Φύση υπολογιστών

- Η λειτουργία των υπολογιστών βασίζεται σε ψηφιακά κυκλώματα
  - Οι είσοδοι και έξοδοι αυτών των κυκλωμάτων λαμβάνουν δύο δυνατές τιμές : high voltage, low voltage.
  - Διαφορετικά, true (1), false (0).
- Άρα, ο υπολογιστής καταλαβαίνει δύο καταστάσεις: 0 και 1

Γιατί να μην χρησιμοποιούνται περισσότερες καταστάσεις, πιο κοντά στην γλώσσα του ανθρώπου;

Δυνατότητες

- Αναπαράσταση πληροφορίας πολλών ειδών:
  - Λογικές τιμές (true ή false)
  - Αριθμούς (16, 2010, -33, 3.14, …)
  - Γράμματα (‘a’, ‘b’, …)
  - Διευθύνσεις υπολογιστών, e-mail, …
  - Εικονοστοιχεία (pixels), χρώματα, βάθος, κ.α.
    - 3 χρώματα Red, Green, Blue
    - 8 bits για κάθε χρώμα → 2^8 = 256 αποχρώσεις
    - 24 bits για κάθε χρώμα → 2^{24} = 16.777.216 αποχρώσεις
  - Μουσική, Βίντεο
- Διαχείριση με πολλούς τρόπους:
  - Ανάγνωση και εγγραφή
  - Αριθμητικές λειτουργίες
  - Λογικές λειτουργίες

Η πληροφορία στον υπολογιστή

- Αποθηκεύεται στην κύρια και στην δευτερεύουσα μνήμη
- Βασική μονάδα αποθήκευσης: δυαδικό ψηφίο (bit - binary digit)
  - byte → 1 byte = 8 bits
- Μονάδες μέτρησης χωρητικότητας
  - 1 Kilobyte (KB) = 2^{10} = 1024 bytes
  - 1 Megabyte (MB) = 2^{20} = 1024 KBs
  - 1 Gigabyte (GB) = 2^{30} = 1024 MBs
  - 1 Terabyte (TB) = 2^{40} = 1024 GBs
  - Petabyte, Exabyte, Zettabyte, Yottabyte

Αριθμητικά συστήματα

- **Δεκαδικό** (βάση 10, ψηφία 0, 1, 2, …, 9)
  - Κάθε θέση του αριθμού αναπαριστά μια δύναμη του 10

    3874 = 3 x 10^3 + 8 x 10^2 + 7 x 10^1 + 4 x 10^0
- **Δυαδικό** (βάση 2, ψηφία 0, 1)
  - Κάθε θέση του αριθμού αναπαριστά μια δύναμη του 2

    1110 = 1 x 2^3 + 1 x 2^2 + 1 x 2^1 + 0 x 2^0 = 14
- Δεκαδικό σε δυαδικό:
  - Διαιρούμε επαναληπτικά με 2 και κρατάμε το υπόλοιπο

    23/2 = 11 Υ = 1
    11/2 = 5 Υ = 1
    5/2 = 2 Υ = 1
    2/2 = 1 Υ = 0
    1/2 = 0 Υ = 1

    Το δυαδικό του 23 είναι το 10111

Αριθμητικά συστήματα

- **Οκταδικό** (βάση 8, ψηφία 0, 1, 2, …, 7)
  - Κάθε θέση του αριθμού αναπαριστά μια δύναμη του 8

    $7442_8 = \mathbf{7 \times 8^3 + 4 \times 8^2 + 4 \times 8^1 + 2 \times 8^0} = 3874_{10}$
  - Ένα οκταδικό ψηφίο → 3 δυαδικά

    $7442_8 = 111\ 100\ 100\ 010_2$
- **Δεκαεξαδικό** (βάση 16, ψηφία 0, 1, 2, …, 9, A, B, C, D, E, F)
  - Κάθε θέση του αριθμού αναπαριστά μια δύναμη του 16

    $F22_{16} = \mathbf{F \times 16^2 + 2 \times 16^1 + 2 \times 16^0} = 3874_{10}$
  - Ένα δεκαεξαδικό ψηφίο → 4 δυαδικά

    $F22_{16} = 1111\ 0010\ 0010_2$

Αριθμητικό σύστημα βάσης b

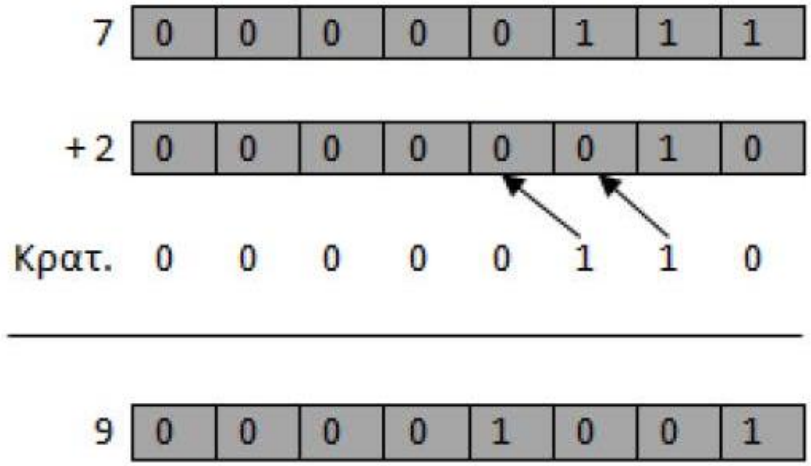
- Ψηφία 0, 1, 2, …, b-1
- Κάθε θέση του αριθμού αναπαριστά μια δύναμη του b, ανάλογη της θέσης
- Ένας αριθμός *n* ψηφίων με συντελεστές d_{n-1} d_{n-2} … d_1 d_0 έχει δεκαδικό ισοδύναμο:
  $d_{n-1}*b^{n-1} + d_{n-2}*b^{n-2} + \ldots + d_1*b^1 + d_0*b^0$
- Αν ο αριθμός περιλαμβάνει και **κλασματικό μέρος m ψηφίων** τότε έχουμε:
  $d_{n-1}*b^{n-1} + \ldots + d_0*b^0 + d_{-1}*b^{-1} + d_{-2}*b^{-2} + \ldots + d_{-m}*b^{-m}$

Αριθμοί σε διαφορετικά συστήματα

Δυαδικό	Οκταδικό	Δεκαεξαδικό	Δεκαδικό
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	A	10
1011	13	B	11
1100	14	C	12
1101	15	D	13
1110	16	E	14
1111	17	F	15

Αριθμητικές λειτουργίες

- **Πρόσθεση** δυο ακεραίων
- Ένας ακέραιος αποθηκεύεται συνήθως σε 2 ή 4 bytes στην μνήμη
- Από τα δεξιά προς τα αριστερά προσθέτουμε κάθε ζευγάρι ψηφίων
- Σημειώνουμε το άθροισμα και προσθέτουμε το κρατούμενο στην επόμενη στήλη



Αριθμητικές λειτουργίες

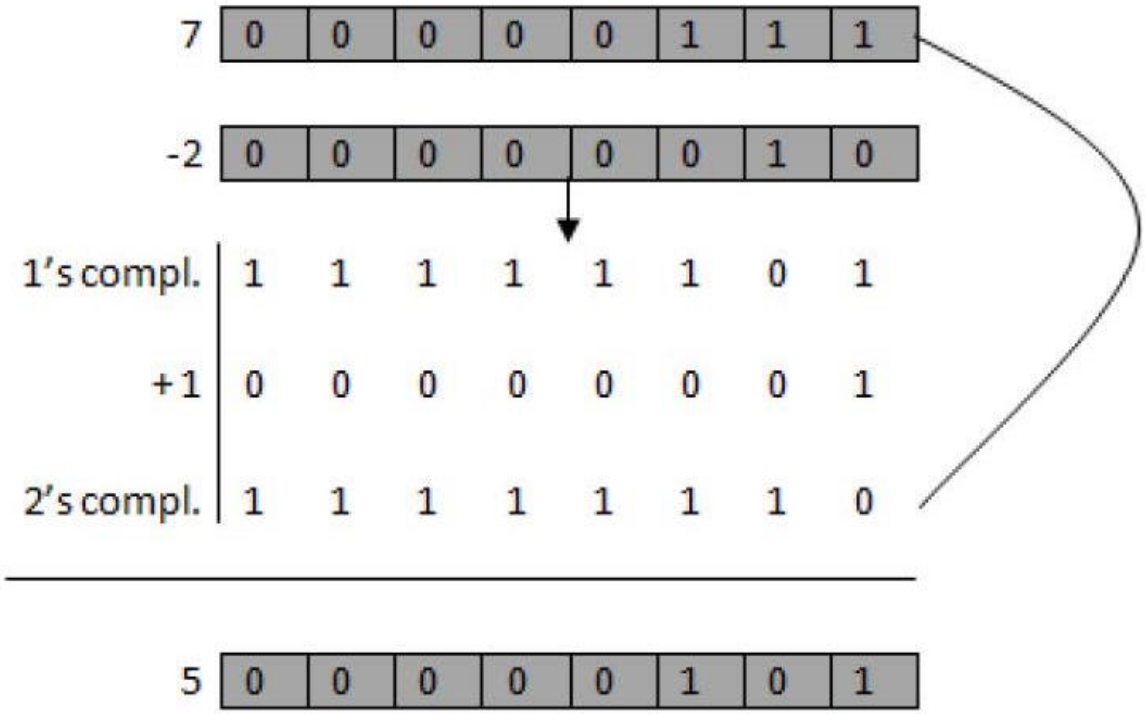
- Ποιος είναι ο μέγιστος (απρόσημος) ακέραιος που μπορεί να αναπαρασταθεί με 2 bytes σε έναν υπολογιστή;
- $2^{16} - 1 = 65.535$
- Τι θα γίνει αν το άθροισμα δυο αριθμών είναι μεγαλύτερο από το μέγιστο ακέραιο;
- Ότι συμβαίνει και με ένα 5-ψηφιο οδόμετρο ενός αυτοκινήτου που από το 99.999 θα επιστρέψει στο 0, 1, …
- Η αριθμητική αυτή ονομάζεται **module 2^n**
  - $A + B \rightarrow A + B \bmod 2^n$
  - $60.000 + 5.800 = 265$

Αριθμητικές λειτουργίες

- **Αφαίρεση** δύο ακεραίων
- Γίνεται με χρήση της πρόσθεσης και του **συμπληρώματος ως προς 2** του αφαιρετέου.
- **Α – Β = Α + Συμπλήρωμα_ως_προς_2(Β)**
- Ισχύει: **Συμπλήρωμα_ως_προς_2(Β) = Συμπλήρωμα_ως_προς_1(Β) + 1**
- Το Συμπλήρωμα_ως_προς_1(Β) υπολογίζεται αντιστρέφοντας κάθε bit του Β.

Αριθμητικές λειτουργίες

- Παράδειγμα αφαίρεσης (υποθέτουμε ακεραίους μήκους 8 bits)



Λογικές λειτουργίες σε bits

AND (&)

&	0	1
0	0	0
1	0	1

86	0	1	0	1	0	1	1	0
& 23	0	0	0	1	0	1	1	1
22	0	0	0	1	0	1	1	0

OR (|)

|	0	1
0	0	1
1	1	1

86	0	1	0	1	0	1	1	0
| 23	0	0	0	1	0	1	1	1
87	0	1	0	1	0	1	1	1

Λογικές λειτουργίες σε bits

XOR(^)

0 εάν ίδια bits

1 διαφορετικά

^	0	1
0	0	1
1	1	0

86	0	1	0	1	0	1	1	0
^ 23	0	0	0	1	0	1	1	1
65	0	1	0	0	0	0	0	1

Λογικές λειτουργίες σε bits

NOT(~)

Μετατροπή των 1 σε 0 και αντίστροφα

~	0	1
	1	0

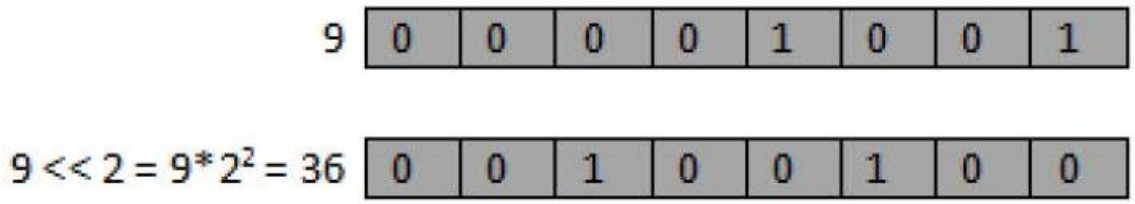
Δώστε μια έκφραση που μηδενίζει τα 3 πρώτα bits (LSBs) ενός αριθμού N για τον οποίο υποθέτουμε πως αποθηκεύεται σε 1 byte.

N = N & ~7

Λειτουργία shift left

shift left (<<)

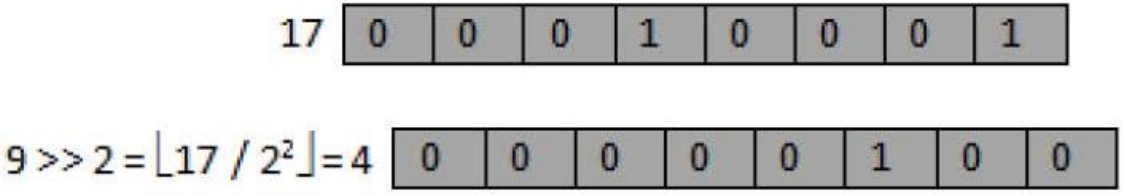
- Μετακινεί έναν αριθμό από bits προς τα αριστερά, γεμίζοντας τα κενά με 0.
- Πολλαπλασιάζει τον αρχικό αριθμό με δυνάμεις του 2



Λειτουργία shift right

shift right(>>)

- Μετακινεί έναν αριθμό από bits προς τα δεξιά, γεμίζοντας τα κενά με 0 (για απρόσημους αριθμούς).
- Διαιρεί τον αρχικό αριθμό με δυνάμεις του 2



Κωδικοποίηση συμβόλων

- Ομαδοποίηση bits για την αναπαράσταση συμβόλων (αριθμοί, χαρακτήρες, σημεία στίξης, κ.α.)
- Κλασικές κωδικοποιήσεις: ASCII, EBCDIC
  - Το ASCII χρησιμοποιεί 7 bits
  - Το EBCDIC χρησιμοποιεί 8 bits

ASCII

American Standard Code for Information Interchange

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

Το διάστημα 65-90 κωδικοποιεί τα κεφαλαία γράμματα. Το διάστημα 97-122 τα πεζά.

Ποια είναι η απόσταση κωδικοποίησης μεταξύ πεζών και κεφαλαίων;

Στοιχεία της Γλώσσας

- Αλφάβητο
  - Οι χαρακτήρες με τους οποίους σχηματίζονται οι λέξεις της γλώσσας
- Λεξιλόγιο
  - Οι λέξεις που χρησιμοποιεί η γλώσσα
- Συντακτικό
  - Οι κανόνες σύνταξης των προτάσεων της γλώσσας
- Σημασιολογία
  - Οι κανόνες ερμηνείας των προτάσεων της γλώσσας

Κατηγορίες λαθών

- Λεκτικά λάθη
  - Λανθασμένη χρήση του λεξιλογίου της γλώσσας
  - Ανιχνεύονται στη διάρκεια της μεταγλώττισης (φάση λεκτικής ανάλυσης)
- Συντακτικά λάθη
  - Οφείλονται σε παραβίαση των συντακτικών κανόνων της γλώσσας
  - Ανιχνεύονται στη διάρκεια της μεταγλώττισης (φάση συντακτικής ανάλυσης
- Λογικά λάθη
  - Οφείλονται στην κακή απόδοση της λύσης του προβλήματος
  - Ανιχνεύονται στο χρόνο εκτέλεσης (λάθος αποτελέσματα)

Λεξιλόγιο της C

- Δεσμευμένες λέξεις (reserved words ή keywords)
  - 32 λέξεις με ειδική σημασιολογία στη C που δεν μπορούν να χρησιμοποιηθούν για άλλο λόγο

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

Λεξιλόγιο της C

- Τελεστές (operators)
  - Αριθμητικοί (+, -, *, /, %)
  - Ανάθεσης (=, +=, -=, κ.ο.κ.)
  - Λογικοί (&&, ||)
  - Συσχετιστικοί (>, <, >=, <=, ==, !=)
  - Διαχείρισης bits (&, ^, |, <<, >>)
  - Μοναδιαίοι (*, &, -, !, ~, ++, --, sizeof)
- Αναγνωριστικά (identifiers)
  - Ονόματα μεταβλητών, συμβολικών σταθερών, συναρτήσεων που καθορίζονται από τον προγραμματιστή
  - Τα ονόματα των συναρτήσεων των βιβλιοθηκών της C δεν είναι μέρος του λεξιλογίου της γλώσσας, αλλά θεωρούνται εξωτερικά ονόματα

Λεξιλόγιο της C

- Σταθερές
  - ακέραιες (123, -78)
  - μεγάλου μήκους ακέραιες (45487L)
  - χαρακτήρα ('x', '\n')
  - Κινητής/σταθερής υποδιαστολής (3.6e-7, 5.3)
- Συμβολοσειρές
  - Μια ακολουθία χαρακτήρων που περικλείεται σε εισαγωγικά "…" ("This is a string")
- Διαχωριστικά (delimeters) όπως , και ;
- Οι λευκοί χαρακτήρες (κενό, tab, newline, σχόλια) αγνοούνται εκτός και εάν εξυπηρετούν άλλα σύμβολα

Ονόματα αναγνωριστικών

- Δεν πρέπει να είναι ίδια με τις δεσμευμένες λέξεις
- Ονόματα που χρησιμοποιούνται από τις βασικές βιβλιοθήκες δεν μπορούν να χρησιμοποιηθούν (λάθος διπλού ορισμού)
- Μπορούν να περιλαμβάνουν γράμματα, αριθμούς και το χαρακτήρα "_" (underscore)
- Πρέπει να αρχίζουν από γράμμα ή το χαρακτήρα "_" και όχι με αριθμητικό ψηφίο.
- Δεν μπορεί να περιέχει κενό χαρακτήρα.
- Διάκριση μεταξύ μικρών και κεφαλαίων (Age, age, AGE)
- Η ANSI C βάζει όριο μήκους τους 31 χαρακτήρες. Οι περισσότεροι σύγχρονοι compilers δεν εφαρμόζουν αυτό το όριο.

Παραδείγματα

name1 ~~1name~~ get_word ~~get-word~~ ~~int~~ mAxVel

Γενικοί κανόνες

- Αποφύγετε ονόματα ενός χαρακτήρα (i, j) εκτός από συγκεκριμένες περιπτώσεις (π.χ. counter σε for loop)
- Χρησιμοποιείτε εκφραστικά ονόματα
  - Αναπαράσταση ταχύτητας: velocity
  - Αναπαράσταση μέγιστης ταχύτητας: MAX_VELOCITY
  - Συνάρτηση εμφάνισης λάθους στην οθόνη: display_error ή displayError
- Χρησιμοποιείτε μικρά γράμματα για τις μεταβλητές και κεφαλαία για τις σταθερές και τις μακροεντολές

Αναπαράσταση δεδομένων

- Ένα υπολογιστικό πρόγραμμα περιλαμβάνει εντολές που επεξεργάζονται δεδομένα
  - Ένα δομημένο πρόγραμμα οργανώνεται σε ανεξάρτητες υπορουτίνες
- Τα δεδομένα εισόδου μπορεί να είναι αριθμητικά δεδομένα, γράμματα ή σύμβολα
- Μετά την επεξεργασία των δεδομένων προκύπτουν
  - Τα τελικά αποτελέσματα που περιμένει ο χρήστης
  - Ενδιάμεσα αποτελέσματα που χρησιμοποιούνται για την παραγωγή των τελικών
- Τα δεδομένα των προγραμμάτων οργανώνονται σε **μεταβλητές** και **σταθερές**

Μεταβλητές

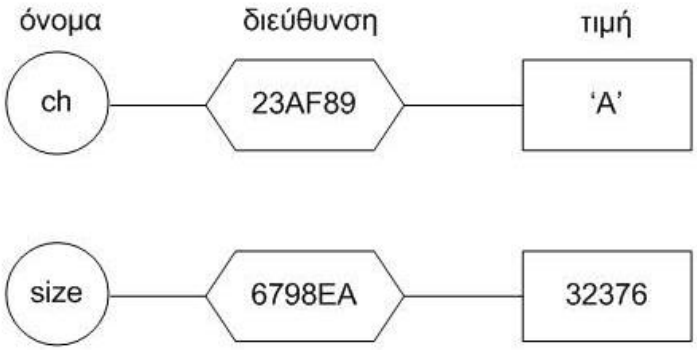
- Οι γλώσσες προγραμματισμού υποστηρίζουν την πρόσβαση στα δεδομένα μέσω συμβολικών ονομάτων που καλούνται μεταβλητές
- Χαρακτηριστικά μεταβλητής
  - Το όνομά της
  - Η διεύθυνσή της στη μνήμη (αναφορά)
  - Η τιμή της
- Το περιεχόμενο (τιμή) μιας μεταβλητής μπορεί να αλλάζει στη διάρκεια εκτέλεσης ενός προγράμματος

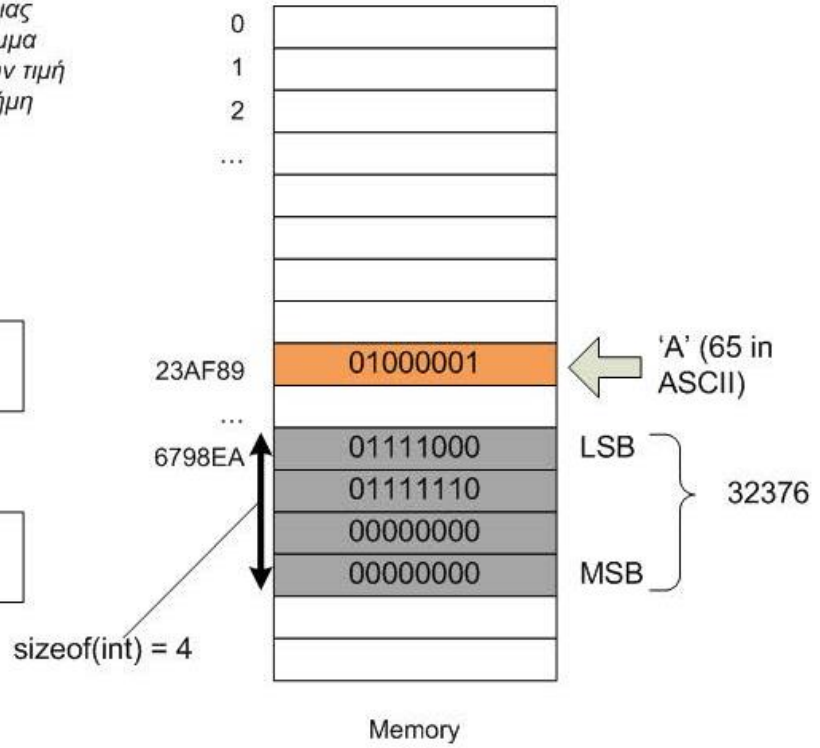
Ερμηνεία μεταβλητών

```
char ch;
int size;

ch = 'A';
size = 0;
…
size = 32376;
```

Η αναφορά του ονόματος μιας μεταβλητής σε ένα πρόγραμμα ισοδυναμεί με αναφορά στην τιμή που αποθηκεύεται στην μνήμη για αυτή την μεταβλητή.





Τύποι δεδομένων

- Ένας τύπος δεδομένων προσδιορίζει το πεδίο τιμών μιας μεταβλητής και τις πράξεις που μπορούν να γίνουν σε αυτή.
- Η C απαιτεί την δήλωση του τύπου των μεταβλητών από τον προγραμματιστή.
- Η δήλωση του τύπου μιας μεταβλητής προσδιορίζει τον αριθμό των bytes που θα δεσμευθούν στην μνήμη (εξαρτάται από την υλοποίηση).
- Βασικοί τύποι δεδομένων (**char**, **int**, **float**, **double**)
- Σύνθετοι τύποι (πίνακες, δομές)
  - int A[100];
  - struct Time { int hours; int minutes; int seconds; };

Τύποι δεδομένων

- Δηλώνοντας το σωστό τύπο για τις μεταβλητές
  - Επιτυγχάνουμε καλύτερη εκμετάλλευση της μνήμης
  - Επιτυγχάνουμε καλύτερο έλεγχο κατά τη μεταγλώττιση του προγράμματος

• μετρητής	int
• βάρος, μάζα, ταχύτητα	float, double
• όνομα, διεύθυνση, ΑΤ	char [] ή char* (string)
• λογική κατάσταση	unsigned char
• βαθμοί φοιτητών	float grades[NMAX]
• κόμβος λίστας	struct Node { .. }

Δήλωση μεταβλητών στη C

- <τύπος δεδομένων> <λίστα μεταβλητών>;

```
int count;
int count, num;
```

- <τύπος δεδομένων> <μεταβλητή>=<τιμή>;

```
int count = 20;
float num = 0.2;
```

Ακέραιος τύπος (short, int, long)

- Αναπαριστά ακεραίους (προσημασμένους)
- Υποστηρίζονται τρεις διαφορετικοί τύποι:
  - `short`
  - `int`
  - `long`
- Πεδίο τιμών για `int` ανάλογα με το μήκος λέξης (word size) του υπολογιστή

  Λέξη 16 bit: -32.768 έως +32.767

  Λέξη 32 bit: -2.147.483.648 έως +2.147.483.647

  2 ≤ bytes in `short` ≤ bytes in `int` ≤ bytes in `long`
- Τυπικά μεγέθη στους σύγχρονους υπολογιστές:
  - 2 bytes (`short`), 4 bytes (`int`), 4 bytes (`long`)

Αναπαράσταση ακέραιων σταθερών

- Οι άνθρωποι χρησιμοποιούν αριθμούς σε **δεκαδική μορφή**, αλλά οι προγραμματιστές μπορεί να χρησιμοποιούν αριθμούς σε **δυαδική**, **οκταδική** ή **δεκαεξαδική** μορφή
- Η αναπαράσταση των ακέραιων σταθερών περιλαμβάνει :
  - Δεκαδική μορφή ως default αναπαράσταση
  - Οκταδική μορφή με πρόθεμα **0** (μηδέν)
  - Δεκαεξαδική μορφή με πρόθεμα **0x**
  - Δεν υποστηρίζεται δυαδική αναπαράσταση
  - Η κατάληξη **L** υποδηλώνει σταθερές `long`
  - Για τις σταθερές `short` δεν χρησιμοποιείται κατάληξη, αλλά πρέπει να γίνει **cast** (μετατροπή τύπου)
- Παραδείγματα :
  - `int` : 2010, -2010, 03732, 0x7DA
  - `long` : 2010L, -2010L, 03732L, 0x7DAL
  - `short` : (short)2010, (short)-2010, (short)03732, (short)0x7DA

Απρόσημος ακέραιος τύπος

- Αναπαριστά απρόσημους ακεραίους
- Υποστηρίζονται τρεις διαφορετικοί τύποι:
  - `unsigned short`
  - `unsigned int`
  - `unsigned long`
- Οι απρόσημοι ακέραιοι μπορεί να αποθηκεύσουν διπλάσιους θετικούς αριθμούς από τους προσημασμένους
  - 32 bit (`int`): 0 έως 2.147.483.647
  - 32 bit (`unsigned int`): 0 έως 4.294.967.295
- Οι απρόσημοι ακέραιοι διευκολύνουν τις αριθμητικές λειτουργίες επάνω σε bits
- Η μίξη προσημασμένων και απρόσημων ακεραίων στην ίδια έκφραση μπορεί να περιπλέξει τις μετατροπές τύπων
- Η γενική οδηγία σε μεικτές εκφράσεις είναι να μετατρέπονται οι προσημασμένοι σε απρόσημους.

Αναπαράσταση απρόσημων

- Οι ακέραιες σταθερές είναι προσημασμένες by default
- Η κατάληξη **U** υποδηλώνει απρόσημη σταθερά
- Οι ακέραιοι που εκφράζονται σε οκταδική ή δεκαεξαδική μορφή αυτόματα θεωρούνται απρόσημοι
- Παραδείγματα :
  - `unsigned int`: 2010U, 03732, 0x7DA
  - `unsigned long`: 2010UL, 03732L, 0x7DA
  - `unsigned short`: (short)2010U, (short)03732, (short)0x7DA

Τύπος **char**

- Αναπαριστά ατομικούς χαρακτήρες
- Η τιμή εσωκλείεται με αποστρόφους (single quotes)
  - `'C'`, `'2'`, `'*'`, `','`
- Λέξη-κλειδί: `char`
  - `char choice='A';`
  - `char x, y;`
- Μέγεθος 1 byte (συνήθως, εύρος τιμών -128 έως 127)
  - Σε μερικές υλοποιήσεις ο τύπος char σημαίνει signed char
  - Σε μερικές υλοποιήσεις ο τύπος char σημαίνει unsigned char
- Στη C κάθε χαρακτήρας αναπαρίσταται ως ακέραιος με τιμή τον αντίστοιχο κωδικό ASCII

ASCII

American Standard Code for Information Interchange

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
16	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
32	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
96	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	DEL

- '0' : ASCII 48
- 'A' : ASCII 65
- 'Z' : ASCII 90
- 'a' : ASCII 97
- 'z' : ASCII 122
- '*' : ASCII 42
- '=' : ASCII 61
- '~' : ASCII 126

Ειδικές τιμές χαρακτήρων

'\0'	ASCII 0	NUL
'\a'	ASCII 7	BEL
'\b'	ASCII 8	BS (backspace)
'\f'	ASCII 12	FF (form feed)
'\n'	ASCII 10	LF (newline)
'\r'	ASCII 13	CR (carriage return)
'\t'	ASCII 9	HT (horizontal tab)
'\v'	ASCII 11	VT (vertical tab)
'\\'	ASCII 92	backslash
'\''	ASCII 39	single quote

Επιτρεπτές τιμές χαρακτήρων

```
char ch;


ch = 'T'; // character form
ch = (char)84;  // decimal form
ch = (char)0124; // octal form
ch = (char)0x54; // hexadecimal form
ch = '\o124'; // octal character form
ch = '\x54'; // hexadecimal character form
ch = (char)127; // largest decimal
ch = (char)-128; // smallest decimal
```

Αριθμητική με χαρακτήρες

```
char ch;


ch = 'a' + 1; // ch is 'b'
ch = '0' + 7; // ch is '7'
ch = 'a' - 32; // ch is 'A'
```

Τύπος **unsigned char**

- Λέξη-κλειδί: `unsigned char`
  - `unsigned char choice = 'A';`
  - `unsigned char x, y;`
- Μέγεθος 1 byte μνήμης (εύρος τιμών 0 έως 255)
- Μπορεί να χρησιμοποιηθεί για να αποθηκεύει μικρούς θετικούς ακεραίους
- Για να υλοποιηθεί ένα πρωτόκολλο επικοινωνίας με μια περιφερειακή συσκευή (bit-map command)
- Διευκολύνει την αριθμητική επάνω σε bits (π.χ. left/right shift)
- Χρησιμοποιείται σε εφαρμογές γραφικών για την αναπαράσταση χρωμάτων (RGB = 3 unsigned chars)

Τύποι πραγματικών αριθμών

- Τύπος **float**
- Τύπος **double**
- Τύπος **long double**

Τύπος **float**

- Αναπαριστά πραγματικούς αριθμούς θετικούς ή αρνητικούς (κινητής υποδιαστολής **απλής** ακρίβειας)
  - Αναπαράσταση σταθερής υποδιαστολής (ακρίβεια συγκεκριμένων δεκαδικών ψηφίων) π.χ., 0.012
  - Αναπαράσταση κινητής υποδιαστολής (αυξημένη ακρίβεια δεκαδικών ψηφιών με επιστημονικό συμβολισμό) π.χ. 6.3E-05
- Μέγεθος εξαρτώμενο από την μηχανή (**συνήθως 4 bytes**)
- Λέξη-κλειδί: `float`
  - `float x;`

Αναπαράσταση float σταθερών

Χρησιμοποιούμε την κατάληξη **F** για να υποδηλώσουμε float σταθερά, διαφορετικά υπονοείται double.

Παραδείγματα :

123.456F 1.23456E2F

.0123456F 1.234546E-2F

-123.456F -1.23456E2F

3.402823E38F μεγαλύτερη τιμή (κατά προσέγγιση)

-3.402823E38F μικρότερη τιμή (κατά προσέγγιση)

1.175494E-38F τιμή που τείνει στο 0 (κατά προσέγγιση)

Αναπαράσταση double σταθερώ

Ο deafault τύπος αριθμών κινητής υποδιαστολής (σταθερές που περιέχουν υποδιαστολή ή το Ε).

Παραδείγματα :

123.456	1.23456E2
.0123456	1.234546E-2
-123.456	-1.23456E2
1.797693E308	μεγαλύτερη τιμή (κατά προσέγγιση)
-1.797693E308	μικρότερη τιμή (κατά προσέγγιση)
2.225074E-308	τιμή που τείνει στο 0 (κατά προσέγγιση)

Τύπος **long double**

- Αναπαριστά πραγματικούς αριθμούς θετικούς ή αρνητικούς (κινητής υποδιαστολής **εκτεταμένης** ακρίβειας)
  - Διπλάσια ακρίβεια δεκαδικών ψηφίων σε σχέση με double.
- Μέγεθος εξαρτώμενο από την μηχανή (**συνήθως 12 bytes**)
- Λέξη-κλειδί: `long double`
  - `long double x;`

Αναπαράσταση long double

Χρησιμοποιούμε την κατάληξη **L** για να υποδηλώσουμε long double σταθερά, διαφορετικά υπονοείται double.

Παραδείγματα :

123.456L 1.23456E2L

.0123456L 1.234546E-2L

-123.456L -1.23456E2L

1.189731E4932L μεγαλύτερη τιμή (κατά προσέγγιση)

-1.189731E4932L μικρότερη τιμή (κατά προσέγγιση)

3.362103E-4932L τιμή που τείνει στο 0 (κατά προσέγγιση)

Δήλωση σταθερών

- Μια δηλωμένη σταθερά είναι ένα όνομα που αντιπροσωπεύει μια τιμή η οποία **δεν αλλάζει** κατά την εκτέλεση του προγράμματος.
- Χαρακτηριστικά σταθεράς
  - Το όνομά της
  - Σε αντίθεση με την μεταβλητή η σταθερά δεν έχει διεύθυνση στη μνήμη (αναφορά)
  - Η τιμή της
- Ο μεταγλωττιστής αντικαθιστά το όνομα με την τιμή
- Η χρήση των σταθερών συνεισφέρει:
  - Στην αναγνωσιμότητα των προγραμμάτων
  - Στην ευκολία τροποποίησης των προγραμμάτων

Δήλωση σταθερών

Η δήλωση σταθερών μπορεί να γίνει με 3 τρόπους:

1. Με **#define**

   ```
   #define <όνομα> <τιμή>
   ```

   Παραδείγματα:

   ```
   #define PI 3.14159
   #define TRUE 1
   ```

2. Με **const μεταβλητή**

   ```
   const <τύπος δεδομένων> <μεταβλητή>=<τιμή>;
   ```

   Παραδείγματα:

   ```
   const double PI = 3.14159;
   const unsigned char TRUE = 1;
   ```

3. Με δομή **απαρίθμησης (enumeration)**

   ```
   enum <απαριθμητής> {<συμβολικό_όνομα>[=<τιμή>], … };
   ```

   Παράδειγμα:

   ```
   enum Month {JAN=1, FEB=2, … , DEC=12};
   ```

Ιδιότητες δήλωσης σταθερών

- Τα #define έχουν εμβέλεια αρχείου
  - Τα #define να ορίζονται στην αρχή του αρχείου και όχι εντός του ορισμού συναρτήσεων
- Τα const δεν λειτουργούν για μεγέθη πινάκων

```
const int maxSize = 100;
...
int Queue[maxSize]; /* Compiler error */
```

- Με enum μπορούν να δηλωθούν μεταβλητές

```
/* Define a type named "enum GameValues" */
enum GameValues {ROCK, SCISSORS, PAPER};
/* Declare "val" to be a variable of type "enum GameValues".
enum GameValues val;
...
val = ROCK; /* val = 0 */
```

- Το enum δεν δουλεύει για άλλους τύπους εκτός int

```
enum  MathConstants { E = 2.71828, PI = 3.14159 }; /* Compiler error */
```

Συμβουλές δήλωσης σταθερών

- Χρησιμοποιήστε enumerations για να δώσετε συμβολικά ονόματα σε ακέραιες σταθερές
- Χρησιμοποιήστε const μεταβλητές για να δώσετε συμβολικά ονόματα σε μη-ακέραιες σταθερές
- Αποφύγετε την χρήση των #define

Τελεστές (operators)

- Σύμβολα ή λέξεις που αναπαριστούν μία συγκεκριμένη λειτουργία που εκτελείται σε ένα ή περισσότερα δεδομένα
- Τα δεδομένα καλούνται τελεστέοι ή έντελα (operands) και μπορεί να είναι:
  - Μεταβλητές
  - Σταθερές
  - Κλήση συνάρτησης που επιστρέφει τιμή
  - Έκφραση (με χρήση παρενθέσεων)
- Οι τελεστές χρησιμοποιούνται για το σχηματισμό εκφράσεων:

```
num + 12
num1 > num2
(a+b)/c+sqrt(x)
```

Κατηγορίες τελεστών

Σε σχέση με τον αριθμό των τελεστέων επί των οποίων εφαρμόζονται :

- Μοναδιαίοι (unary)

  –9 , !x, ++a, &val, …

- Δυαδικοί (binary)

  x + y, x<<1, a=z, …

- Τριαδικοί (ternary)

  (a>b)?1:0

Κατηγορίες τελεστών

Την λειτουργία που εκτελούν :

- Αριθμητικοί : **++**, —, **+**(unary), –(unary), *, **/**, **%**, **+**, –
- Ανάθεσης : **=**, **+=**, –=, *=, **/=**, **%=**
- Συσχετιστικοί : **<**, **<=**, **>**, **>=**, **==**, **!=**
- Λογικοί : **!**, **&&**, **||**
- Διαχείρισης δεικτών : **[]**, *, **&**
- Διαχείρισης δομών :→, .
- Διαχείρισης bits : **~**, **<<**, **>>**, **&**, **^**, **|**, **&=**, **^=**, **|=**, **<<=**, **>>=**
- Κλήση συνάρτησης : **()**
- Ρητή μετατροπή τύπου : **(type)**
- Μέγεθος τύπου: **sizeof**
- Τριαδική έκφραση συνθήκης: **?:**
- Ακολουθία : **,**

Προσεταιριστικότητα τελεστών

- Αριστερά-προς-τα-δεξιά (τ = τελεστής)

  a τ b τ c ⇔ (a τ b) τ c

  τ a τ b ⇔ (τ a) τ b

- Δεξιά-προς-τα-αριστερά

  a τ b τ c ⇔ a τ (b τ c)

  τ a τ b ⇔ τ a (τ b)

Τελεστής ανάθεσης

- Στο αριστερό μέρος καταχωρείται η τιμή του δεξιού μέρους
- <μεταβλητή> = <έκφραση>;

```
mikos = 5;
embadon = mikos * platos;
x = x + 1;
x = y = 0.0; /* πολλαπλή ανάθεση */
while ((i = getchar()) != EOF) …
/* Διάβασμα χαρακτήρα. Ανάθεση στο i.
Υπολογισμός αυτού του χαρακτήρα λόγω ().
Σύγκριση του χαρακτήρα με EOF.
Υπολογισμός σε 0 (FALSE) ή 1 (TRUE). */
```

- Ο τύπος της μεταβλητής πρέπει να είναι συμβατός με το αποτέλεσμα της έκφρασης

Ειδικοί τελεστές

- Μοναδιαίος αύξησης/μείωσης (++, --)

```
++i; ή i++; → i = i + 1;
Όμως:
i = 5;
x = ++i; /* x=6, i=6 */
x = i++; /* x=5, i=6 */
--x; ή x--; → x = x - 1;
```

- Σύνθετης ανάθεσης (+=, -=, *=, /=)

```
x += 10; → x = x +10;
```

- Υπό συνθήκη (? :)

```
<έκφρ1> ? <εκφρ2> : <εκφρ3>;
(a > b) ? a : b;
```

Αν a > b τότε a, αλλιώς b

- sizeof(έκφραση): επιστρέφει τον αριθμό των bytes που καταλαμβάνει η έκφραση

```
sizeof(int)
sizeof(x+y)
```

Παράδειγμα

```
• int x = 10;
• int y = 20;
• ++x;
• y = --x;
• y += x--;
• y -= x++  ;
```

x	y
11	20
10	10
9	20
10	11

Εκφράσεις

- Συνδυασμός ενός ή περισσοτέρων τελεστέων και ενός ή περισσοτέρων τελεστών
- Αριθμητικές (αποτέλεσμα αριθμητικού τύπου)

```
(5*x+y/4)*8
```

- Σύγκρισης (αποτέλεσμα λογικού τύπου)

```
x==3      a!=b     (x+y) >= 4
```

- Λογικές (αποτέλεσμα λογικού τύπου)

```
(x<5) && (x>=1)          (x==0) || (y==0)
```

Τύπος έκφρασης

Καθορίζεται από τους τύπους των τελεστών

Έκφραση	**Αποτέλεσμα**
int τ int	int
int τ char	int
int τ double	double
double τ double	double

Υπολογισμός εκφράσεων

- Προτεραιότητα
  - Χωρισμός των τελεστών σε ομάδες διαφορετικού επιπέδου προτεραιότητας

    (+, -): χαμηλότερο επίπεδο

    (*, /): υψηλότερο επίπεδο

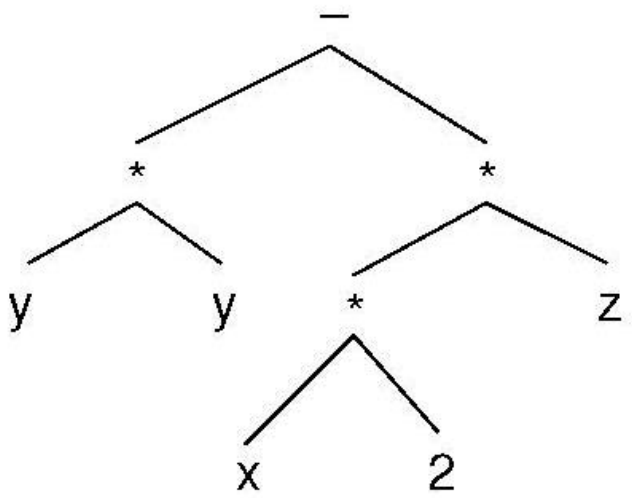
    `x-y*z` → `x-(y*z)`
- Προσεταιριστικότητα
  - Καθορισμός κατεύθυνσης εφαρμογής τελεστών ίδιας προτεραιότητας
  - Αριστερή προσεταιριστικότητα:

    10+8-2 → (10+8)-2
  - Δεξιά προσεταιριστικότητα:

    num1=num2=10 → num1=(num2=10)

Δέντρο αφηρημένης σύνταξης

- y*y-x*2*z → (y*y)-((x*2)*z)



Μετατροπές τύπων

- Υπονοούμενες (implicit)
  - Ο στενότερος τύπος μετατρέπεται στον ευρύτερο
  - (char < int < long < float < double < long double)

    1/2 → 0

    f = i = 3.3; → f →3.0, i → 3
- Ρητές (explicit)

  (<τύπος δεδομένων>)έκφραση

  (float)2 → 2.0

  (float)3/2 → 1.5

  (float)(3/2) → 1.0