

מבוא למדעי המחשב

מצגת 2 : משתנים, אופרטורים

תכנית שנייה: המרת נתונים

```
/* program converts the distance of a marathon:  
26 miles and 385 yards to kilometers */  
#include <stdio.h>  
void main()  
{  
    printf ("A marathon is %lf kilometers\n", 1.609*(26 + 385/1760.0));  
}
```

תכנית שנייה: המרת נתונים

/* program converts the distance of a marathon
26 miles and 385 yards to kilometers */

#include <stdio.h>

void main()

{

int miles, yards;

double kilometers;

miles = 26;

yards = 385;

kilometers = 1.609 * (miles + yards / 1760.0);

printf ("A marathon is %lf kilometers\n", kilometers);

}

תכנית שנייה: המרת נתונים

```
/* program converts the distance of a marathon  
26 miles and 385 yards to kilometers */
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

סימן ';' בסוף כל פקודה (כדי שהקומפיילר ידע היכן מסתיימת הפקודה)

```
int miles, yards;
```

```
double kilometers;
```

שורה ריקה, לשיפור הקריאות של הקוד

```
miles = 26;
```

```
yards = 385;
```

```
kilometers = 1.609 * (miles + yards / 1760.0);
```

```
printf ("A marathon is %lf kilometers\n", kilometers);
```

```
}
```

תכנית שנייה: המרת נתונים

```
/* program converts the distance of a marathon  
26 miles and 385 yards to kilometers */
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int miles, yards;
```

```
    double kilometers;
```

הצהרת משתנים

משתנה הוא תא זכרון שיש לו שם וטיפוס

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf ("A marathon is %lf kilometers\n", kilometers);
```

```
}
```

- הקצאת **מקומות בזיכרון**, כדי שהתכנית תוכל לשמור בהם ערכים, לשנות אותם ולקרוא מהם את הערכים שאוחסנו בהם.
- לכל מקום כזה נותנים **שם**, כדי שהתכנית תוכל להתייחס אליו.
- שם של משתנה מורכב מאותיות, ספרות וקו תחתון. התו הראשון לא יכול להיות ספרה. **רצוי לתת שם בעל משמעות.**

הצהרת משתנים

- הקצאת מקומות בזיכרון, כדי שהתכנית תוכל לשמור בהם ערכים, לשנות אותם ולקרוא מהם את הערכים שאוחסנו בהם.
- לכל מקום כזה נותנים שם, כדי שהתכנית תוכל להתייחס אליו.
- שם של משתנה מורכב מאותיות, ספרות וקו תחתון. התו הראשון לא יכול להיות ספרה. רצוי לתת שם בעל משמעות.
- שם משתנה לא יכול להיות מילה שמורה: לדוגמא, ~~int int;~~
- יש הבדל בין אותיות גדולות וקטנות: לדוגמא, $A \neq a$
- ההצהרה מציינת טיפוס (סוג) של משתנה.
- למשל, miles הוא משתנה מטיפוס `int` (מספר שלם = integer).
- גודל הזיכרון לצורך שמירת הערך של המשתנה תלוי בסוג שלו.

תכנית שנייה: תמונת הזכרון

/* program converts the distance of a marathon
26 miles and 385 yards to kilometers */

#include <stdio.h>

void main()

{

int miles, yards;

double kilometers;

miles = 26;

yards = 385;

kilometers = 1.609*(miles+yards/1760.0);

printf ("A marathon is %lf kilometers\n", kilometers);

}

miles yards kilometers

?

?

?

תכנית שנייה: המרה נתונים

```
/* program converts the distance of a marathon  
26 miles and 385 yards to kilometers */
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int miles, yards;
```

```
    double kilometers;
```

השמה

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609*(miles+yards/1760.0);
```

השמה

```
    printf ("A marathon is %lf kilometers\n", kilometers);
```

```
}
```

הערה: בשפת C אין משמעות להשמה הבאה:

```
26 = miles;
```

תכנית שנייה: תמונת הזכרון

/* program converts the distance of a marathon
26 miles and 385 yards to kilometers */

#include <stdio.h>

void main()

{

int miles, yards;

double kilometers;

miles = 26;

yards = 385;

kilometers = 1.609*(miles+yards/1760.0);

printf ("A marathon is %lf kilometers\n", kilometers);

}

miles	yards	kilometers
26	?	?

תכנית שנייה: תמונת הזכרון

```
/* program converts the distance of a marathon  
26 miles and 385 yards to kilometers */
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int miles, yards;
```

```
    double kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609*(miles+yards/1760.0);
```

```
    printf ("A marathon is %lf kilometers\n", kilometers);
```

```
}
```

miles

yards

kilometers

26

385

?

תכנית שנייה: תמונת הזכרון

```
/* program converts the distance of a marathon  
26 miles and 385 yards to kilometers */
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int miles, yards;
```

```
    double kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards/1760.0);
```

```
    printf ("A marathon is %lf kilometers\n", kilometers);
```

```
}
```

miles

yards

kilometers

26

385

42.185969

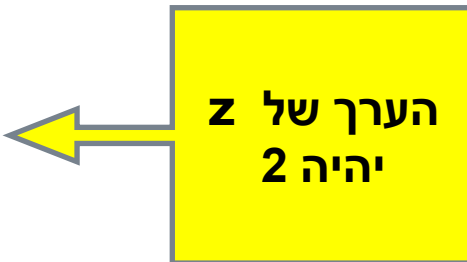
פעולות חשבון בסיסיות

■ $+$, $-$, $*$, $/$: הפעולות האריתמטיות הבסיסיות.

פעולות חשבון בסיסיות

- **+ , - , * , /** : הפעולות האריתמטיות הבסיסיות.
- אם פעולה אריתמטית מבוצעת על שני ערכים מטיפוס מסויים, התוצאה תהיה מאותו הטיפוס.
- זה נכון גם לפעולת **חילוק** על שני שלמים!!
התוצאה תהיה מספר שלם... למשל:

```
void main()  
{  
    int x, y, z;  
    x=7;  
    y=3;  
    z=x/y;  
}
```



פעולות חשבון בסיסיות: חלוקה ושארית בשלמים

- התוצאה של חלוקה בשלמים היא הערך השלם של הערך האמיתי.
- **%** : שארית החלוקה "מודולו" (modulo)
- (פעולה זו מוגדרת אך ורק עבור מספרים שלמים!)

למשל:

$6 \% 3 = 0$,	$6 / 3 = 2$	■
$7 \% 3 = 1$,	$7 / 3 = 2$	■
$8 \% 3 = 2$,	$8 / 3 = 2$	■

תכנית שנייה: המרה נתונים

```
/* program converts the distance of a marathon  
26 miles and 385 yards to kilometers */
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    printf ("A marathon is %lf kilometers\n", 1.609*(26 + 385/1760.0));
```

```
}
```

אם פעולה אריתמטית מבוצעת על שני ערכים, אחד מטיפוס שלם ואחד מטיפוס ממשי, התוצאה תהיה על פי הטיפוס הממשי.

לדוגמא: $7 / 2.0 == 3.5$

ניתן גם לבקש המרה בין טיפוסים באופן מפורש:

לדוגמא: $7 / (\text{double}) 2 == 3.5$

סדר קדימות של פעולות

"פעולות" נקראות גם "אופרטורים"

- **סדר הקדימות של חישוב אופרטורים אריתמטיים**
(כלומר, באיזה סדר מבצעים את הפעולות בביטויים מורכבים), הוא כפי שמוגדר במתמטיקה. כלומר:

- סוגריים $()$
- כפל $*$, חילוק $/$, מודולו $\%$
- חיבור $+$, חיסור $-$
- השמה $=$

תכנית שנייה: המרה נתונים

```
/* program converts the distance of a marathon  
26 miles and 385 yards to kilometers */
```

```
#include <stdio.h>
```

```
void main()
```

```
{
```

```
    int miles, yards;
```

```
    double kilometers;
```

```
    miles = 26;
```

```
    yards = 385;
```

```
    kilometers = 1.609 * (miles + yards / 1760.0);
```

```
    printf ("A marathon is %lf kilometers\n", kilometers);
```

```
}
```

שימו לב לסדר הקדימות בביטוי: +, /, *, ()
למשל, אופרטור החילוק יתבצע לפני החיבור



תכנית שנייה - פלט

```
printf("A marathon is %lf kilometers\n", kilometers);
```

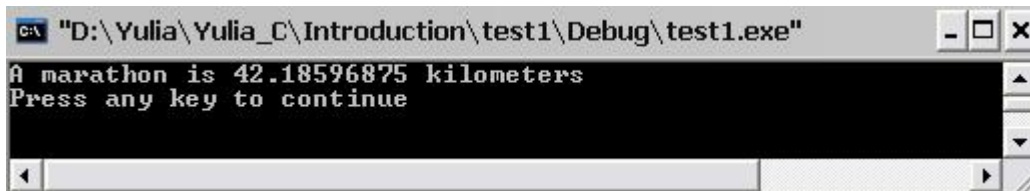
- ניתן לכלול בהדפסה של printf תוכן של משתנים.
- לצורך כך יש לציין:
- את המיקום בטקסט שבו יודפס הערך של המשתנה.
- מהו סוג המשתנה.
- בדוגמא הנ"ל:
- **%lf** מציין שיש להדפיס **במקומו** ערך של משתנה אשר מכיל מספר ממשי.
- שם המשתנה הזה, **kilometers**, מציין בהמשך, מייד לאחר המחרוזת המיועדת להדפסה.

`printf("A marathon is %lf kilometers\n", kilometers);`



```
C:\ "D:\Yulia\Yulia_C\Introduction\test1\Debug\test1.exe"
A marathon is 42.185969 kilometers
Press any key to continue
```

`printf("A marathon is %.8lf kilometers\n", kilometers);`



```
C:\ "D:\Yulia\Yulia_C\Introduction\test1\Debug\test1.exe"
A marathon is 42.18596875 kilometers
Press any key to continue
```


תכנית שנייה: תמונת הזכרון

/* program converts the distance of a marathon
26 miles and 385 yards to kilometers */

#include <stdio.h>

void main()

{

int miles, yards;

double kilometers;

miles = 26;

yards = 385;

kilometers = 1.609*(miles+yards/1760.0);

printf ("A marathon is %lf kilometers\n", kilometers);

}

miles

yards

kilometers

26

385

42.185969

תכנית שנייה: תמונת הזכרון "במציאות"

/* program converts the distance of a marathon
26 miles and 385 yards to kilometers */

#include <stdio.h>

void main()

{

int miles, yards;

double kilometers;

miles = 26;

yards = 385;

kilometers = 1.609*(miles+yards/1760.0);

printf ("A marathon is %lf kilometers\n", kilometers);

}

miles

yards

kilometers

11010

110000001

.....

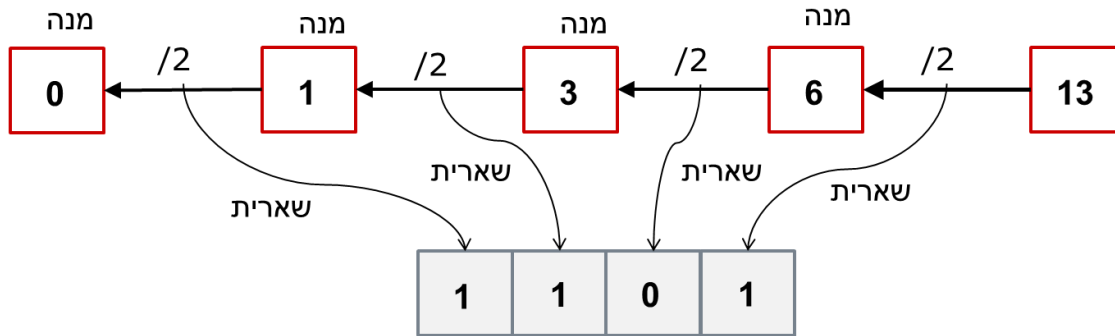
/* 26 = $2^4 + 2^3 + 2^1$ */

/* 385 = $2^8 + 2^7 + 2^0$ */

Binary system ספירה בבסיס 2 (בינארי)

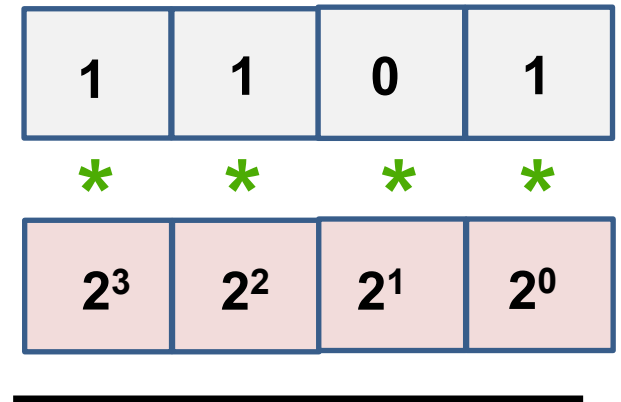
Decimal value	Binary value
0	0
1	1
2	10
3	11
4	100
.....
9	1001
10	1010

המרה ממספר עשרוני למספר בינארי



ניתן למצוא כך את הייצוג הבינארי של כל מספר עשרוני טבעי

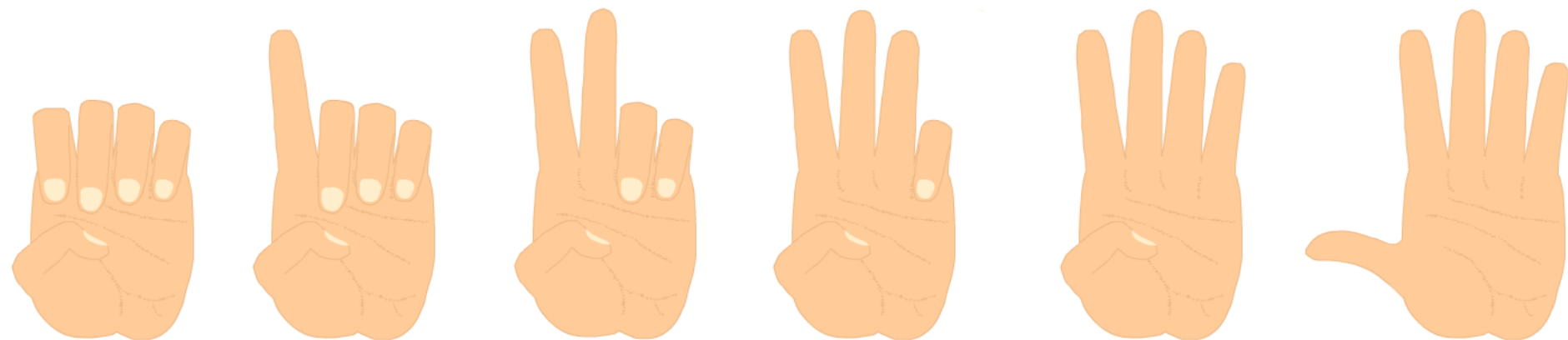
המרה ממספר בינארי למספר עשרוני

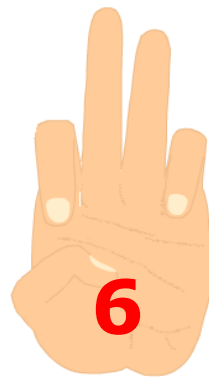
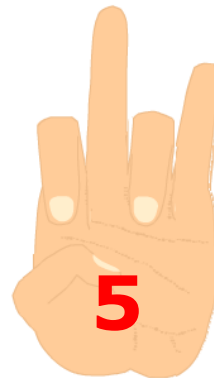


$$\begin{aligned}
 &1 * 2^0 + 0 * 2^1 + 1 * 2^2 + 1 * 2^3 \\
 &= 1 + 0 + 4 + 8 \\
 &= 13
 \end{aligned}$$

חידה - לספור על כף יד אחת

■ עד איזה מספר אפשר לספור על כף יד אחת??





וכן הלאה...
עד?

ייצוג מספרים בבסיס כלשהו

- כשרוצים לציין שיש להתייחס לרצף ספרות כאל מספר בבסיס מסויים, כותבים את הבסיס בכתב תחת. למשל:
 10_2 הוא מספר בינארי ואילו 10_{10} הוא מספר דצימאלי.

תלוי באיזה
בסיס...

כמה זה 10?

$$10_2 = 0 \cdot 2^0 + 1 \cdot 2^1 = 0 + 2 = 2_{10}$$

יש 10 סוגי
אנשים:
אלו שמבינים
בינארית
ואלו שלא.

ייצוג מספרים בבסיס כלשהו - המשך

■ בהינתן מספר בבסיס r :

$$d_{n-1} \cdots d_0$$

■ הערך המספרי שהוא מייצג הוא:

$$d_0 \times r^0 + d_1 \times r^1 + \cdots + d_{n-1} \times r^{n-1}$$

בסיס מיוחד: 16 (Hexadecimal)

■ הספרות ההקסדצימליות:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.



■ ערך של מספר הקסדצימלי:

C=12

$$\begin{array}{r} 7 * 16^0 + C * 16^1 + 0 * 16^2 + 2 * 16^3 = \\ 7 \quad + 192 \quad + 0 \quad + 8192 \\ = 8391 \end{array}$$

2	0	C	7
*	*	*	*
16^3	16^2	16^1	16^0

דצימאלי	HEX	בינארי
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

דצימאלי	HEX	בינארי
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

$$10_2 = ?$$

$$10_{16} = ?$$

דצימאלי	HEX	בינארי
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
10	A	1010
11	B	1011
12	C	1100
13	D	1101
14	E	1110
15	F	1111

$$10_2 = 2_{10}$$

$$10_{16} = 16_{10}$$

ASCII code

characters	'a'	'b'	'c'	'z'
values	97	98	99		112
characters	'A'	'B'	'C'	'Z'
values	65	66	67		90
characters	'0'	'1'	'2'	...	'9'
values	48	49	50		57
characters	'&'	'*'	'+'		
values	38	42	43		

התווים רשומים בין גרשיים.
בפרט, '0' ≠ 0

התאמה בין int ל-char

int x=49; // the same as: x = '1'

char y='a'; // the same as: y = 97

printf ("x=%d y=%c", x, y);

x=49 y=a פלט:

printf ("x=%c y=%d", x, y);

x=1 y=97 פלט:

התאמה בין int ל-char

int x=49; // the same as: x = '1'

char y='a'; // the same as: y = 97

printf ("x_new=%d y_new=%d", x+1, y+1);

x_new=50 y_new=98 :פלט

x=49;

y='a';

printf ("x_new=%c y_new=%c", x+1, y+1);

x_new=2 y_new=b :פלט



טיפוסים בקורס שלנו

- אנו נשתמש בעיקר בשלושה טיפוסים:

`int`

– לייצוג מספרים שלמים נשמש בטיפוס:

`double`

– לייצוג מספרים ממשיים נשתמש בטיפוס:

`char`

– לייצוג הקוד של תווים נשתמש בטיפוס:



- There are four *standard signed integer* types:

char, short, int, long.

■ השוני בין הטיפוסים הוא "הגודל"

■ דוגמה להצהרה ראשונית:

- **int** miles, yards;
- **long** distance;
- **char** letter;

■ יחד עם הצהרה אפשר גם לאתחל ערכים ראשוניים:

- **int** miles=26, yards;

- There are three *real floating* types:
float, double, long double

■ דוגמה להצהרה ראשונית:

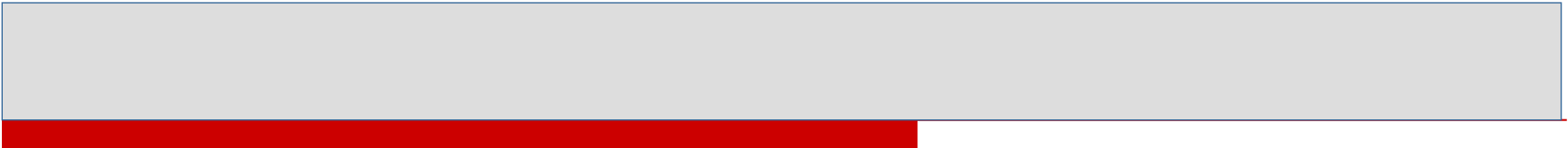
- **double** x, y = 3.6, min = -45.12345;

sizeof

```
printf("%d %d %d %d\n", sizeof(char), sizeof(short), sizeof(int), sizeof(long));
```

```
printf("%d %d %d\n", sizeof(float), sizeof(double), sizeof(long double));
```

// הגודל של כל טיפוס תלוי גם בסוג המחשב וגם בקומפילר



ייצוגי טיפוסים בפקודת הדפסה

int – %d ■ (בבסיס 10 כרגיל)

int – %x ■ (בבסיס 16)

char – %c ■

float – %f ■ (הצגה עם נקודה עשרונית 21.54)

double – %lf ■ (הצגה עם נקודה עשרונית 21.54)

```
int a=3, b=2;  
double x=3, y=2;  
printf("a/b = %d\n", a/b);  
printf("a/b = %lf\n", a/b);  
printf("a/b = %lf\n", (double)a/b);  
printf("x/y = %lf\n", x/y);  
printf("x/y = %d\n", x/y);
```

אם רוצים

שהמחשב יתייחס

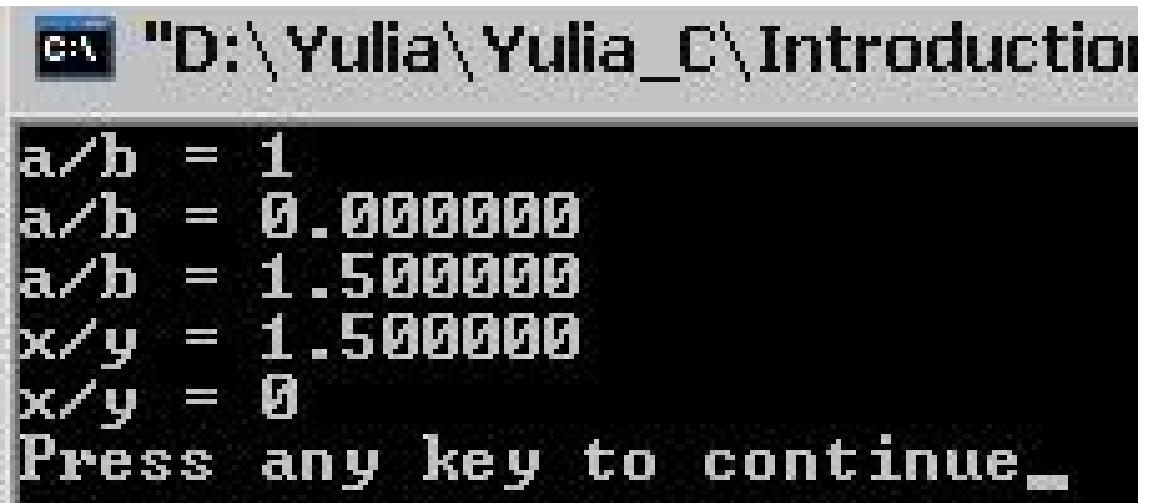
למשתנה באופן חד-

פעמי כבעל טיפוס

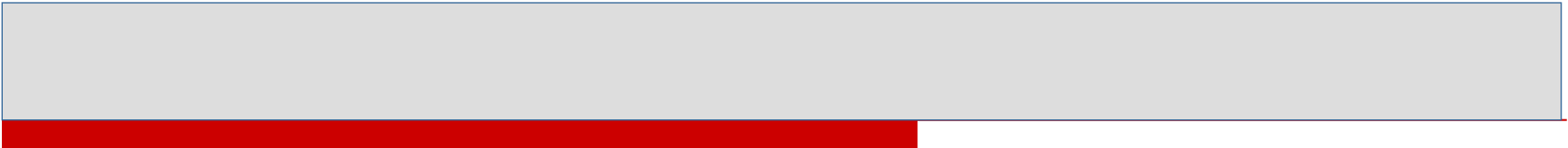
אחר, אז ניתן לציין

זאת במפורש. זה

נקרא **casting**.



```
ca\ "D:\Yulia\Yulia_C\Introduction  
a/b = 1  
a/b = 0.000000  
a/b = 1.500000  
x/y = 1.500000  
x/y = 0  
Press any key to continue_
```



- לא תמיד ידוע מראש מהם הערכים שאותם צריך לעבד.

- לצורך זה יש לקלוט ערכים כאלה מהמשתמש ולשמור במשתנים שהגדרנו.

- פונקציה ראשונה שמאפשרת לקלוט: `scanf()`

דוגמה 2: מחשבון לפעולת חיבור

- כתבו תכנית הקולטת שני מספרים שלמים, מחשבת ומדפיסה סכומם.

■ הצעה לפתרון:

- נקלוט ערכים שלמים למשתנים `num1` ו- `num2`
- נציב את סכומם במשתנה `sum` ב-
- נדפיס את `sum`

```
#include <stdio.h>
void main() {

    int  num1, num2, sum;

    printf("Enter an integer\n");
    scanf("%d", &num1);
    printf("Enter an integer\n");
    scanf("%d", &num2);

    sum = num1 + num2;

    printf("answer: %d + %d = %d\n", num1, num2, sum);
}
```

```
#include <stdio.h>
void main() {
```

```
    int  num1, num2, sum;
```

```
    printf("Enter an integer\n");
    scanf("%d", &num1);
    printf("Enter an integer\n");
    scanf("%d", &num2);
```

```
    sum = num1 + num2;
```

```
    printf("answer: %d + %d = %d\n", num1, num2, sum);
}
```

C:\WINDOWS\system32\cmd.exe

```
Enter an integer
35
Enter an integer
15
answer: 35 + 15 = 50
Press any key to continue . . .
```


הפונקציה scanf()

■ דוגמה:

- `scanf("%d", &a);`
 - `%d` – קוד מיוחד, מציין ל-C לקלוט מספר שלם (כמו בפלט)
 - סימן `&` מציין כתובת בזיכרון המחשב שבה נמצא המשתנה שישמור את הערך (בדוגמה- משתנה `a`)

ייצוג כללי לשימוש `scanf`:

`scanf ("ייצוג %", &משתנה);`



- **קוד אישי** למכונת צילום מוגדר כסכום של 2 ספרות אחרונות ושל 2 ספרות ראשונות של ת.ז.
- הנחה: הספרה המשמעותית ביותר בת"ז $0 < z$
- כתבו תכנית בשפת C אשר מקבלת ערך ת.ז. ומדפיסה קוד אישי לשימוש במכונת צילום.
- **דוגמה:** עבור ת.ז. 123456789
החישוב: $89 + 12$
והפלט: 101

- נקלוט למשתנה id ערך ת"ז
- את המנה של id ב- 100000000 נציב ב- start
- את שארית המנה id ב- 100 נציב למשתנה end
- נחבר את end עם start ונציב במשתנה code
- נדפיס את code

```
#include <stdio.h>
void main() {
```

```
    long id;
    int start, end, code;
```

הצהרה

```
    printf("enter id\n");
    scanf("%ld", &id);
```

קריאת קלט

```
    start = id / 100000000;
    end = id % 100;
    code = start + end;
```

חישובים

```
    printf("Code : %d \n", code);
```

הדפסת תוצאות

```
}
```



לביטוי כדוגמת: $x = x + 2 ;$

ניתן לכתוב ביטוי שקול: $x += 2 ;$

המשמעות: x מקבל את ערכו הקודם $+ 2$

ניתן לבצע הצבה מקוצרת לכל האופרטורים:

$+, -, *, / , \%$

לדוגמה:

$x *= y + 1 ;$

הביטוי

$x = x * (y + 1) ;$

שקול לביטוי

$x = x * y + 1 ;$

ולא לביטוי

הגדלה והקטנה עצמית

■ ++ אופרטור זה מוסיף 1 (increment).

■ -- אופרטור זה מוריד 1 (decrement).

$x++ \Leftrightarrow x = x + 1;$ $x-- \Leftrightarrow x = x - 1;$

■ ניתן להשתמש עבור משתנה מכל טיפוס

■ ניתן לשלב בביטוי מורכב:

$a = b++ - c;$

■ אופרטורים אלו יכולים להופיע בשתי צורות:

Pre increment/decrement $++i; --j;$

Post increment/decrement $k++; a--;$


```
int x = 3;
```

```
x++;
```

– כעת x הוא 4.

```
++x;
```

– כעת x הוא 5.

```
int x = 3, y;
```

```
y = ++x;
```

כעת x הוא 4, y הוא 4

```
int x = 3, y;
```

```
y = x++;
```

כעת x הוא 4, y הוא 3

הגדלה והקטנה עצמית

- כאשר ++ח או ++ח מופיעים כמשפט עצמאי (לא כחלק מביטוי או משפט השמה) אין הבדל בין שמאל לימין

- אם האופרטור נמצא מימין למשתנה (הגדלה/הקטנה מאוחרת), לדוגמה ++ח , אז לחישוב הביטוי משתמשים בערך של ח ולאחר מכן מגדילים ח ב-1.

- אם האופרטור נמצא משמאל למשתנה (קדם הגדלה/הקטנה), לדוגמה ++ח , אז קודם מגדילים ח ב-1, ולאחר מכן משתמשים בערך החדש של ח לחישוב הביטוי.

מה זה "לאחר מכן"?

```
void main()
{
    int x=5, y=8, z;

    z = x++ + --y;
    printf (" x=%d, y=%d, z=%d \n", x, y, z);
    z = x-- + y++;
    printf (" x=%d, y=%d, z=%d \n", x, y, z);
}
```

מה יודפס אחר ביצוע התכנית?

מה זה "לאחר מכן"?

```
void main()
```

```
{
```

```
    int x=5, y=8, z;
```

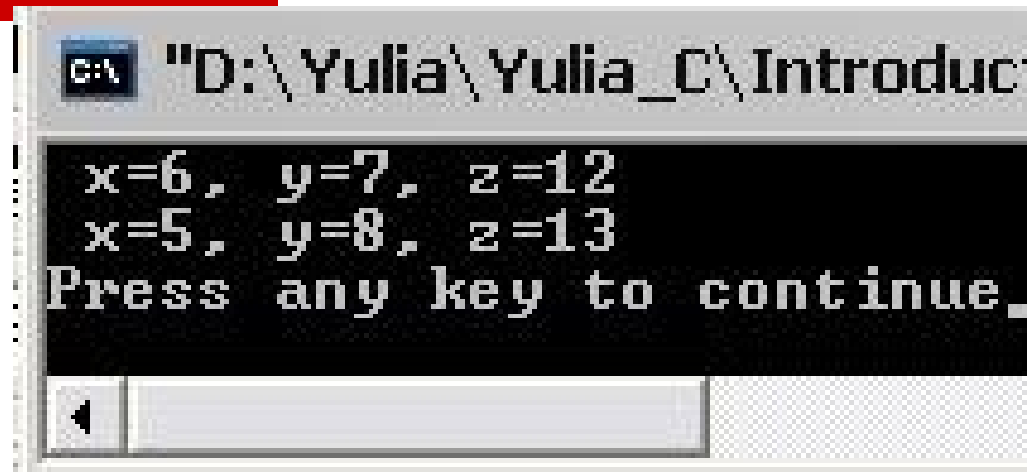
```
    z = x++ + --y;
```

```
    printf (" x=%d, y=%d, z=%d \n", x, y, z);
```

```
    z = x-- + y++;
```

```
    printf (" x=%d, y=%d, z=%d \n", x, y, z);
```

```
}
```



```
C:\ "D:\Yulia\Yulia_C\Introduc
x=6, y=7, z=12
x=5, y=8, z=13
Press any key to continue.
```

מה יודפס אחר ביצוע התכנית?



ניהול עיצוב הדפסה:
מספר תווים בהדפסה - cnt1 - **% cnt1.cnt2 type**
מספר תווים אחרי נקודה - cnt2