

# Προγραμματισμός I (HY120)

Διάλεξη 5:  
Κυριολεκτικά – Συνδυασμοί /  
Μετατροπές Τύπων – Αριθμητική  
Χαρακτήρων





# Κυριολεκτικά (literals)

- Κάποιες μεταβλητές του προγράμματος πρέπει συνήθως να **αρχικοποιηθούν** με **συγκεκριμένη τιμή**.
  - Υπάρχει επίσης η περίπτωση να επιθυμούμε να χρησιμοποιήσουμε μια ειδική συγκεκριμένη τιμή σε εκφράσεις αποτίμησης, π.χ. γνωστές σταθερές (π, e, ... ).
- Μια έκφραση που ορίζει μια **τιμή χωρίς αναφορά σε** κάποια **μεταβλητή** ονομάζεται «**κυριολεκτικό**».
  - Για κάθε τύπο δεδομένων ορίζονται διαφορετικές μορφές προσδιορισμού κυριολεκτικών.
- Η τιμή των κυριολεκτικών, καθώς και εκφράσεων που χρησιμοποιούν αποκλειστικά και μόνο κυριολεκτικά, είναι ήδη γνωστή πριν την εκτέλεση του κώδικα.



# Κυριολεκτικά char

- Η έκφραση της μορφής '<c>' συμβολίζει τον αντίστοιχο εκτυπώσιμο χαρακτήρα ASCII <c>.
- Η έκφραση '\<c>' συμβολίζει τον αντίστοιχο ειδικό μη εκτυπώσιμο χαρακτήρα ASCII (βλέπε manual).
- Η έκφραση '\x<d1d2>' συμβολίζει τον χαρακτήρα ASCII με τον αντίστοιχο δεκαεξαδικό κωδικό d1d2 (d1, d2 αριθμητικά ψηφία).
- Η έκφραση '\<d1d2d3>' συμβολίζει τον χαρακτήρα ASCII με τον αντίστοιχο οκταδικό κωδικό d1d2d3 (d1, d2, d3 αριθμητικά ψηφία).
  - Π.χ.:
    - 'a', '\x61', '\141'      χαρακτήρας a
    - '\n', '\x0A', '\012'      newline / linefeed



# Κωδικοποίηση ASCII

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	TAB	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2		!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

## ● Μερικά χρήσιμα κόλπα:

- Οι αλφαβητικοί χαρακτήρες έχουν διαδοχικές τιμές, αν ερμηνευτούν ως ακέραιοι.
- Το ίδιο και οι χαρακτήρες των δεκαδικών ψηφίων



# Κυριολεκτικά `int`

- Αν η έκφραση αρχίζει με δεκαδικό ψηφίο διαφορετικό του 0, τότε ερμηνεύεται με το δεκαδικό σύστημα.
  - Αν αρχίζει με 0 τότε ερμηνεύεται σύμφωνα με το οκταδικό σύστημα,
  - Αν αρχίζει με 0x τότε ερμηνεύεται σύμφωνα με το δεκαεξαδικό σύστημα.
- Εκφράσεις με κατάληξη `l` ή `L` ερμηνεύονται ως `long int`
  - Με κατάληξη `u` ή `U` ως `unsigned int`
  - Με κατάληξη `ul` ή `UL` ως `unsigned long int`
  - Διαφορετικά ως `int`.
- Π.χ.:
  - `0xFF` τιμή 255 (-1 ως `char`)
  - `32768u` τιμή 32768 (-32768 ως `short`)
  - `0123` τιμή 83

# Κυριολεκτικά `float` και `double`



6

- Η έκφραση πρέπει να δίνεται σε συμβατική μορφή, δηλαδή σε δεκαδικό σύστημα.
  - Προαιρετικά, μπορεί να δίνεται με δεκαδικά ψηφία ή/και με δεκαδικό εκθέτη που μπορεί να λαμβάνει και αρνητικό πρόσημο.
- Εκφράσεις με κατάληξη `f` ή `F` ερμηνεύονται ως `float`, ενώ με κατάληξη `l` ή `L` ως `long double`
  - Διαφορετικά ως `double`.

# Μετατροπές τιμών μεταξύ ΤΥΠΩΝ



7

- Γίνονται αυτόματα σε αποτιμήσεις και αναθέσεις:  
`char -> int, short -> int, float -> double`
  - Προσοχή στην μετατροπή `char -> int` καθώς γίνεται `sign extension` αν το `char` είναι `signed`.
- Αν ένα από τα ορίσματα μιας πράξης είναι `long double, double, float, long, unsigned, int` ΤΟΤΕ και το άλλο όρισμα «**προάγεται**» αντίστοιχα
- Όταν μια «μεγάλη» τιμή ανατίθεται σε «μικρότερη» μεταβλητή τότε **χάνεται μέρος των δεδομένων**.
- Ο προγραμματιστής μπορεί να εκβιάσει μια «παράνομη» μετατροπή με το λεγόμενο **type casting**.



# Μερικά Παραδείγματα

```
char c1='\xff'; unsigned char c2='\xff';  
short i1; unsigned short i2;  
  
i1 = i2 = c1;          /* i1,i2 γίνεται -1,65535 (=216-1) */  
i1 = i2 = c2;          /* i1,i2 γίνεται 255,255 */
```

```
double d1=10.0,d2; int i1=3,i2=10;  
  
d2 = d1 / i1;          /* d2 γίνεται 3.33... */  
d2 = (double)i2 / i1   /* d2 είναι 3.33... */  
d2 = (double)(i2 / i1); /* d2 είναι 3.0 */
```

```
int i; char c;  
  
i = 256;               /* i γίνεται 256 */  
c = i;                 /* c γίνεται 0 */  
i = c;                 /* i γίνεται 0 */
```





# Αριθμητική με Χαρακτήρες

- Ένας από τους «ιδιωτισμούς» της αυτόματης μετατροπής τύπων είναι η **αριθμητική με χαρακτήρες**.
- Μπορούμε να συνδυάσουμε ένα χαρακτήρα με ένα ακέραιο ή ένα χαρακτήρα με ένα χαρακτήρα:

```
'a' + 1           /* 98, 0x62, 'b' */
```

```
'b' - 'a'        /* 1 */
```

```
'5' - '3' + '0' /* 50, 0x32, '2' */
```

- Μια συνάρτηση που δέχεται παράμετρο ένα ακέραιο, μπορεί να δεχτεί σαν παράμετρο έναν χαρακτήρα

```
putchar(97);      /* 'a' */
```

```
putchar('a'+2);  /* 'c' */
```



# Παράδειγμα

```
/* ανάγνωση δύο δεκαδικών ψηφίων, υπολογισμός της
   διαφοράς τους και εκτύπωση της ως ακέραιος */
#include <stdio.h>

int main(int argc, char *argv[]) {

    char c1,c2;
    int diff;

    printf("enter two chars: ");
    c1 = getchar();
    c2 = getchar();

    diff = c1-c2;

    printf("%d\n",diff);
    return(0);
}
```