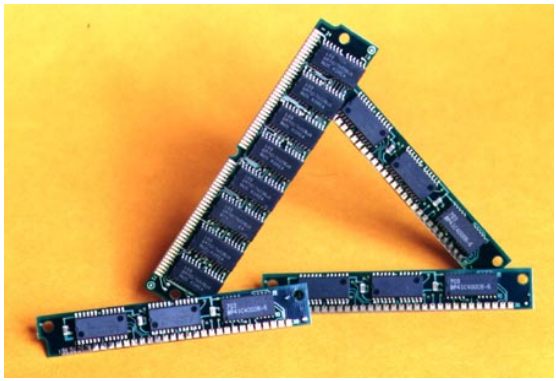
Προγραμματισμός Ι

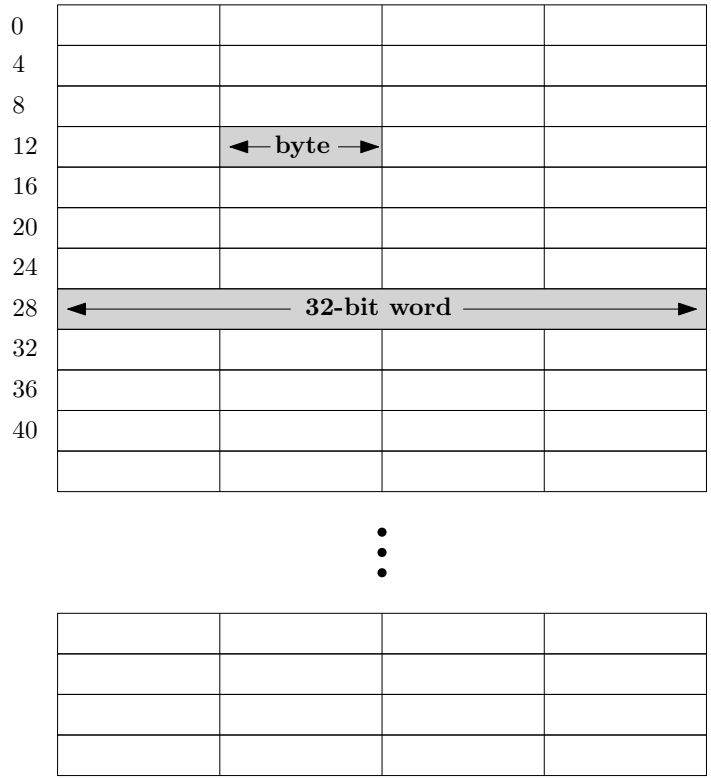
Μεταβλητές, Τύποι και Σταθερές

Δημήτρης Μιχαήλ

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Η Μνήμη του Υπολογιστή





Η μνήμη είναι σαν ένας πίνακας και μπορούμε να προσπελάσουμε ένα στοιχειο της μνήμης με έναν αριθμό. Στο παραπάνω παράδειγμα βλέπουμε την περίπτωση μιας 32-bit αρχιτεκτονικής.

Μεταβλητές στην C

Η γλώσσα C μας παρέχει ένα εύκολο τρόπο να προσπελάσουμε την μνήμη του υπολογιστή χωρίς να θυμόμαστε αριθμούς και διευθύνσεις.

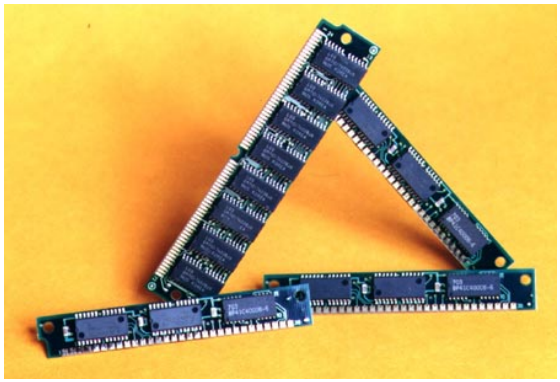
Μεταβλητές στην C

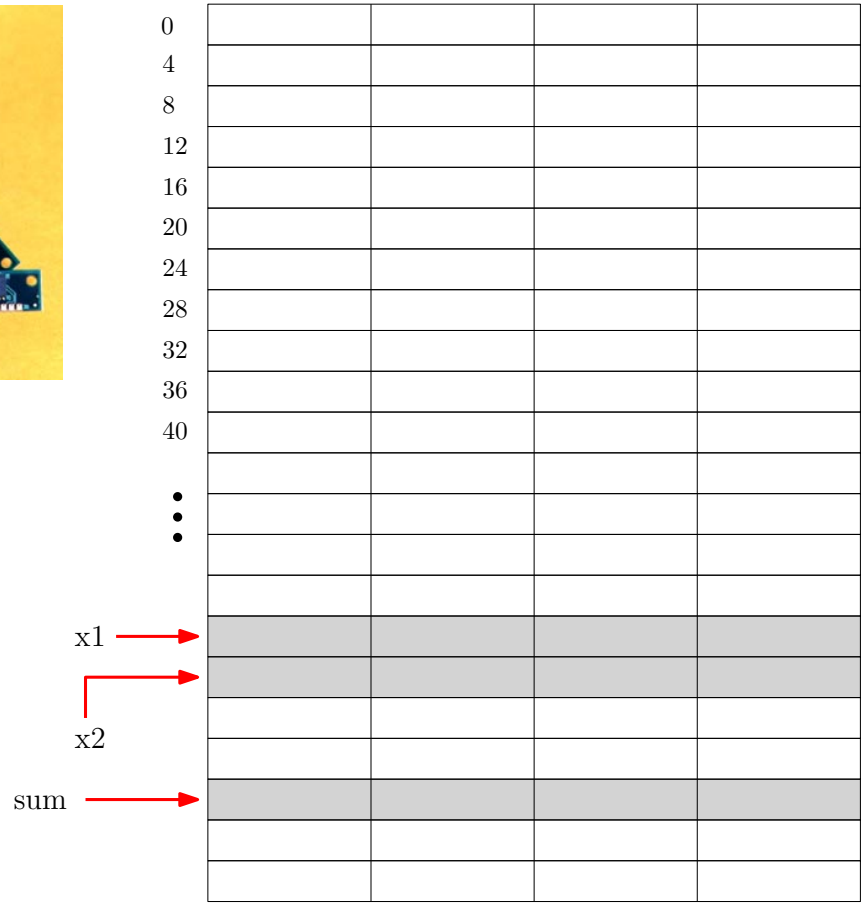
```c
/* Variable Example */
#include <stdio.h>

int main() {
    int x1, x2, sum;
}
```

Στο παραπάνω πρόγραμμα λέμε στον υπολογιστή να δεσμεύσει 3 θέσεις μνήμης για να αποθηκεύσουμε ακέραιους αριθμούς. Αυτές οι 3 θέσεις μνήμης πρέπει να ονομάζονται **`x1`**, **`x2`** και **`sum`** αντίστοιχα.

Ουσιαστικά ο compiler θυμάται τις διευθύνσεις με τα παραπάνω ονόματα.

Η Μνήμη του Υπολογιστή





Ουσιαστικά τα ονόματα είναι συντομεύσεις για τις διευθύνσεις.

Μεταβλητές στην C

```
1  /* Variable Example */
2  #include <stdio.h>
3
4  int main() {
5      int x1, x2, sum;
6  }
```

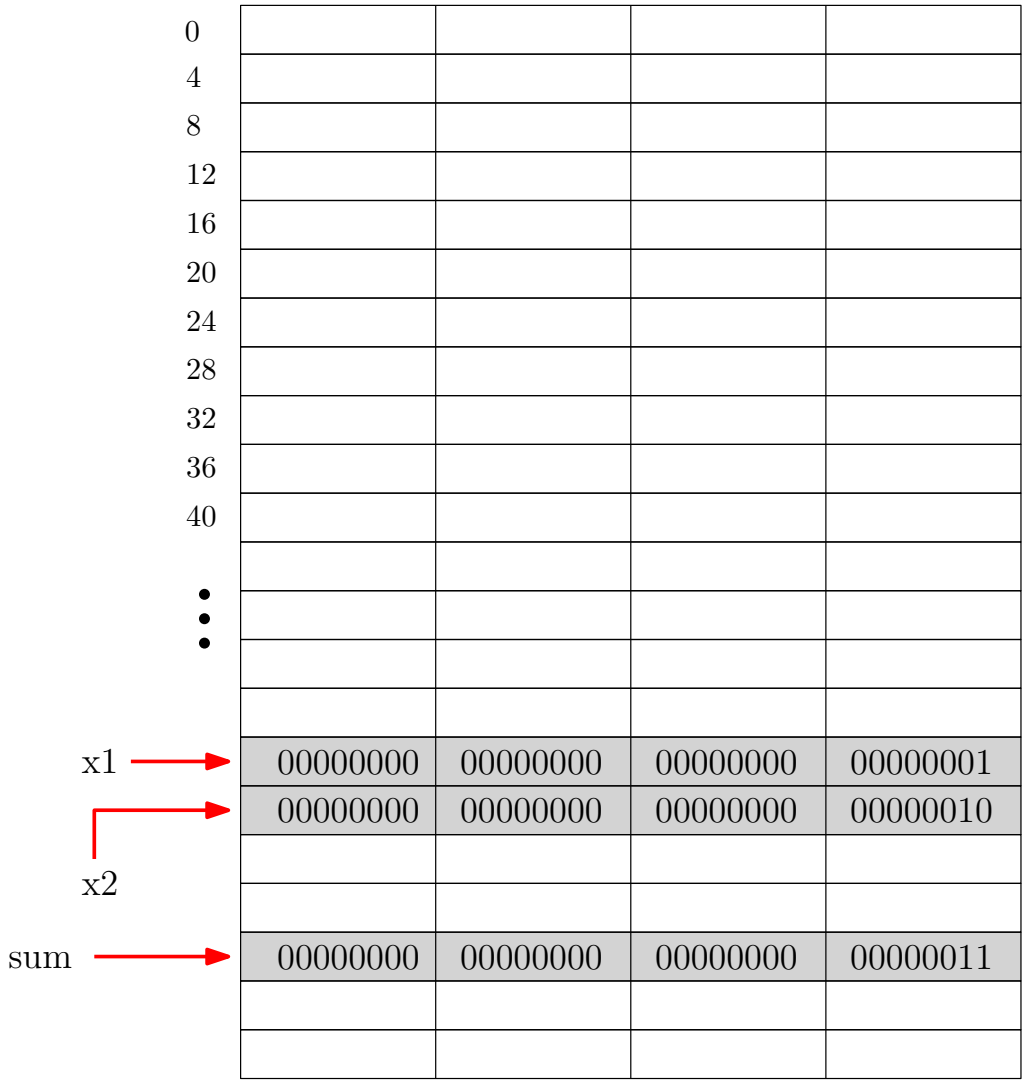
Οι θέσεις μνήμης που δεσμεύτηκαν δεν έχουν αρχικοποιηθεί, είναι καθήκον του προγραμματιστή να δώσει αρχικές τιμές.

Μεταβλητές στην C

```
1  /* Variable Example */
2  #include <stdio.h>
3
4  int main() {
5      int x1, x2, sum;
6
7      x1 = 1;
8      x2 = 2;
9      sum = x1 + x2;
10 }
```

Το παραπάνω πρόγραμμα αφού δεσμεύσει την μνήμη για τις μεταβλητές, αποθηκεύει την τιμή 1 στην θέση μνήμης x1, την τιμή 2 στην θέση μνήμης x2 και την τιμή 3 στην θέση μνήμης sum.

Μεταβλητές στην C



Ουσιαστικά τα ονόματα είναι συντομεύσεις για τις διευθύνσεις.

Μεταβλητές στην C

Κάθε μεταβλητή έχει:

1. όνομα
2. τύπο
3. τιμή

Μεταβλητές στην C

Όνομα

Για να είναι αποδεκτό από τον μεταγλωττιστή ένα όνομα στη C πρέπει να ξεκινάει με κάποιο χαρακτήρα (και όχι αριθμό), να μην περιέχει κενά και να μην έχει το ίδιο όνομα με κάποια συγκεκριμένα αλφαριθμητικά που χρησιμοποιεί η C όπως **`main`** (δεσμευμένες λέξεις).

λάθος ορισμός μεταβλητής

```
1 int main;
2 int 3x;
3 int hello world;
```

σωστός ορισμός μεταβλητής

```
1 int Main;
2 int x123456;
3 int hello_world;
```

Μεταβλητές στην C

Τύπος

Όταν ορίζουμε μία μεταβλητή λέμε στον μεταγλωττιστή τι είδους πληροφορία θα αποθηκεύσουμε στην θέση μνήμης που θα μας κρατήσει.

Βασικοί τύποι μεταβλητών

1. ακέραιοι: `int x`;
2. χαρακτήρας: `char x`;
3. κινητής υποδιαστολής (προσεγγιστική αναπαράσταση πραγματικών αριθμών): `float x`;

Υπάρχουν περισσότεροι τύποι δεδομένων. Θα τους δούμε με μεγαλύτερη λεπτομέρεια μόλις μάθουμε λίγα πράγματα για την αναπαράσταση αριθμών στους υπολογιστές.

Μεταβλητές στην C

Στην γλώσσα C οι μεταβλητές πρέπει να ορίζονται στην αρχή του κομματιού που θα χρησιμοποιηθούν. Για παράδειγμα

```
1  int main() {
2      int x, y, z; /* declare all variables */
3
4      x = 1;
5      y = 2;
6      z = x + y;
7
8      int i;          /* NO! */
9  }
```

Μεταβλητές στην C

Από την έκδοση C99 και μετά μπορούμε να ορίσουμε τις μεταβλητές και αργότερα.

```
1  int main() {
2      int x, y, z; /* declare some variables */
3
4      x = 1;
5      y = 2;
6      z = x + y;
7
8      int i;          /* YES! */
9
10 }
```

Ο γενικός κανόνας καλού προγραμματισμού, είναι πλέον, ότι δηλώνουμε τις μεταβλητές μας όσο πιο κοντά στο σημείο όπου θα τις χρησιμοποιήσουμε.

Ο τύπος `int`

Ο τύπος `int` αποθηκεύει ακέραιους και το μέγεθος του συχνά εξαρτάται από την αρχιτεκτονική του υπολογιστή στον οποίο βρισκόμαστε.

- πρέπει να μπορεί να πάρει όλες τις τιμές στο διάστημα $[-32767, 32767]$
- σε πολλές σύγχρονες αρχιτεκτονικές μπορεί να πάρει όλες τις τιμές στο διάστημα $[-2147483648, 2147483647]$

Ο τύπος `char`

Ο τύπος **`char`** χρησιμοποιείται για να αποθηκεύουμε χαρακτήρες.

```
1  int main() {
2      char c;
3
4      c = 'x';
5  }
```

Στην C οι χαρακτήρες κωδικοποιούνται με αριθμούς χρησιμοποιώντας την κωδικοποίηση ASCII.

Ο τύπος `char`

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]

Οι 95 εκτυπώσιμοι χαρακτήρες από 32 έως 126 (δεκαδικό). Οι χαρακτήρες από 0 έως 31 είναι ειδικοί χαρακτήρες για τον έλεγχο των συσκευών εξόδου, πχ το 8 είναι το BACKSPACE, το 13 είναι το ENTER και το 27 είναι το ESC.

`printf` και χαρακτήρες

```
1  #include <stdio.h>
2
3  int main() {
4     char c = 'a';
5
6     printf("character %c is number %d in ASCII encoding\n", c, c);
7  }
```

Το παραπάνω πρόγραμμα τυπώνει:

character a is number 97 in ASCII encoding

Ο τύπος `float`

Είναι ένας τύπος που αναπαριστά προσεγγιστικά τους πραγματικούς αριθμούς.

```
1  #include <stdio.h>
2
3  int main() {
4      float pi = 3.14159265;
5
6      printf("pi is approximately %f\n", pi);
7  }
```

Τύποι και Όρια Ακεραίων

τύπος	ελάχιστη τιμή	μέγιστη τιμή
`char`	≤ -127	$\geq +127$
`unsigned char`	≤ 0	$\geq +255$
`short int`	≤ -32767	$\geq +32767$
`unsigned short int`	≤ 0	$\geq +65535$
`int`	≤ -32767	$\geq +32767$
`unsigned int`	≤ 0	$\geq +65535$
`long`	≤ -2147483647	$\geq +2147483647$
`unsigned long`	≤ 0	$\geq +4294967295$

Οι σύγχρονοι compilers έχουν μεγαλύτερα όρια από αυτά που λέει το πρότυπο της ANSI C. Για παράδειγμα ο τύπος **`int`** έχει συνήθως τα όρια του **`long`** που φαίνονται παραπάνω.

Τύποι και Όρια Αριθμών Κινητής Υποδιαστολής

τύπος	ελάχιστη τιμή	μέγιστη τιμή
`float`	$\leq -1E-37$	$\geq 1E+37$
`double`	$\leq -1E-37$	$\geq 1E+37$

Οι σύγχρονοι compilers έχουν μεγαλύτερα όρια από αυτά που λέει το πρότυπο της ANSI C. Για παράδειγμα ο **gcc** στον υπολογιστή μου έχει όρια:

$$-1.175494E-38 \leq x \leq 3.402823E+38$$

για τύπο `float` και:

$$-2.225074E-308 \leq x \leq 1.797693E+308$$

για τον τύπο `double`.

Η `printf`

Η γενική μορφή της **`printf`**() φαίνεται παρακάτω:

```
int printf(const char * format, ...);
```

Το αλφαριθμητικό **`format`** περιέχει το κείμενο που θα εκτυπωθεί μαζί με ειδικές ακολουθίες χαρακτήρων που βοηθούν στην εκτύπωση των μεταβλητών που ακολουθούν. Οι ειδικές αυτές ακολουθίες έχουν την εξής γενική μορφή:

```
%[flags][width][.precision][length]specifier
```

Εκτός από τον specifier όλα τα άλλα στοιχεία είναι προαιρετικά.

Specifiers της `printf()`

Οι περισσότεροι specifiers φαίνονται στον παρακάτω πίνακα:

specifier	Έξοδος	Παράδειγμα
c	χαρακτήρας	a
d ή i	δεκαδικός αριθμός με πρόσημο	392
e	επιστημονικός συμβολισμός με e	3.9265e+2
E	επιστημονικός συμβολισμός με E	3.9265E+2
f	δεκαδικός αριθμός κινητής υποδιαστολής	392.65
o	οκταδικός με πρόσημο	610
s	αλφαριθμητικό	sample
u	δεκαδικός ακέραιος χωρίς πρόσημο	7235
x	δεκαεξαδικός αριθμός χωρίς πρόσημο	7fa
X	δεκαεξαδικός αριθμός χωρίς πρόσημο με κεφαλαία	7FA

Η γραμμή

```
printf("%d, %o and %x\n", 27, 27, 27);
```

εκτυπώνει 27, 33 and 1b.

`printf()` και padding

Το width είναι ένας αριθμός που λέει στην **`printf`** πόσα κενά να προσθέσει στην έξοδο ώστε το αποτέλεσμα να έχει τόσους πολλούς χαρακτήρες.

```
 1  #include <stdio.h>
 2  #include <stdlib.h>
 3
 4  int main()
 5  {
 6      int a = 100;
 7      int b = 1000;
 8      int c = 10000;
 9
10      printf("%7d\n", a);
11      printf("%7d\n", b);
12      printf("%7d\n", c);
13      printf("%7d\n", 1000000);
14  }
```

εκτυπώνει

```
    100
   1000
  10000
1000000
```

`printf()` και precision

Το precision έχει διαφορετική έννοια για κάθε τύπο:

- για ακέραιους τύπους υποδηλώνει τον ελάχιστο αριθμό ψηφίων που πρέπει να εκτυπωθούν (πιθανώς με μηδενικά στην αρχή)
- για αριθμούς κινητής υποδιαστολής υποδηλώνει τον αριθμό των ψηφίων μετά την υποδιαστολή
- για αλφαριθμητικά υποδηλώνει τον μέγιστο αριθμό χαρακτήρων που θα εκτυπωθούν

ακέραιοι και precision

Υποδηλώνει τον ελάχιστο αριθμό ψηφίων που πρέπει να εκτυπωθούν (πιθανώς με μηδενικά στην αρχή)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      int a = 100;
6      int b = 10000;
7      int c = 1000000;
8
9      printf( "%10.7d\n", a );
10     printf( "%.7d\n", b );
11     printf( "%.7d\n", c );
12
13     return 0;
14 }
```

εκτυπώνει

```
    0000100
0010000
1000000
```

float και precision

Υποδηλώνει τον ελάχιστο αριθμό ψηφίων που πρέπει να εκτυπωθούν (πιθανώς με μηδενικά στην αρχή)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main() {
5      float pi = 3.14159265;
6
7      printf( "%.3f\n", pi );
8      printf( "%.10f\n", pi );
9
10     return 0;
11 }
```

εκτυπώνει

```
3.142
3.1415927410
```

`printf()` και Ειδικοί Χαρακτήρες

Οι ειδικοί χαρακτήρες στην C φαίνονται παρακάτω:

χαρακτήρας	κωδικός ASCII	ειδικός χαρακτήρας στην C
newline	10	`'\n'`
tab	9	`'\t'`
carriage return	13	`'\r'`
backspace	8	`'\b'`
form feed	12	`'\f'`
backslash	92	`'\\'`
single quotation mark	39	`'\''`
double quotation mark	34	`'\"'`
null character	0	`'\0'`

για παράδειγμα ο παρακάτω κώδικας

```
printf("Very\tSimple\nExample");
```

τυπώνει

```
Very        Simple
Example
```

Σταθερές

Υπάρχουν διάφορα είδη σταθερών:

- κυριολεκτική σταθερά
- συμβολικές σταθερές
- δηλωμένες σταθερές με την χρήση του **`const`**

Σταθερές

Υπάρχουν διάφορα είδη σταθερών:

- κυριολεκτική σταθερά: πληκτρολογείται μέσα στον κώδικα

```
1  int count;
2
3  count = 3;
```

- συμβολικές σταθερές
- δηλωμένες σταθερές με την χρήση του **`const`**

Σταθερές

Υπάρχουν διάφορα είδη σταθερών:

- κυριολεκτική σταθερά
- συμβολικές σταθερές: ο προ-επεξεργαστής μας επιτρέπει να ορίσουμε συμβολοσειρές οι οποίες κατά την διάρκεια εκτέλεσης του, αντικαθίστανται με τις εκάστοτε τιμές.

```
1  #include <stdio.h>
2
3  #define MAX 100
4
5  int main()
6  {
7      int x = MAX;
8      printf( "%d\n", x );
9
10     return 0;
11 }
```

- δηλωμένες σταθερές με την χρήση του **`const`**

Σταθερές

Υπάρχουν διάφορα είδη σταθερών:

- κυριολεκτική σταθερά
- συμβολικές σταθερές
- δηλωμένες σταθερές με την χρήση του **`const`**:

  χρησιμοποιείται στις δηλώσεις των μεταβλητών και λέει στον μεταγλωττιστή πως μία μεταβλητή δεν θα αλλάξει ποτέ τιμή.

```
1  int main()
2  {
3      const int x = 3;
4  }
```

Σε αντίθετη περίπτωση ο compiler χτυπάει λάθος.