

מבוא לתכנות מערכות

הרצאה 1

מבוא, בסיס השפה, פקודות קלט/פלט

מבוסס על הרצאות של אוניברסיטת חיפה, ומבוא למדעי המחשב, אורט בראודה, תעו"ן

מבנה המחשב



ייצוג מידע במחשב

- כל הנתונים מיוצגים כמספרים (כולל אינפורמציה כגון טקסט, תמונות, קול, סרטים וכו').
- הזיכרון הראשי של המחשב הוא טבלה ארוכה של מספרים.
- לכל תא בטבלה (תא זיכרון) יש כתובת.
- הנתונים במחשב הם מספרים המיוצגים בספרות בינאריות.
- ◆ bit (ביט) – ספרה בבסיס 2 (0 או 1).
- ◆ byte (בית) – 8 ביטים (256 אפשרויות).
- בתאי הזיכרון ניתן לשמור נתונים. התאים רצופים ולכל byte יש כתובת משלו.

פתרון בעיות בעזרת מחשב

● על מנת שיהיה ניתן לפתור בעיה באמצעות מחשב, עליה להיות מוגדרת בצורה פורמלית.

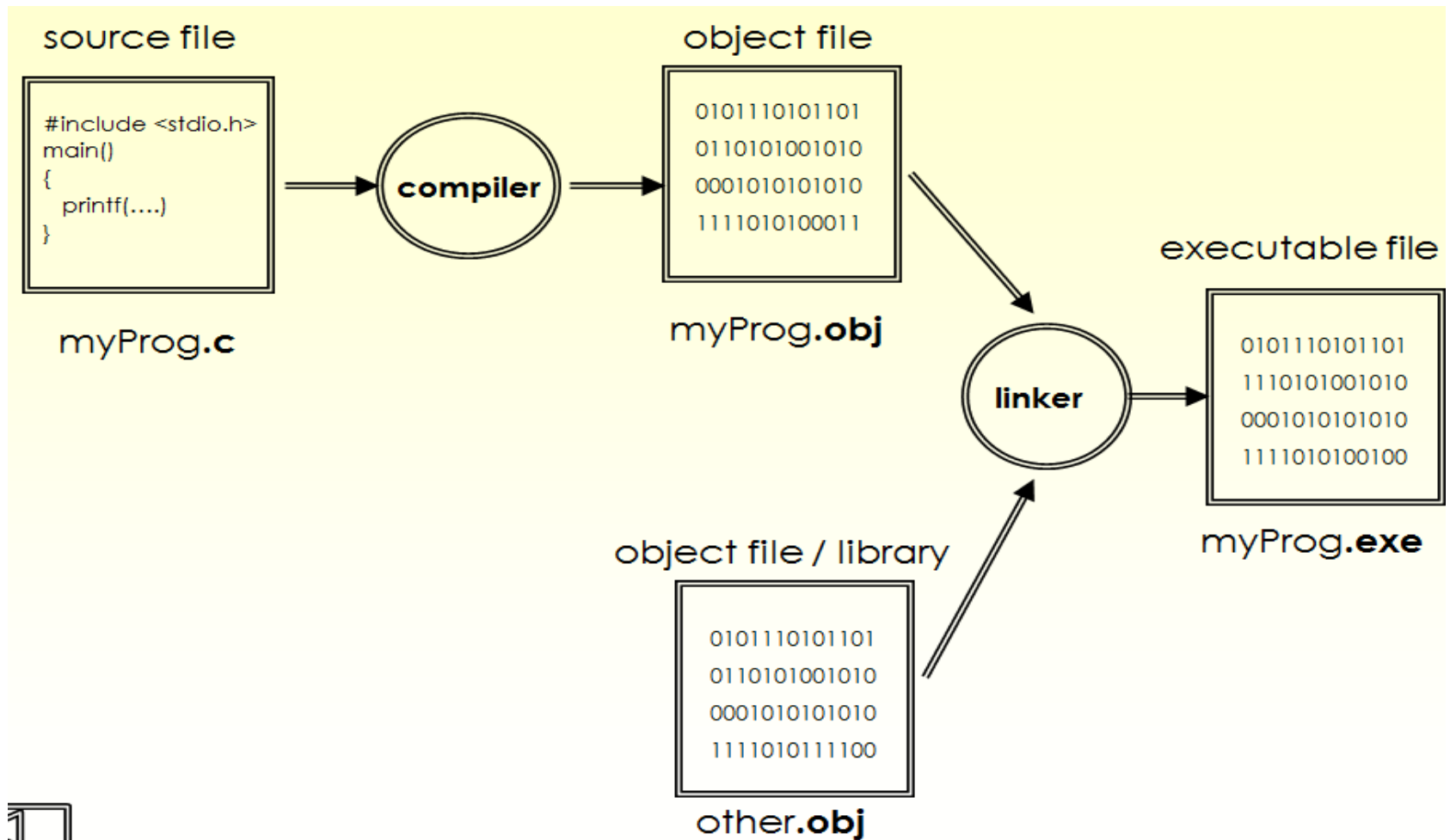
● שליבי הפתרון:

1. הבנה: ניתוח הבעיה
2. בחירת שיטת הפתרון: ניסוח אלגוריתם
3. מימוש: כתיבת תכנית מחשב
4. בדיקה (debugging)

● פרמטרים להערכת תכנית כתובה:

1. נכונות
2. יעילות

שפות תכנות עיליות



שגיאות

סוגי שגיאות:

1. שגיאות תחביר:

התכנית לא תעבור קומפילציה. תתקבל התראה מהקומפיילר.

2. שגיאות ריצה:

- התכנית עוברת קומפילציה אך "עפה" בזמן ריצה.
- קורה כאשר אנו מבקשים מן המחשב לעשות דברים שהוא אינו מסוגל. למשל – חלוקת מספר באפס.
- גילוי ע"י הרצת התכנית על מספר קלטים (מקרי קצה).

3. שגיאות לוגיות:

- התכנית עובדת אך מפיקה פלט שגוי.
- קושי לגילוי ולפיתרון.

מבנה כללי של תכנית

- התוכנית Hello World:

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

```
int main ( )  
{  
    printf("Hello World!\n");    /* Print! */  
    return 0;  
}
```

- בכל תוכנית C חייבת להיות פונקציה בשם main אחת ויחידה.
- בסיום עבודתה, הפונקציה main תחזיר ערך מסוג int שיכול לשמש את מערכת הפעלה.
- הנחיה ל- compiler (#) כי ספריית הקלט/פלט הסטנדרטית הנמצאת בקובץ stdio.h, תהיה בשימוש.
- \n הוא קבוע בקרה, משמעותו: לרדת שורה.
- הסימן; (נקודה פסיק) מסמן סוף משפט ב C והוא חובה בסיום כל הוראה.

משתנים – Variables

משתנה – variable: הוא מקום שמור בזיכרון עבור נתון, (כלומר – מין סל).

- **ערכו** של נתון יכול להשתנות במהלך התוכנית.
- **מזהה** – identifier: הוא "שם המשתנה" שנועד להקל על המשתמש לפנות אליו.
- **כתובת המשתנה** – address: המקום הפיזי בזיכרון המחשב שבו מאוחסן הנתון.
- **טיפוס** – type: סוג המשתנה - קובע את כמות הזיכרון הנדרשת עבור משתנה זה, וטווח הערכים שניתן להציב בו.

- לדוגמא:

```
int num = 15;
```

הוגדר משתנה בשם num עבור מספר שלם עם תוכן 15.

הטיפוס שנקבע עבור משתנה נשאר קבוע במהלך כל חיי התכנית, והמקום שיוקצה עבורו יישאר קבוע אף הוא.

תחביר הגדרת משתנים

הגדרה כוללת שם של טיפוס, ולאחריו רשימה של שמות משתנים, אחד או יותר, מופרדים בפסיקים זה מזה. בסוף הרשימה יבוא הסימן נקודה-פסיק (;):

```
int lower, upper, step;
char c;
```

כתיבה קומפקטית

זה שקול ל:

```
int lower; /* lower range of numbers*/
int upper; /* upper range of numbers*/
int step;
char c;
```

מאפשר לכתוב הערות (כל מה שנמצא בין /* לבין */) - הקומפיילר מתעלם לחלוטין מהערות.

כאשר מגדירים משתנים, ניתן ל**אתחל** אותם (וניתן לאתחל מספר משתנים בהגדרה אחת):

```
char one='1';
int i=0;
double x=2.4, y=3.5, z;
```

טיפוסי נתונים בסיסיים בשפת C (data types)

ייצוג מספרים:



שלמים (int, long int) ◆

ממשיים (double, float) ◆

ייצוג טקסט:



תווים (char) ◆

מחרוזות ◆

הטיפוסים השלמים

char , int, long int, unsigned char, unsigned int, unsigned long int

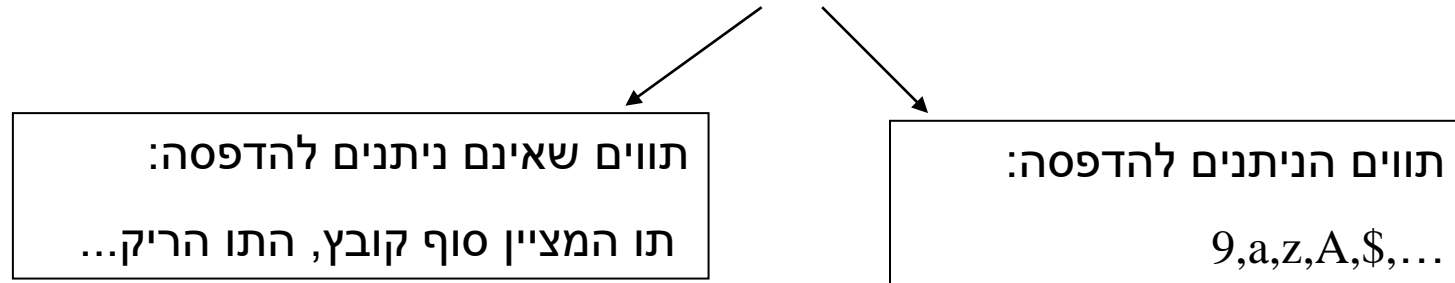
שם הטיפוס	הגדרה	bytes	טווח המספרים
תו	char	1	[-128, 127]
שלם	int	2	[-32768, 32767]
		4	$[-2^{31}, 2^{31}-1]$
שלם ארוך	long int	4	$[-2^{31}, 2^{31}-1]$
תו לא מסומן	unsigned char	1	[0, 255]
שלם לא מסומן	unsigned int	2	[0, 65535]
		4	[0, 4,294,967,295]
שלם ארוך לא מסומן	unsigned long int	4	[0, 4,294,967,295]

מספרים ממשיים float, double

שפת C תומכת ב-3 טיפוסים נתונים למספרים ממשיים (בהתאם לתחום ולדיוק הדרושים):

- **float** – דיוק של לפחות 5 ספרות אחרי הנקודה העשרונית ותחום מספרים של עד 10^{37} . גודל 4 בתים.
 - **double** - דיוק של לפחות 10 ספרות אחרי הנקודה העשרונית ותחום מספרים של עד 10^{37} . גודל 8 בתים.
 - **long double** – דיוק של לפחות כמו double ואולי יותר. גודל 10 בתים.
- הממדים המדויקים הם תלויי מימוש והשפה אינה מתחייבת לגביהם.

תווים characters



- מיוצגים במחשב ע"י byte אחד בדיוק (= 8 סיביות) <= יש 256 תווים שונים.
- התווים מיוצגים בזיכרון ע"י מספרים – ישנם תקנים הקובעים ערך מספרי לכל תו.
- התקן הנפוץ – תקן ASCII:
- ◆ מייחס לכל תו מספר בין 0 ל-255.
- ◆ האותיות הקטנות מיוצגות ע"י מספרים רציפים. כנ"ל לגבי האותיות הגדולות והספרות.
- שפת C מייצגת תווים בזיכרון ע"י שמירת ערך מספרי עפ"י תקן כלשהוא.

אלפבית

תווים המותרים בשימוש בתכניות C:

● אותיות לטיניות קטנות:

a,b,c,...,z

● אותיות לטיניות גדולות:

A,B, C,...,Z

● ספרות:

0,1,2,...,9

● רווחים:

רווח, tab, enter וכו'

● סימנים אחרים:

+, -, *, /, =, (,), [,], {, }, ', ", #, %, &, _, !, ?, \, <, >, ~, ^, |, .., ...

● תו מטיפוס char מוצג בין גרשיים. לדוגמא: '0', 'b', 'A', '*', '\\', \' ' .

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

מה בין ייצוג מספרים שלמים לייצוג תווים?

טיפול הנתונים של תווים הוא למעשה טיפוס הנתונים של מספר שלם, שייצוגו הפנימי צר במיוחד (רק בית אחד).

שימו לב :

כל הפעולות המותרות על שלמים מותרות גם על תווים, למשל: אופרטורים אריתמטיים.

קבועים

- לפעמים בתוכנית אנו מעוניינים בערך קבוע, למשל קבוע הבא: 3.14159265359.
- נגדיר:

```
#define PI 3.14159265359
```

- הוראה זו תגרום לכך, שכל המופעים של PI בתוכנית, יוחלפו בערך המוגדר.
- קבוע אינו תופס מקום בזיכרון, וכמובן שאינו ניתן לשינוי.
- בדרך כלל, מגדירים את כל הקבועים, מתחת לספריות.
- דוגמאות נוספות:

– קבוע מספרי שלם:

```
#define TOP_LIMIT 100
```

– קבוע מטיפוס תו:

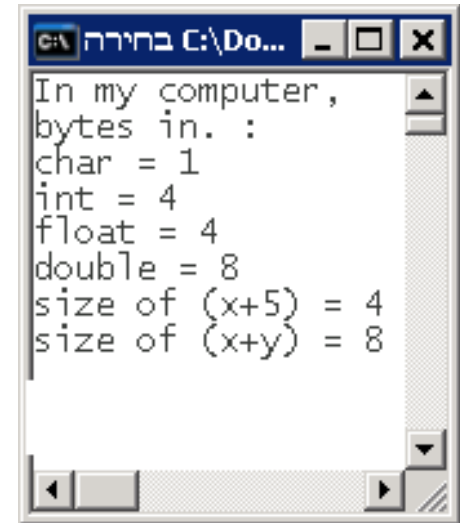
```
#define XOP 'x'
```

- לקבועים, מקובל לתת שם מאותיות גדולות בלבד.
- שימו לב שבסוף הגדרת קבוע אין ;

האופרטור sizeof()

- אופרטור זה מחזיר את הגודל בבתים של הביטוי, טיפוס או משתנה שבסוגריים.
- אופן השימוש: sizeof (type) או sizeof (expression)
- דוגמא:

```
#include <stdio.h>
int main()
{
    int x = 5 ;
    double y = 7.5 ;
    printf ("In my computer,\n bytes in. : \n");
    printf ("char = %d\n", sizeof( char ) );
    printf ("int = %d\n", sizeof( int ) );
    printf ("float = %d\n", sizeof( float ) );
    printf ("double = %d\n", sizeof( double ) );
    printf ("size of (x+5) = %d\n", sizeof( x+5 ) );
    printf ("size of (x+y) = %d\n", sizeof( x+y ) );
    return 0;
}
```



```
In my computer,
bytes in. :
char = 1
int = 4
float = 4
double = 8
size of (x+5) = 4
size of (x+y) = 8
```

Operators (פעולות)

פעולות בינאריות

השפה מכירה 5 פעולות חשבוניות בסיסיות:

$x = a + b;$ חיבור ■

$x = b - a;$ חיסור ■

$x = a * b;$ כפל ■

$x = b / a;$ חילוק ■

$x = c \% a;$ שארית (מודולו) ■

• חיסור אונרי: הפיכה לנגדי $-a;$

למשל $b = -a;$

למשל $b = -b;$

כלומר, לא צריך את: $b = (-1) * b;$

Operators (המשך)

הגדלה והקטנה עצמית (פעולות אונריות):

- הגדלה `++ a ;` או `a ++;`
- מתמטית, פעולה זו שקולה להגדלת a ב-1: `a=a+1 ;`

- הקטנה `--b ;` או `b --;`
- מתמטית, פעולה זו שקולה להקטנת b ב-1: `b=b-1 ;`

(המשך) Operators

מה יבצע קטע הקוד הבא? ■

a	b	c
?	?	?

```
int a, b, c;
```

```
a = 1;
```

```
a++;
```

```
b = a++;
```

```
c = --b;
```

```
/* Now a becomes 2 */
```

```
/* b <- a, b becomes 2, a becomes 3 */
```

```
/* b becomes 1, c <- b, c becomes 1 */
```

כתיב מקוצר

• C מאשרת לנו לכתוב באופן מקוצר את הפעולות הבאות:

• $a += b;$ $\Leftrightarrow a = a + b;$

• $a -= b;$ $\Leftrightarrow a = a - b;$

• $a *= b;$ $\Leftrightarrow a = a * b;$

• $a /= b;$ $\Leftrightarrow a = a / b;$

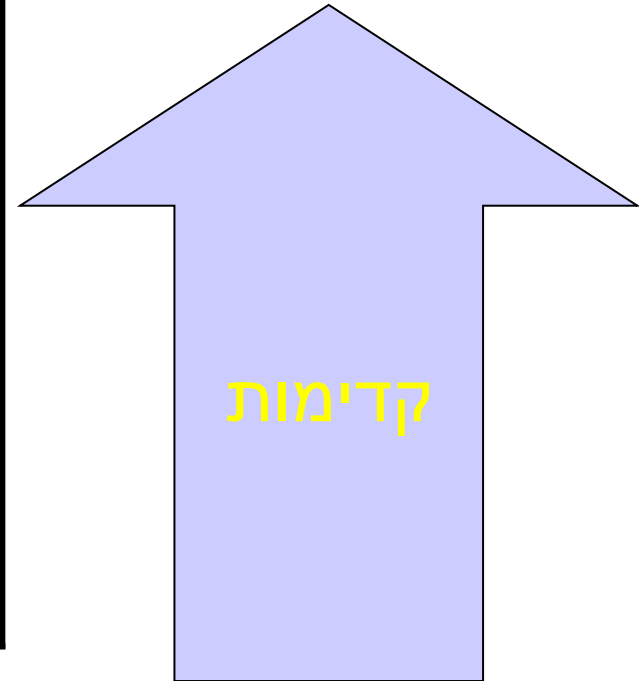
• $a \% = b;$ $\Leftrightarrow a = a \% b;$

Operators השוואתיים (לוגיים)

- השפה מכירה 6 פעולות השוואתיות:
- גדול $>$
 $x = a > b;$
- קטן $<$
 $x = a < b;$
- שווה $==$
 $x = a == b;$
- גדול או שווה $>=$
 $x = a >= b;$
- קטן או שווה $<=$
 $x = a <= b;$
- שונה $!=$
 $x = a != b;$
- ניתן לסכם פעולה לוגית בתוצאה שתהיה נכונה (true) או שקרית (false) ותתורגם ל-1 או 0 בהתאמה.
- מומלץ להשתמש בסוגריים כדי לא לזכור את קדימות האופרטורים.
- למשל: $x = (a == b);$

קדימויות האופרטורים

()
-, ++, -- sizeof()
(casting)
*, /, %
+, -



■ סדר החישוב הוא משמאל לימין.

משפטי השמה

- משפט השמה (הצבה) נותן ערך למשתנה, (דורס ערך קודם).
- משפט ההשמה יכול להיות בכל מקום בתוכנית.

- נניח שהגדרנו:

```
int a = 2, b, c, d;  
float x;
```

- משפטי השמה בין טיפוסים זהים:
הערך של המשתנה a יוצב במשתנה b $b = a$;

- ערך ההשמה מותאם לטיפוס אליו מתבצעת ההשמה:

`x=c;`

- הערך של c, יוצב במשתנה x – ויותאם לסוג המשתנה - 2.0.
על משתנה c פעולה זו לא תשפיע.

השמה בין טיפוסים שונים

- נניח כי הגדרנו:

- `int a = 2;`
- `double x = 3.0;`

- מה אם נבצע השמה?

- `a = x;`

- `a` הוא משתנה מסוג `int` שתופס 4 בתים, אבל `x` הוא משתנה מסוג `double` שתופס 8 בתים.

- בשפת C, משתנה אינו יכול לשנות את גודלו (מס' הבתים שהוא תופס) או את סוגו במהלך תוכנית,

- לכן הערך בן 8 בתים המועבר אל `a` יקוצץ ויוכנס אל `4byte`, לא בטוח שנקבל ערך נכון...

- כל ביטוי השמה אל `int` (למשל), יקבל פורמט של `int` !

אילוח טיפוס משתנה – Casting

- ניתן, באופן זמני, לשנות את ההתייחסות לטיפוס של המשתנה.

```
int a = 1, b = 2;
float r;
```

- נניח כי הגדרנו:

- הביטוי ; $r = a/b$, יכניס ל- r את הערך 0.0.

זאת מאחר ותוצאת החלוקה a/b היא 0 (פעולה בין שני משתנים מטיפוס מסוים נותנת תוצאה מאותו הטיפוס), ערך זה יומר ל 0.0 .

- הביטוי ; $r = (\text{float}) a/b$, יציב ב- r את 0.5.

זאת מאחר והתוצאה של החלוקה תומר ל- `double` רק בשורה הזאת .

- הביטוי ; $r = (\text{float}) (a/b)$, יציב ב- r את 0.0. זאת מאחר ותוצאת החילוק a/b הנה 0 (חלוקה בשלמים), והמרת הסוג לא תשנה ערך זה.

מעט על קלט/פלט

קבלת קלט • ← הפונקציה scanf

הדפסת פלט • ← הפונקציה printf

הפונקציה printf

- מדפיסה נתונים להתקן פלט (ברירת מחדל ← למסך).

- פרמטרים:

- מחרוזת בקרה (מתארת את האופן שבו יעוצבו הנתונים המודפסים).

- מספר כלשהוא של פרמטרים נוספים, מטיפוסים שונים, המפרטים את הנתונים שיש להדפיס (לפי סדר המתאים לסדר במחרוזת הבקרה).

דוגמא:

```
printf("The average of %d and %d is %f\n", 6, 7, (6+7)/2);
```

מחרוזת הבקרה

מודפסים כלשונם ← תווים רגילים

בעל ← תו בקרה – (מתחיל ב-\)
משמעות מיוחדת בשפה

תו מיוחד המציג סדרת תווי עיצוב – קובעים
כיצד יודפסו הנתונים.

הפונקציה printf

code	type	format
d	int	decimal (base ten) number
o	int	octal number (no leading '0' supplied in printf)
x or X	int	hexadecimal number (no leading '0x' supplied in printf; accepted if present in scanf) (for printf, 'X' makes it use upper case for the digits ABCDEF)
ld	long	decimal number
u	unsigned	decimal number
lu	unsigned long	decimal number
c	char	single character
s	char pointer	string
f	float	number with six digits of precision
lf	double	number with six digits of precision

printf() – פורמטים

- שייכת לספריה – `<stdio.h>`, מציגה נתונים מסך, הפרמטר המועבר נמצא בסוגריים.

- קיימות שתי אפשרויות ראשיות:

1. הדפסת מחרוזת בלבד, הפקודה תראה:

```
printf("the string to be displayed");
```

2. שילוב ערכים, הפקודה תורכב משני חלקים:

```
printf("%ctrl_char", variable_name);
```

למשל

```
printf("Result is %d", sum);
```

- `printf()` מחזירה כערך את מספר התווים שהודפסו.

printf() – פורמטים (המשך)

הדפסת escape sequences :

Escape sequence	Description
\'	Output the single quote (') character.
\"	Output the double quote (") character.
\?	Output the question mark (?) character.
\\	Output the backslash (\) character.
\a	Cause an audible (bell) or visual alert.
\b	Move the cursor back one position on the current line.
\f	Move the cursor to the start of the next logical page.
\n	Move the cursor to the beginning of the next line.
\r	Move the cursor to the beginning of the current line.
\t	Move the cursor to the next horizontal tab position.
\v	Move the cursor to the next vertical tab position.

הפונקציה printf - דוגמא

```
#include <stdio.h>

int main()
{
    int i=7;
    char ch='3';
    double price=2.30;
    printf("The average of 6 and %d is %f\n", i, (6+i)/2.0);
    printf("The character %c has the value of %d\n", ch, ch);
    printf("The price of %d %ss is $%.2lf\n", i, "apple", i*price);
    return 0;
}
```

OUTPUT:

The average of 6 and 7 is 6.500000

The character 3 has the value of 51

The price of 7 apples is \$16.10

קלט מפורמט - scanf

- קולטת ערכים מן המשתמש.
- כמו printf() מורכבת ממחרוזת בקרה, ואחריה רשימת משתנים.
- במחרוזת יופיעו טיפוסים הנתונים שיש לקרוא מן המשתמש, ובסדר זה יופיעו אחריה שמות המשתנים לתוכם ייקראו ערכים אלו.
- לפני שם של משתנה יופיע הסימן & (ampersand) שפירושו: ה"כתובת של".

דוגמה:

```
scanf("%d %d", &num1, &num2);
```

- גם scanf () מחזירה ערך: מספר הארגומנטים שנקלטו!
- תווי הבקרה (פורמטים) של scanf() זהים לאילו של printf().

תוכנית לחישוב סכום שני מספרים

/* Calculates and presents the sum of two input integer numbers */

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

```
int main( )
```

```
{
```

```
    int a, b, sum;           /*variable declaration */
```

```
    printf ("Enter two integer numbers\n");
```

```
    scanf ("%d %d", &a, &b);
```

```
    sum = a+b;
```

```
    printf("%d + %d = %d", a, b, sum);
```

```
    return 0;
```

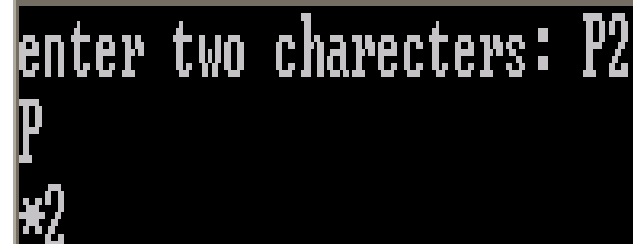
```
}
```

פעולות: קלט פלט על תווים

putchar() מקבלת תו ומציגה אותו על המסך
getchar() קולטת תו מן המקלדת ומחזירה אותו

```
char c1, c2;  
printf("enter two characters: ");  
c1=getchar();  
c2=getchar();  
putchar(c1);  
putchar('\n');  
putchar('*');  
putchar(c2);
```

ב- <stdio.h>



```
enter two characters: P2  
P  
*2
```