מבוא לתכנות

הרצאה 1 – מבוא, קלט/פלט, משתנים ופעולות חשבון

סמסטר 1

מידע על הקורס

- הרצאות
- שיעורי בית – 10% מהציון הסופי
  - אחת לשבועיים\חודש
- שעות תרגול במעבדה
- אתר הקורס + פורום
- בחינה 90% מהציון הסופי
- 80% חובת נוכחות
- מגיעים באיחור מחכים 15 דקות בחוץ ונכנסים ביחד.
- אסור להוציא טלפון סלולרי מהתיק!!!

שיעורי בית

- מטלות כתיבת קוד בC#
- חשוב לתרגל ולקיים את המתודולוגיות והסטנדרטיזציה של כתיבת הקוד שנלמדו.

אתר הקורס

- כל החומרים נמצאים באתר הקורס.
- מייל המרצה: ruppincs@gmail.com
- מקורות מידע מרכזיים:
  - הרצאות והמצגות שלהן
  - שיעורי בית וביצועם.
  - הפורום
  - Stackoverflow.com

תכנות

- מהי תוכנת מחשב?
- כיצד ניתן ליצור תוכנת מחשב?
- מהם האתגרים של התהליך?

תוכנת מחשב

- תוכנת מחשב היא רצף הוראות למחשב.

```
lw      $1, $2(80)
add     $1, $1, $1
sw      $1, $2(80)
lw      $3, $2(40)
mul     $4, $1, $3
divi    $4, 2
sw      $4, $2(40)
jeq     $6, $1, $4
lw      $1, $2(80)
add     $1, $4, $1
sw      $1, $2(80)
```

```
int foo(int x) {
 int y;
 x = x + 1;
 y = (x * z) / 2;
 if (y == 0)
    return x;
 x = x + y;
 return x;
}
```

```
(define (foo x)
 (let
   ((a (+ x 1))
    (y (/ (* x 2)
          2)))
  (if (= y 0) x
      (+ x y))))
```

שפות תכנות

- שפת תכנות היא מכלול הוראות מוגדרות ומוכרות עבור המחשב, ותחביר שמגדיר כיצד להשתמש בהן.

- ישנם 2 סוגים של שפות תכנות:
  - שפות עיליות: #C, Java, C, ++C, Python....
  - שפות סף: שפת Assembly

שפות תכנות

- מחשב מבין שפות סף בלבד. לכל סוג מחשב יש שפת סף משלו.
- *שפות עיליות* ניתנות לתרגום ל*שפות סף*.
- תהליך התרגום יכול להתבצע מראש, או רק כאשר מבצעים את ההוראות העיליות.

שפות תכנות

- <u>קומפילציה:</u>
תהליך תרגום כלל התוכנית הכתובה בשפה עילית, אל תוכנית הכתובה בשפה סף.

- <u>מפרש :</u>
תהליך ביצוע תוכנית שכתובה בשפה עילית באמצעות תרגום כל הוראה לשפת סף בעת ביצועה.

קומפילציה

תוכנית
להרצה

קוד סף ב0/1

קומפיילר

תוכנית עילית

תוכנית
להרצה

מבצע

מפרש

תוכנית
ברמה גבוהה

מפרש

C#

- בקורס זה נלמד את שפת התכנות C#
- C# היא שפת תכנות עילית
- היא חזקה, אך במקביל קלה לשימוש ולתכנות
- כדי להריץ תוכנית C#, נשתמש בקומפיילר של C#, ואז נריץ את התוכנית באמצעות המבצע (נערכת ההפעלה) של המחשב

התוכנית הראשונה שלנו

- בואו נכתוב את התוכנית הראשונה שלנו!

- התוכנית הראשונה שלנו תציג את המילים הללו על המסך:

Hello, World!

התוכנית הראשונה שלנו

- זה קל מאוד לכתוב תוכנית שכזו בשפת #C:

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

```
Hello, World!
```

הרצת התוכנית שלנו

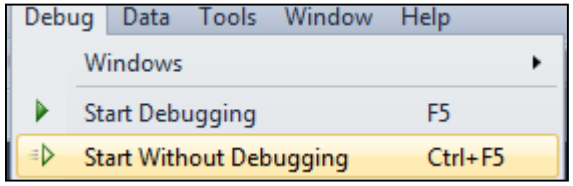
- ראשית נפתח את סביבת העבודה של VS ונפתח פרויקט מסוג 'Console Application'.
ניתן לו שם, נחליט על המיקום שלו (היכן הקבצים שניצור יישמרו) ונמתין לפתיחתו.

- נכניס את הקוד שלנו בתוך השורה הבאה:

```
static void Main(string[] args)
{
    //insert your Code here!
}
```

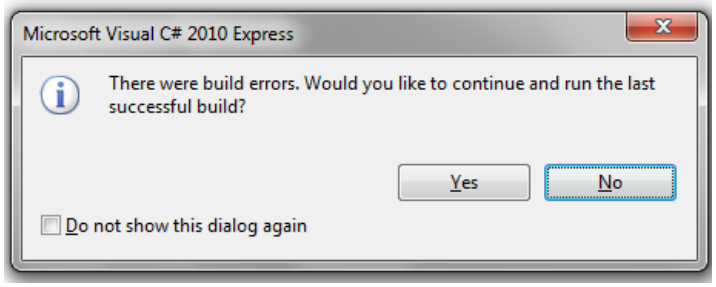
הרצת התוכנית שלנו

- נוכל להריץ את התוכנית שלנו תוך שימוש בקומפיילר של C# באמצעות 3 דרכים:
  - שמירת הקוד המקומפל בתור file.exe ולהריץ אותו.
  - לחיצה על Ctrl+F5 בVS
  - או



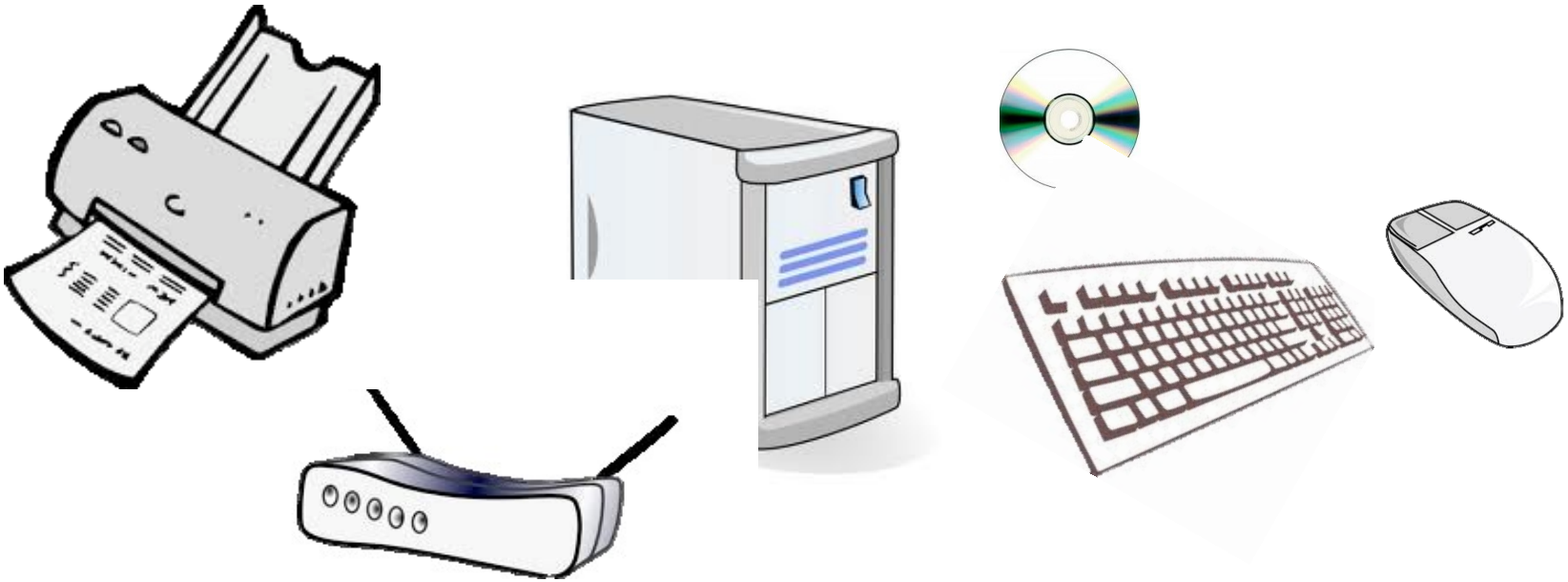
הרצת התוכנית שלנו

- נוכל לשנות את התוכנית ולהריצה שוב באמצעות Ctrl + F5 למשל כדי לראות את התוצאות החדשות.
- שימו לב שאם ישנה טעות כלשהי בקוד, הקומפיילר יודיע על כך:



- במקרה זה יש ללחוץ על "לא" והתהליך יופסק. לחיצה על "כן" משמעותה הוראה למחשב להריץ את התוכנית בגירסתה הקודמת טרם השינויים שנוספו!

ערכים וסוגים



- הנתונים שהתוכנית שלנו עובדת איתם יכולים להיות מסוגים רבים. לדוגמה:
  - טקסט (דוגמת "Hello, World!")
  - מספרים (4, 7, -10, 3.25)
- אנו קוראים לזה הסוג של הנתון.

ערכים וסוגים

- הסוגים הבסיסיים בשפת C# הינם:
  - מחרוזת – רצף של תווים. מוגדר כ string.
    לדוגמה: "3", "a", "A", "Hello", "Hello, World!"
  - מספר שלם – ללא נקודה עשרונית. מוגדר כ int.
    דוגמאות: 0 ,3222- ,45- ,967384 ,10 ,1
  - מספר ממשי – כולל נקודה עשרונית. מוגדר כ double/float (עבור float צריך "f").
    דוגמאות: 7d ,0.0f ,31.0 ,2.71f- ,3.1415
  - תו – תו אחד ויחיד. מוגדר כ char.
    דוגמאות: '7' ,'@' ,'A' ,'a'
  - בוליאני – אמת או שקר. מוגדר כ bool.
    דוגמאות: true, false.

משתנים

- בדרך כלל כשאנו יוצרים תוכנית, אנו נרצה לשמור ערך עבור שימוש עתידי.
- אנו יכולים לשמור ערכים על ידי כך שניתן להם שמות, ואז נשתמש בשמות אלו בהמשך כדי למשוך את הערך ששמרנו.
- השם הזה נקרא *משתנה* – מכיוון שלאותו שם יכול להכיל ערכים שונים בריצה של תוכנית אחת או בריצות שונות.
- משתנה הוא תא זיכרון במחשב שניתן לתת לו שם. ניתן להכניס אליו מידע ולקרוא את המידע.

משתנים

- כדי ליצור משתנה פשוט צריך לרשום את ה**סוג** שלו ואת ה**שם** הרצוי:

```
int x;
```

פקודות מסתיימות ב ;

- כעת נוכל לשים בו ערך: `x=7;`

שם את צד ימין בתוך צד שמאל

- נוכל לעשות זאת בפקודה אחת: `int x=7;`
- כעת נוכל להשתמש במשתנה זה על ידי קריאה בשמו:

```
int x = 7;
Console.WriteLine(x);
int y;
y = x + 3;
```

משתנים

- בינואר, מחיר הדלק היה 7.1 ש"ח
- בפברואר, מחיר הדלק עלה ב8%.
- במרץ, ירד ב5%.
- מה היה מחיר הדלק במרץ?

```
double jan = 7.1;
double feb = jan * (1 + 0.08);
double mar = feb * (1 - 0.05);
Console.WriteLine(mar);
```

משתנים

- משתנה יכול להכיל כל *סוג* של מידע. לדוגמה:

שרשור מחרוזות

```
string name = "John";
string welcome_message = "Hello " + name;
string goodbye_message = "Goodbye " + name;
Console.WriteLine(welcome_message);
Console.WriteLine(goodbye_message);
```

- מהם שמות המשתנים כאן?

משתנים

- שם המשתנה יכול להיות כל מה שתרצו, אך ישנם מספר חוקים:
  - חייב להתחיל עם אות או _ (עדיפות לאות קטנה)
  - יכול להכיל גם ספרות
  - לא יכול להיות <u>מילה שמורה</u>
  - רגיש לאותיות קטנות / גדולות
  - צריך להיות בעל משמעות!

אופרטורים ואופרנדים

- אופרטור הוא סמל אשר מייצג פעולה כלשהי (חישובית) על ערך אחד או יותר.
- לדוגמה, האופרטורים +,-,*,/ מבצעים את ארבעת הפעולות המתמטיות הפשוטות על 2 מספרים:
  - 1 + 1
  - 2 * x
- הערכים לצידי האופרטורים נקראים <u>*אופרנדים*</u>. הם יכולים להיות קבועים או משתנים.

אופרטור /

- אופרטור ה / מבצע חילוק בין 2 מספרים.
- אך יש לשים לב להתנהגות הבאה:
  - כאשר מבוצע על 2 מספרים שלמים, התוצאה תהיה שלמה:
    10 / 2 = 5
    15 / 2 = 7
    (השארית נמחקת ולא מתעגלת!)
  - כאשר מבוצע על מספר עשרוני ועוד מספר (שלם או עשרוני), התוצאה תהיה מספר עשרוני:
    10.0 / 2 = 5.0
    15 / 2.0 = 7.5
- אנו משתמשים בנקודה עשרונית כדי לקבוע שמספר הוא עשרוני:
  - 5 is an integer, 5.0 is a double, 5f is a float

ביטויים

- ביטוי הוא שילוב בעל משמעות של ערכים ואופרטורים
- לביטוי יש ערך משלו, אשר מהווה את התוצאה של <u>*הפעלת*</u> <u>*האופרטור*</u> על <u>*האופרנדים*</u>:
  - 3 * 4 (ערך: 12)
  - 7 + x (ערך: תלוי מהו x)

ביטויים

- ביטוי יכול להיות מורכב יותר:
  - 2 + 3 + 4 – 1
  - 4 * (8 + 13)
  - ”Hello” + “,” + “ World!”
  - 2 * 4 + 6 / 3
- מהם הערכים של הביטויים הנ"ל?

סדר פעולות

- מספר ביטויים יכולים להיות מבלבלים כאשר מנסים להבין מהו ערכם:

3 * 4 + 6 / 2

- מהו הערך של הביטוי הנ"ל?

סדר פעולות

- הקומפיילר של C# מודע לסדר הפעולות:
  - סוגריים: ()
  - חזקה: Math.Pow(2,3) -> 2^3
  - כפל וחילוק: * /
  - חיבור וחיסור: + -

התוכנית השנייה שלנו

- כעת, בואו ניצור תוכנית שנייה.
  להלן התכנון שלה:
  - להציג הודעה: "הכנס את שמך"
  - נקרא את השם שהמשתמש מכניס
  - נציג את ההודעה: "שלום " + השם שהוכנס

אלגוריתם

- אלגוריתם היא דרך נפוצה לתאר את *הזרם* של התוכנית שאנו מתכננים
- היא מציינת את ה*צעדים* שאנו צריכים לנקוט כדי לגרום לתוכנית לעשות מה שאנו רוצים
- התוכנית אשר מקודדת בעקבות האלגוריתם נקראת ה*מימוש* של האלגוריתם

התוכנית השנייה שלנו

- הצעד הראשון באלגוריתם שלנו הוא:
  - להציג הודעה: "הכנס את שמך"
- אנו יודעים כיצד לממש את הצעד הזה:

```
Console.WriteLine("Enter your name:");
```

- הצעד הבא באלגוריתם הוא:
  - לקרוא את השם שהמשתמש מכניס
- כיצד תממשו צעד זה?

התוכנית השנייה שלנו

- נוכל להשתמש בתחביר הבא על מנת לקרוא קלט מהמשתמש עד שהוא ילחץ על מקש הEnter:

```
string <variable> = Console.ReadLine();
```

כאשר <variable> הוא שם המשתנה שיש לאחסן את הערך בו.
פקודת ()ReadLine תמיד תחזיר מחרוזת!

- אז הקוד שלנו יהיה:

```
string name = Console.ReadLine();
```

כך המשתנה *name* יכיל את הערך שיוכנס על ידי המשתמש.

התוכנית השנייה שלנו

- הצעד האחרון באלגוריתם שלנו הוא:
  - הצגת ההודעה: "שלום " + השם שהוכנס
- זהו מהלך שגם אותו אנו יודעים לבצע:

```
Console.WriteLine("Welcome " + name);
```

התוכנית השנייה שלנו

- סיימנו את התוכנית השנייה שלנו!
  להלן הקוד המלא שלה:

```
Console.WriteLine("Enter your name:");
string name = Console.ReadLine();
Console.WriteLine("Welcome " + name);
```

- כדי להריץ אותה, נשמור אותה בתור file.cs,
  נקמפל ונפעיל אותה.

הערה

- הערה היא מחרוזת אשר לא מבוצעת. היא מתחילה עם '//' ונגמרת באותה שורה, או משתמשת ב */ ו /* עבור מספר שורות.

```
int x = 7;
Console.WriteLine(x);
int y;
y = x + 3; //this will add 3 to x
/*this
is
a
comment*/
```

תווים מיוחדים

- ישנם תווים מיוחדים בהם משתמשים במחרוזות:
  - \n : ירידת שורה
  - \t : ביצוע רווח Tab
  - \b : תזוזת תו אחד לאחור
  - \\ : הדפסת תו \
  - \” : הדפסת תו "
  - ...

מסמני מקום

- יש דרך אחרת להדפסה למסך. בדרך זו נוכל להיעזר בפורמט אלגנטי ונוח לקריאה.
כאן אנו נעזרים במסמני מקום כדי "לשמור על המקום" שבו יופיעו הערכים.

```
Console.WriteLine("the second marker is {1},
the first marker is {0}\n", 1,2);
//will print “the second marker is 2, the
first marker is 1”
Console.WriteLine("real number x.xx:{0:F2}\n
", 1.2345);
Will print “real number x.xx:1.23”
```

מסמני מקום 2

- יש שיטה נוספת לסימון מקום. שימוש ב$ לפני המחרוזת ושימוש ב "...{בטוי בקוד}..."

```
int num1=1, num2=2;
Console.WriteLine($"the second variable is
{num2}, the first variable is {num1}\n");
//will print “the second variable is 2, the
first variable is 1”
```

המרת קלט

- כפי שנאמר, פקודת ה()ReadLine מחזירה תוצאת מחרוזת בלבד. אז מה קורה כאשר אנו מעוניינים במספר, למשל מספר שלם int?

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

- ומה בנוגע למספר עשרוני double?

```
double heigth = double.Parse(Console.ReadLine());
```

Appendix a

C# Type	.Net Framework (System) type	Signed?	Bytes Occupied	Possible Values
sbyte	System.Sbyte	Yes	1	-128 to 127
short	System.Int16	Yes	2	-32768 to 32767
int	System.Int32	Yes	4	-2147483648 to 2147483647
long	System.Int64	Yes	8	-9223372036854775808 to 9223372036854775807
byte	System.Byte	No	1	0 to 255
ushort	System.Uint16	No	2	0 to 65535
uint	System.UInt32	No	4	0 to 4294967295
ulong	System.Uint64	No	8	0 to 18446744073709551615
float	System.Single	Yes	4	Approximately $\pm 1.5 \times 10^{-45}$ to $\pm 3.4 \times 10^{38}$ with 7 significant figures
double	System.Double	Yes	8	Approximately $\pm 5.0 \times 10^{-324}$ to $\pm 1.7 \times 10^{308}$ with 15 or 16 significant figures
decimal	System.Decimal	Yes	12	Approximately $\pm 1.0 \times 10^{-28}$ to $\pm 7.9 \times 10^{28}$ with 28 or 29 significant figures
char	System.Char	N/A	2	Any Unicode character (16 bit)
bool	System.Boolean	N/A	1 / 2	true or false

Appendix b

Character	Escape Sequence
'	\'
"	\"
\	\\
Alert	\a
Backspace	\b
Form feed	\f
New Line	\n
Carriage Return	\r
Horizontal Tab	\t
Vertical Tab	\v
A unicode character specified by its number e.g. \u200	\u
A unicode character specified by its hexidecimal code e.g. \xc8	\x
null	\0 (zero)