזות באמצעות אופרטור == או !=

```
if (name == "Jack Robinson")  
    Console.WriteLine("Hello Jack!");  
else  
    Console.WriteLine("You're not Jack!");
```

- לא אפשרי להשתמש באופרטורים השוואתיים הבאים על מחרוזות: <, >, <=, >= (אבל על תווים כן!)
- בC#, אותיות גדולות הן בעלות ערך מספרים קטן יותר מאשר אותיות קטנות:

```
Console.WriteLine('A' < 'a');//True
```

# תרגיל 3

- כתבו את הפונקציה `string Reverse(string word)`
- בהינתן מחרוזת – החזירו את אותה המחרוזת, רק הפוך.
  - דוגמא `Reverse("hello") = "olleh"`
- ראשית, בואו נגדיר את האלגוריתם במילים.

## תרגיל 4 - פאלינדרום

- פאלינדרום הוא מילה, ביטוי, מספר או רצף כלשהו אשר ייראה אותו הדבר מכל כיוון קריאה.  
דוגמה: ראדאר, rotor, ....step on no pets

