

ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ ΥΠΟΛΟΓΙΣΤΩΝ & ΥΠΟΛΟΓΙΣΤΙΚΗ ΦΥΣΙΚΗ

Μέρος 2ο

ΝΙΚΟΛΑΟΣ ΣΤΕΡΓΙΟΥΛΑΣ



ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΦΑΛΜΑΤΑ ΜΕΤΑΓΛΩΤΤΙΣΗΣ

- Η γλώσσα C κάνει αυστηρή **διάκριση μεταξύ πεζών και κεφαλαίων**.
- Ο μεταγλωττιστής ενδέχεται να δείξει **μηνύματα λάθους ή προειδοποιήσεις** για συγκεκριμένες γραμμές.
- Το πραγματικό σφάλμα μπορεί να βρίσκεται σε προηγούμενες γραμμές!
- Διορθώνουμε τα σφάλματα **ένα-ένα** και τρέχουμε κάθε φορά ξανά το μεταγλωττιστή.
- Ο μεταγλωττιστής εντοπίζει μόνο τυπογραφικά ή συντακτικά σφάλματα, **όχι λογικά σφάλματα (!)**

ΛΟΓΙΚΑ ΣΦΑΛΜΑΤΑ

- ➔ Ένα πρόγραμμα είναι πολύ πιθανό να περιέχει **λογικά σφάλματα (bugs)**. Καθώς αυτά δεν ανιχνεύονται από το μεταγλωττιστή, το πρόγραμμα θα τρέξει, αλλά τα αποτελέσματα θα είναι **μη-αναμενόμενα**. Τότε χρειάζεται αποσφαλμάτωση (**debugging**).
- ➔ Δεν αρκεί να τρέξει ένα πρόγραμμα. Πρέπει **να ελέγχουμε πάντοτε τα αποτελέσματα** για να σιγουρευτούμε πως εκτελέστηκαν ακριβώς οι εντολές που είχαμε κατά νου.
- ➔ **Δε φταίει ποτέ "ο υπολογιστής"** εάν ένα πρόγραμμα δεν έβγαλε σωστό αποτέλεσμα, **αλλά πάντοτε ο προγραμματιστής**.

ΔΗΛΩΣΗ ΣΤΑΘΕΡΑΣ

```
#include <stdio.h>

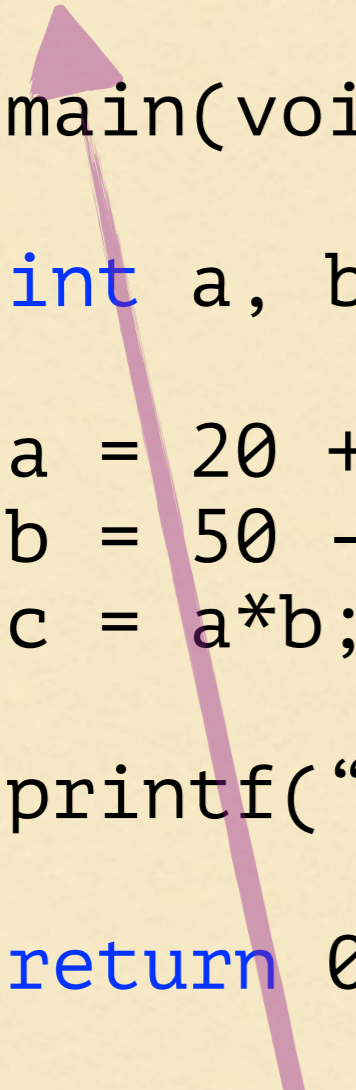
#define NUM 10

int main(void)
{
    int a, b, c;

    a = 20 + NUM;
    b = 50 - NUM;
    c = a*b;

    printf("%d %d %d\n", a, b, c);

    return 0;
}
```



Η οδηγία **#define** ορίζει μια **σταθερά** και καθορίζει την τιμή της.

ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

```
#include <stdio.h>

#define NUM 10

int main(void)
{
    int a, b, c;

    a = 20 + NUM;
    b = 50 - NUM;
    c = a*b;

    printf("%d %d %d\n", a, b, c);

    return 0;
}
```

Η εντολή `int a, b, c;` ορίζει τρεις νέες μεταβλητές τύπου `int` (integer - **ακέραιος**).

ΚΑΘΟΡΙΣΜΟΣ ΤΙΜΗΣ ΜΙΑΣ ΜΕΤΑΒΛΗΤΗΣ

```
#include <stdio.h>

#define NUM 10

int main(void)
{
    int a, b, c;

    a = 20 + NUM;
    b = 50 - NUM;
    c = a*b;

    printf("%d %d %d\n", a, b, c);

    return 0;
}
```

Με το `=` καθορίζουμε την τιμή που λαμβάνει μια μεταβλητή.

ΚΑΘΟΡΙΣΜΟΣ ΤΙΜΗΣ ΜΙΑΣ ΜΕΤΑΒΛΗΤΗΣ

```
#include <stdio.h>

#define NUM 10

int main(void)
{
    int a, b, c;

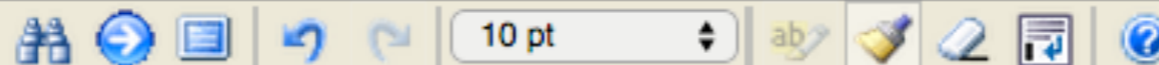
    a = 20 + NUM;
    b = 50 - NUM;
    c = a*b;

    printf("%d %d %d\n", a, b, c);
    return 0;
}
```

Με την `printf` τυπώνουμε τις τιμές που λαμβάνουν οι μεταβλητές `a`, `b`, `c`. Η μορφοποίηση για `int` είναι `%d`.

ΜΕΤΑΓΛΩΤΤΙΣΗ ΚΑΙ ΕΚΤΕΛΕΣΗ ΜΕ RexTester

Language: C (gcc) Editor: EditArea



```
1 #include <stdio.h>
2
3 #define NUM 10
4
5 int main(void)
6 {
7     int a, b, c;
8
9     a = 20 + NUM;
10    b = 50 - NUM;
11    c = a*b;
12
13    printf("%d %d %d\n",a, b, c);
14
15    return 0;
16 }
17
18
```

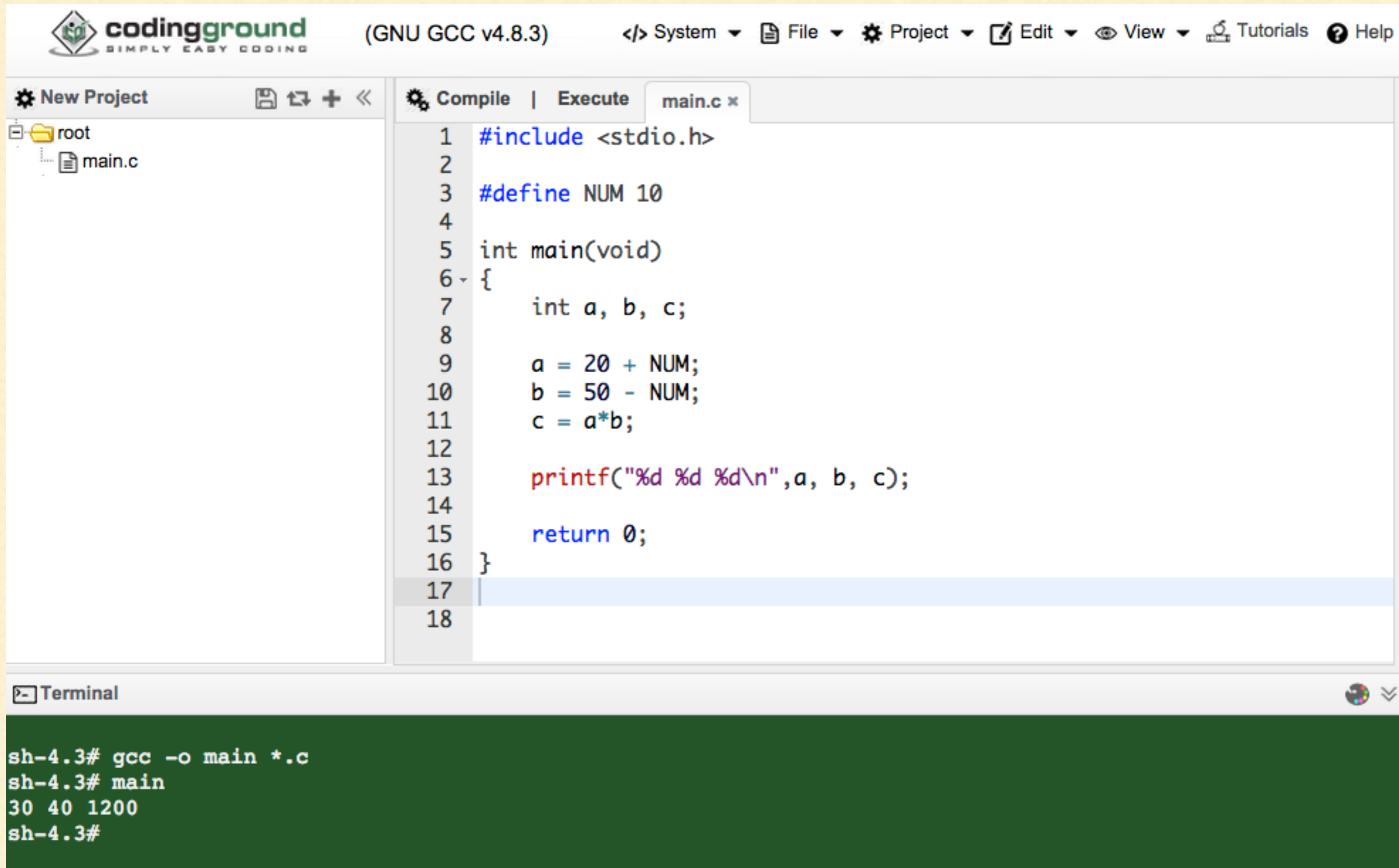
Position: Ln 1, Ch 1 Total: Ln 16, Ch 160

Run it **Save it** Show compiler warnings **[+]** Compiler args **[+]** Show input **Put on a wall** **?**

Compilation time: 0.12 sec, absolute running time: 0.04 sec, cpu time: 0 sec, memory peak: 3 Mb, absolute service time: 0.17 sec

30 40 1200

ΜΕΤΑΓΛΩΤΤΙΣΗ ΚΑΙ ΕΚΤΕΛΕΣΗ ΜΕ CodingGround



The image shows the CodingGround IDE interface. The top bar includes the logo, the text "(GNU GCC v4.8.3)", and a menu with options: System, File, Project, Edit, View, Tutorials, and Help. The left sidebar shows a file explorer with a "root" folder containing a "main.c" file. The main editor area displays the following C code:

```
1 #include <stdio.h>
2
3 #define NUM 10
4
5 int main(void)
6 {
7     int a, b, c;
8
9     a = 20 + NUM;
10    b = 50 - NUM;
11    c = a*b;
12
13    printf("%d %d %d\n",a, b, c);
14
15    return 0;
16 }
17
18
```

Below the editor is a terminal window with the following output:

```
sh-4.3# gcc -o main *.c
sh-4.3# main
30 40 1200
sh-4.3#
```

ΜΕΤΑΓΛΩΤΤΙΣΗ ΚΑΙ ΕΚΤΕΛΕΣΗ ΜΕ CodeLite

The screenshot displays the CodeLite IDE interface. The main editor shows the following C code:

```
1 #include <stdio.h>
2
3 #define NUM 10
4
5 int main(void)
6 {
7     int a, b, c;
8
9     a = 20 + NUM;
10    b = 50 - NUM;
11    c = a*b;
12
13    printf("%d %d %d\n", a, b, c);
14
15    return 0;
16 }
17
18
```

The Output View at the bottom shows the execution results:

```
Current working directory: /Users/niksterg/Dropbox/ProgCourse/Ch2-1/Debug
Running program: /usr/bin/open /Applications/codelite.app/Contents/MacOS/codelite-terminal.app --args --exit --wait --working-directory "/Users/niksterg/Dropbox/ProgCourse/Ch2-1-
Program exited with return code: 0
```

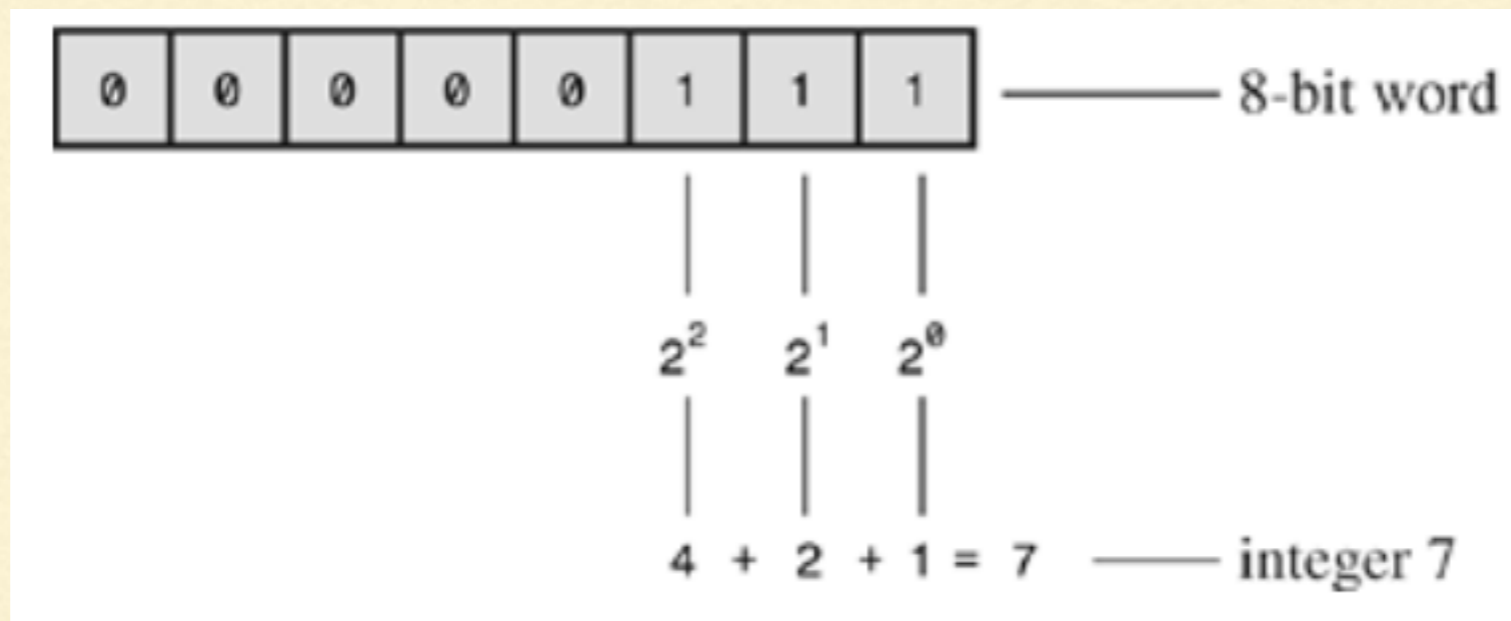
The terminal window also shows the output of the program:

```
30 40 1200
Hit ENTER to continue...
```

The bottom status bar indicates the current cursor position: Ln 13 Col 12 Pos 133, Normal bookmark. The environment is set to Default, Debug: Default.

ΜΝΗΜΗ RAM ΥΠΟΛΟΓΙΣΤΗ

- Η μνήμη (RAM - random access memory) ενός υπολογιστή αποτελείται από πολλές θέσεις αποθήκευσης δεδομένων με διαδοχική αρίθμηση.
- Το τυπικό μέγεθος μνήμης ακέραιου αριθμού είναι **8 bit = 1 byte**, π.χ.



- Ένας υπολογιστής με **2GB RAM** έχει μνήμη:

$$2 * 1024 * 1.024 * 1024 = 2.147.483.648 \text{ bytes}$$

ΜΝΗΜΗ RAM ΥΠΟΛΟΓΙΣΤΗ

- Κάθε θέση μνήμης μπορεί να έχει ένα **όνομα** και ένα **περιεχόμενο**.
- **Μεταβλητή** ονομάζεται μία θέση μνήμης που της δίνουμε ένα **συγκεκριμένο όνομα**.
- Η **τιμή μιας μεταβλητής** είναι το **περιεχόμενο** αυτής της θέσης μνήμης (μπορεί να αλλάξει πολλές φορές κατά τη διάρκεια εκτέλεσης του προγράμματος).

ΟΝΟΜΑΤΑ ΜΕΤΑΒΛΗΤΩΝ

- ➔ Μπορεί να αποτελείται από πεζά και κεφαλαία γράμματα του λατινικού αλφαβήτου, από ψηφία, καθώς και του `_` (underscore).
- ➔ Ο πρώτος χαρακτήρας πρέπει να είναι γράμμα ή `_`.
- ➔ Η γλώσσα C κάνει διάκριση μεταξύ των πεζών και κεφαλαίων γραμμάτων (*case sensitive*).
- ➔ Οι δεσμευμένες λέξεις της C απαγορεύεται να χρησιμοποιηθούν ως ονόματα μεταβλητών.

ΔΕΣΜΕΥΜΕΝΕΣ ΛΕΞΕΙΣ ΤΗΣ C

Οι ακόλουθες λέξεις είναι **δεσμευμένες** από τη C για δικές της εντολές και δεν πρέπει να δηλώνονται ως ονόματα μεταβλητών:

<code>auto</code>	<code>do</code>	<code>goto</code>	<code>signed</code>	<code>unsigned</code>
<code>break</code>	<code>double</code>	<code>if</code>	<code>sizeof</code>	<code>void</code>
<code>case</code>	<code>else</code>	<code>int</code>	<code>static</code>	<code>volatile</code>
<code>char</code>	<code>enum</code>	<code>long</code>	<code>struct</code>	<code>while</code>
<code>const</code>	<code>extern</code>	<code>register</code>	<code>switch</code>	
<code>continue</code>	<code>for</code>	<code>return</code>	<code>typedef</code>	
<code>default</code>	<code>float</code>	<code>short</code>	<code>union</code>	

ΔΗΛΩΣΗ ΜΕΤΑΒΛΗΤΩΝ

Η δήλωση μιας μεταβλητής γίνεται συνήθως στην **αρχή του προγράμματος** (οποσδήποτε όμως πριν αυτή χρησιμοποιηθεί) με τον ακόλουθο τρόπο:

τύπος_δεδομένων όνομα_μεταβλητής;

π.χ. **int** a;

Κάθε μεταβλητή έχει έναν συγκεκριμένο **τύπο δεδομένων**. Η **εκχώρηση** της τιμής στη μεταβλητή μπορεί να γίνει και με τη δήλωσή της:

π.χ. **int** a=100;

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Τύπος	Συνηθισμένο μέγεθος (bytes)	Εύρος τιμών (min-max)	Ψηφία ακρίβειας
<code>char</code>	1	-128 ... 127	
<code>short</code>	2	-32.768 ... 32.767	
<code>int</code>	4	-2.147.483.648...2.147.483.647	
<code>long</code>	4	-2.147.483.648...2.147.483.647	
<code>float</code>	4	Χαμηλότερη θετική τιμή: $1.17 * 10^{-38}$ Υψηλότερη θετική τιμή: $3.4 * 10^{38}$	6
<code>double</code>	8	Χαμηλότερη θετική τιμή: $2.2 * 10^{-308}$ Υψηλότερη θετική τιμή: $1.8 * 10^{308}$	15
<code>unsigned char</code>	1	0 ... 255	
<code>unsigned short</code>	2	0 ... 65535	
<code>unsigned int</code>	4	0 ... 4.294.967.295	
<code>unsigned long</code>	4	0 ... 4.294.967.295	

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

Τύπος	Συνηθισμένο μέγεθος (bytes)	Εύρος τιμών (min-max)	Ψηφία ακρίβειας
char	1	-128 ... 127	
float	4	Χαμηλότερη θετική τιμή: $1.17 \cdot 10^{-38}$ Υψηλότερη θετική τιμή: $3.4 \cdot 10^{38}$	6
double	8	Χαμηλότερη θετική τιμή: $2.2 \cdot 10^{-308}$ Υψηλότερη θετική τιμή: $1.8 \cdot 10^{308}$	15
unsigned char	1	0 ... 255	
unsigned short	2	0 ... 65535	
unsigned int	4	0 ... 4.294.967.295	
unsigned long	4	0 ... 4.294.967.295	

Για τον τύπο char, οι αριθμοί αντιστοιχούν στη θέση των χαρακτήρων στον πίνακα ASCII.

ΠΙΝΑΚΑΣ ΧΑΡΑΚΤΗΡΩΝ ASCII

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]

ΠΑΡΑΔΕΙΓΜΑ ΕΚΤΥΠΩΣΗΣ ΧΑΡΑΚΤΗΡΑ ASCII

```
1  #include <stdio.h>
2
3  int main(void)
4  {
5      char a;
6
7      a=65;
8
9      printf("Character = %c\n", a);
10     printf("Character = %c\n", 65);
11     printf("Character = %c\n", 'A');
12
13     return 0;
14 }
```

./Ch2-2

```
Character = A
Character = A
Character = A
```

ΤΥΠΟΙ ΔΕΔΟΜΕΝΩΝ

- ➔ Το μέγεθος της μνήμης που δεσμεύει ένας τύπος δεδομένων μπορεί να διαφέρει από υπολογιστή σε υπολογιστή (π.χ. ο τύπος `int` μπορεί να δεσμεύει 2 bytes σε κάποιον υπολογιστή και όχι 4 bytes).
- ➔ Ο τελεστής `sizeof` μας δίνει την πληροφορία για το πόσες οκτάδες δεσμεύει ένας τύπος δεδομένων στον υπολογιστή (θα τον δούμε στη συνέχεια).
- ➔ Χρησιμοποιούμε τον τύπο `float` μόνο όταν η ακρίβεια των δεκαδικών ψηφίων δεν είναι τόσο σημαντική.
- ➔ Για ακρίβεια πολλών δεκαδικών ψηφίων, χρησιμοποιούμε τον τύπο `double`.

ΕΚΧΩΡΗΣΗ ΤΙΜΩΝ ΣΕ ΜΕΤΑΒΛΗΤΕΣ

- ➔ Για μεταβλητές τύπου **float** ή **double** χρησιμοποιείται η **τελεία (.)** για το δεκαδικό μέρος και όχι το κόμμα (,)

π.χ. `float a = 1.24;`

- ➔ Αν **μπροστά από μία ακέραια τιμή** υπάρχει το ψηφίο **0**, τότε αυτή η τιμή ερμηνεύεται σαν **οκταδικός αριθμός**

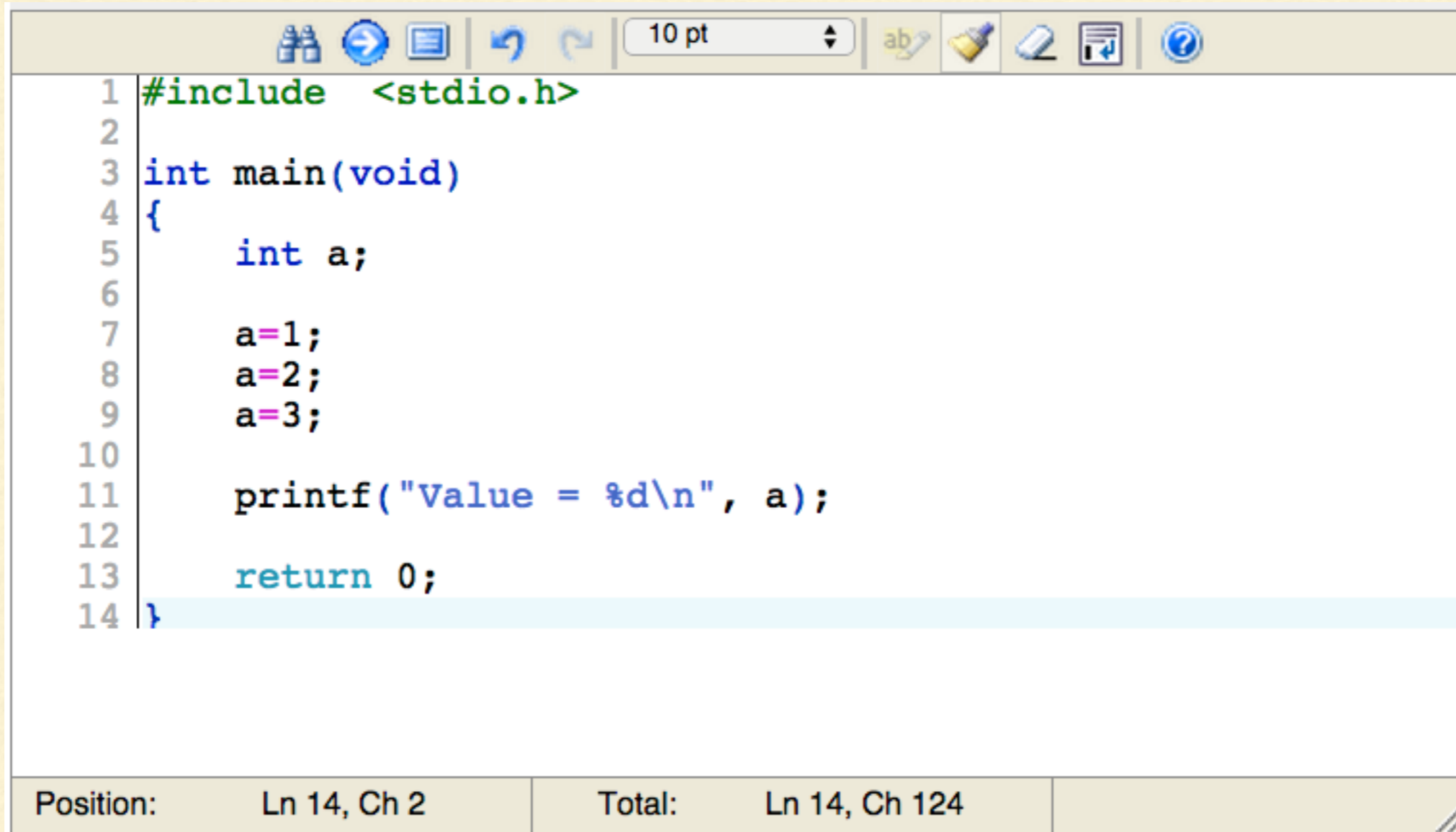
π.χ. `int a = 0100;` -> `a=64`

- ➔ Αν **μπροστά από μία ακέραια τιμή** υπάρχει το **0x** ή το **0X**, τότε αυτή η τιμή ερμηνεύεται σαν **δεκαεξαδικός αριθμός**

π.χ. `int a = 0x10;` -> `a=16`

ΕΚΧΩΡΗΣΗ ΤΙΜΩΝ ΣΕ ΜΕΤΑΒΛΗΤΕΣ

Εάν η τιμή της μεταβλητής αλλάξει στη διάρκεια εκτέλεσης του προγράμματος, **ισχύει πάντα η πιο πρόσφατη εκχώρηση τιμής:**



```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     int a;
6
7     a=1;
8     a=2;
9     a=3;
10
11     printf("Value = %d\n", a);
12
13     return 0;
14 }
```

Position: Ln 14, Ch 2 Total: Ln 14, Ch 124

π.χ. το παραπάνω πρόγραμμα θα τυπώσει **Value = 3** .

ΕΚΧΩΡΗΣΗ ΤΙΜΩΝ ΣΕ ΜΕΤΑΒΛΗΤΕΣ

➔ Η τιμή που εκχωρείται σε μια μεταβλητή **πρέπει να αντιστοιχεί στον τύπο της**, αλλιώς αποκόπτεται (δε στρογγυλοποιείται) ή μετατρέπεται.

π.χ. **int** a = 5.7; -> a = 5 (αποκοπή)

π.χ. **double** x = 5; -> x = 5.0 (μετατροπή)

➔ Η τιμή που εκχωρείται σε μια μεταβλητή **πρέπει να είναι μέσα στο επιτρεπτό εύρος τιμών**.

π.χ. **short** x = 40000; -> x = -25536 (λάθος)

π.χ. **char** ch = 130; -> (δεν υπάρχει στον πίνακα ASCII !)

ΕΠΙΣΤΗΜΟΝΙΚΗ ΣΗΜΕΙΟΓΡΑΦΙΑ

→ Σε μεταβλητές τύπου `int`, `float`, `long`, `double` (και οι αντίστοιχες `unsigned`) μπορούμε να εκχωρήσουμε τιμές στην επιστημονική σημειογραφία, χρησιμοποιώντας το `e` ή το `E` για να δηλώσουμε τη δύναμη του **10**.

π.χ. `double x = 5.2e-2;` \rightarrow `x = 0.052`

π.χ. `double x = 46E-3;` \rightarrow `x = 0.046`

ΣΤΑΘΕΡΕΣ

➔ Αν θέλουμε μια μεταβλητή να έχει πάντα σταθερή τιμή, τη δηλώνουμε ως `const`.

π.χ. `const int a = 100;`

Η τιμή εκχωρείται υποχρεωτικά κατά τη δήλωση και δε μπορεί να αλλάξει αργότερα.

➔ Εναλλακτικά, μπορούμε να δηλώσουμε σταθερές μέσω της μακροεντολής `#define`.

π.χ. `#define A 100`

(στην αρχή του προγράμματος, με κεφαλαία και χωρίς ;).

Η ΣΥΝΑΡΤΗΣΗ `printf`

→ Η γενική μορφή της `printf` είναι

```
printf("μορφοποίηση", μεταβλητή1, μεταβλητή2, ...);
```

Η `μορφοποίηση` μπορεί να περιέχει:

→ λέξεις και χαρακτήρες,
π.χ. `Value =`

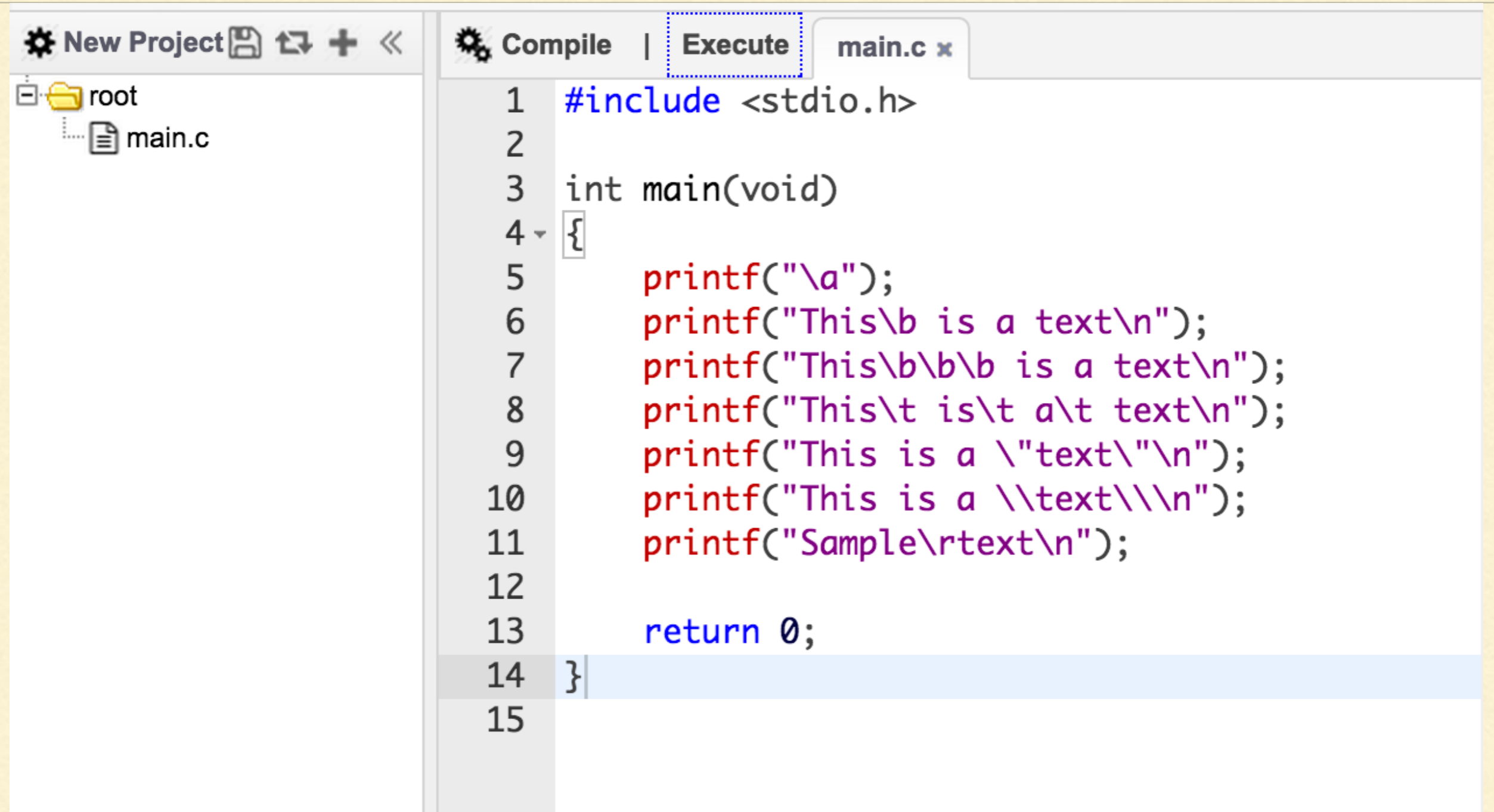
→ ακολουθίες διαφυγής
π.χ. `\n`

→ προσδιοριστικά μετατροπής
π.χ. `%d`

ΑΚΟΛΟΥΘΙΕΣ ΔΙΑΦΥΓΗΣ

Ακολουθία διαφυγής	Σημασία
<code>\a</code>	Χρησιμοποιείται για τη δημιουργία ηχητικού σήματος.
<code>\b</code>	Χρησιμοποιείται για τη διαγραφή του τελευταίου χαρακτήρα, όπως το πλήκτρο <code>backspace</code> .
<code>\n</code>	Χρησιμοποιείται για την αλλαγή γραμμής, όπως το πλήκτρο <code>Enter</code> .
<code>\r</code>	Χρησιμοποιείται για την επαναφορά του δρομέα στην αρχή της τρέχουσας γραμμής.
<code>\t</code>	Χρησιμοποιείται για τη μετακίνηση του δρομέα σε μία απόσταση ίση με το μήκος του <code>tab</code> , όπως το πλήκτρο <code>tab</code> .
<code>\\</code>	Χρησιμοποιείται για την εμφάνιση της ανάστροφης κεκλιμένης (<code>\</code>).
<code>\"</code>	Χρησιμοποιείται για την εμφάνιση των διπλών εισαγωγικών (<code>"</code>).

ΠΑΡΑΔΕΙΓΜΑ ΣΤΟ CodingGround



The screenshot shows the CodingGround IDE interface. On the left, a file explorer shows a 'root' directory containing a 'main.c' file. The main editor area displays the following C code:

```
1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("\a");
6     printf("This\b is a text\n");
7     printf("This\b\b\b is a text\n");
8     printf("This\t is\t a\t text\n");
9     printf("This is a \"text\"\n");
10    printf("This is a \\text\\n");
11    printf("Sample\rttext\n");
12
13    return 0;
14 }
15
```

The 'Execute' button is highlighted with a red dashed box. Below the code editor, a terminal window shows the output of the program:

Terminal

```
T is a text
This   is      a      text
This is a "text"
This is a \text\
textle
sh-4.3#
```

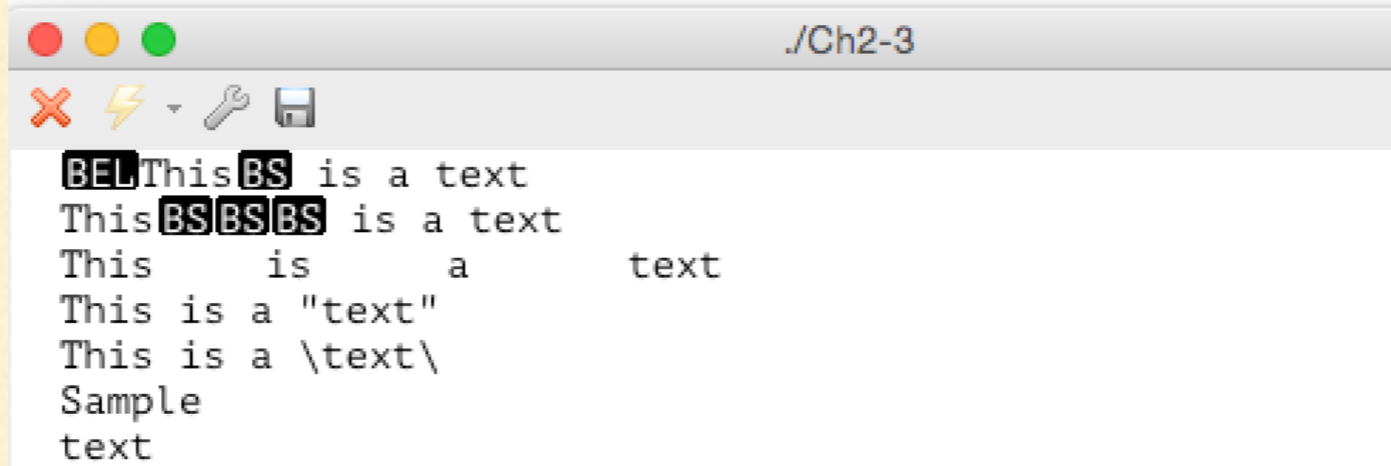
ΠΑΡΑΔΕΙΓΜΑ ΣΤΟ CodeLite

ΠΡΟΣΟΧΗ: Η σωστή εκτύπωση των ακολουθιών διαφυγής εξαρτάται και από την κωδικοποίηση του terminal στο οποίο εκτυπώνουμε. Π.χ. ο μεταγλωττιστής CodeLite έχει ένα ενσωματωμένο terminal που (τουλάχιστον στο OS X) δεν είναι πλήρως συμβατό με ASCII, με αποτέλεσμα τη **λανθασμένη εκτύπωση των \b και \r (!)** Όμοιο πρόβλημα εμφανίζει το RexTester.

```
#include <stdio.h>

int main(void)
{
    printf("\a");
    printf("This\b is a text\n");
    printf("This\b\b\b is a text\n");
    printf("This\t is\t a\t text\n");
    printf("This is a \"text\"\n");
    printf("This is a \\text\\n");
    printf("Sample\rtext\n");

    return 0;
}
```



```
./Ch2-3
BELThisBS is a text
ThisBSBSBS is a text
This is a text
This is a "text"
This is a \text\
Sample
text
```

ΠΡΟΣΔΙΟΡΙΣΤΙΚΑ ΜΕΤΑΤΡΟΠΗΣ

Χαρακτήρας μετατροπής	Σημασία
c	Χρησιμοποιείται για την εμφάνιση του χαρακτήρα που αντιστοιχεί σε μία ακέραια τιμή.
d ή i	Χρησιμοποιείται για την εμφάνιση ενός ακεραίου.
u	Χρησιμοποιείται για την εμφάνιση ενός μη-προσημασμένου ακεραίου.
f	Χρησιμοποιείται για την εμφάνιση ενός πραγματικού αριθμού. Η εξ' ορισμού ακρίβεια είναι έξι δεκαδικά ψηφία.
s	Χρησιμοποιείται για την εμφάνιση των χαρακτήρων ενός αλφαριθμητικού.
e ή E	Χρησιμοποιείται για την εμφάνιση ενός πραγματικού αριθμού σε επιστημονική μορφή. Ανάλογα με την επιλογή, εμφανίζεται το γράμμα e ή E πριν από τον εκθέτη.
g ή G	Χρησιμοποιείται για την εμφάνιση ενός πραγματικού αριθμού σε κανονική ή επιστημονική μορφή.
p	Χρησιμοποιείται για την εμφάνιση μίας διεύθυνσης μνήμης σε δεκαεξαδική μορφή.
x ή X	Χρησιμοποιείται για την εμφάνιση ενός μη-προσημασμένου ακεραίου σε δεκαεξαδική μορφή. Με το $\text{\$x}$ τα δεκαεξαδικά ψηφία (a-f) εμφανίζονται πεζά, ενώ με το $\text{\$X}$ εμφανίζονται ως κεφαλαία (A-F).
o	Χρησιμοποιείται για την εμφάνιση ενός μη-προσημασμένου ακεραίου σε οκταδική μορφή.
\%	Χρησιμοποιείται για την εμφάνιση του χαρακτήρα %.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    printf("%c\n", 'w');
    printf("%d\n", -1000);
    printf("%f\n", 2.73);
    printf("%s\n", "Αυτό είναι κείμενο.");
    printf("%e\n", 120.35);
    printf("%X\n", 24);
    printf("%o\n", 24);

    return 0;
}
```



./Ch2-4



```
w
-1000
2.730000
Αυτό είναι κείμενο.
1.203500e+02
18
30
```

ΕΚΤΥΠΩΣΗ ΤΙΜΩΝ ΜΕΤΑΒΛΗΤΩΝ

- ➔ Δηλώνουμε τις μεταβλητές που θέλουμε να εκτυπωθούν οι τιμές τους μετά τη μορφοποίηση (" ") με τη χρήση κόμματος (,). Πολλές μεταβλητές διαχωρίζονται μεταξύ τους με κόμμα.
- ➔ Οι μεταβλητές αντιστοιχίζονται μία-προς-μία, από αριστερά προς τα δεξιά, με τα προσδιοριστικά μετατροπής που δηλώσαμε μέσα στη μορφοποίηση (" ").
- ➔ ΠΡΟΣΟΧΗ: Αν οι μεταβλητές είναι **λιγότερες** από τα προσδιοριστικά μετατροπής, τότε για τα επιπλέον προσδιοριστικά εμφανίζονται **τυχαίες τιμές!**
- ➔ Αντίστοιχα, αν οι μεταβλητές είναι **περισσότερες** από τα προσδιοριστικά μετατροπής, τότε **δεν εκτυπώνονται οι τιμές τους!**

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

int main(void)
{
    int a, b;

    a = 10;
    b = 30;

    printf("Η τιμή = %d\n", a);
    printf("Οι τιμές είναι %d και %d\n", a, b);
    printf("Το άθροισμα = %d\n", a+b);
    printf("Οι τιμές είναι %d και %d και %d\n", a, b);
    printf("Η τιμή = %d\n", a, b);

    return 0;
}
```



./Ch2-5



```
Η τιμή = 10
Οι τιμές είναι 10 και 30
Το άθροισμα = 40
Οι τιμές είναι 10 και 30 και -829946796
Η τιμή = 10
```

ΚΑΘΟΡΙΣΜΟΣ ΑΚΡΙΒΕΙΑΣ ΣΤΗΝ ΕΚΤΥΠΩΣΗ

Ο καθορισμός του αριθμού των δεκαδικών ψηφίων που θα εκτυπωθούν για έναν πραγματικό αριθμό γίνεται π.χ. ως `%.3` -> 3 δεκαδικά ή `%.*` -> κανένα δεκαδικό. Παραδείγματα:

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

```
int main(void)
```

```
{
```

```
    float a = 1.2345;
```

```
    printf("Η τιμή = %f\n", a);
```

```
    printf("Η τιμή = %.3f\n", a);
```

```
    printf("Η τιμή = %.*f\n", 3, a);
```

```
    printf("Η τιμή = %.f\n", a);
```

```
    return 0;
```

```
}
```



./Ch2-6



```
Η τιμή = 1.234500
```

```
Η τιμή = 1.235
```

```
Η τιμή = 1.235
```

```
Η τιμή = 1
```

← (ισοδύναμος τρόπος)

← (έγινε στρογγυλοποίηση)

ΠΛΑΤΟΣ ΠΕΔΙΟΥ

Το **πλάτος πεδίου** είναι ο συνολικός αριθμός των χαρακτήρων που καταλαμβάνει η εκτύπωση στην οθόνη, με στοίχιση του αριθμού δεξιά. Αν τα συνολικά ψηφία του αριθμού ξεπερνούν το πλάτος πεδίου, τότε στην πράξη το πλάτος πεδίου αγνοείται.

```
#include <stdio.h>

int main(void)
{
    int a = 100;
    float b = 1.2345;

    printf("%10d\n", a);
    printf("%10f\n", b);
    printf("%10.3f\n", b);
    printf("%*.3f\n", 10, b);
    printf("%2d\n", a);
    printf("%2f\n", b);

    return 0;
}
```

(ισοδύναμος τρόπος)

(αριστερό
περιθώριο)

```
100
1.234500
1.235
1.235
100
1.234500
```