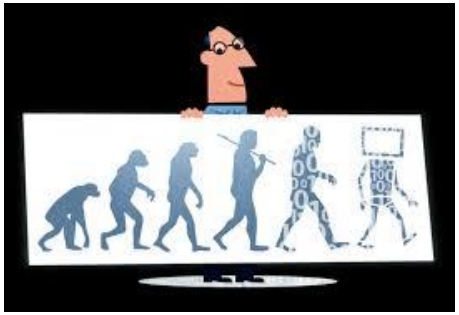
# *Δομημένος προγραμματισμός*



*https://commons.wikimedia.org/*

Βασικοί τύποι δεδομένων,
η συνάρτηση printf (),
αριθμητικοί τελεστές.

# Περίγραμμα της ύλης που θα καλυφθεί

- Δεδομένα, μεταβλητές, σταθερές
- Τύποι δεδομένων
  - char, int, float
- Η συνάρτηση printf()
- Τελεστές αριθμητικών πράξεων
- Ρητή μετατροπή τύπων (casting)

*https://www.publicdomainpictures.net*

# Τα δεδομένα

- Βασικές λειτουργίες ενός προγράμματος:
  - η επεξεργασία δεδομένων και
  - η εξαγωγή αποτελεσμάτων (δηλαδή άλλα δεδομένα).
- Τα δεδομένα (data) είναι απαραίτητα στοιχεία
- Απαιτείται λοιπόν δέσμευση *χώρων μνήμης* για να αποθηκευτούν αυτά τα δεδομένα, κατά την διάρκεια της εκτέλεσης του προγράμματος.
- Οι γλώσσες προγραμματισμού μας δίνουν την δυνατότητα να δηλώσουμε κάποιους *τύπους* χώρων μνήμης, για την αποθήκευση δεδομένων (*τύποι δεδομένων*).
- Αυτοί οι χώροι μνήμης, ονομάζονται ανάλογα με την χρήση τους:
  - *σταθερές* ή
  - *μεταβλητές*.

# Χώροι μνήμης

Δήλωση στη C

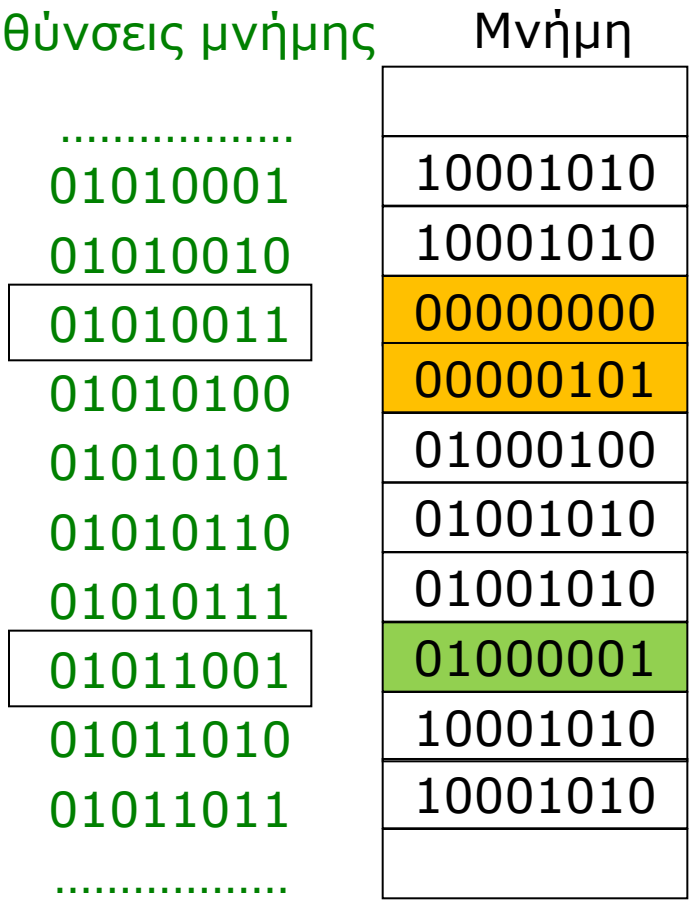
```
int   NUM = 5 ;
```

Η διεύθυνση του NUM (&NUM) →

Δήλωση στη C

```
char LET = 'A' ;
```

Η διεύθυνση του LET (&LET) →




- Με τη C μπορούμε να δώσουμε ονόματα στους χώρους μνήμης και να καταχωρίσουμε δεδομένα.
- Ο τελεστής & επιστρέφει την διεύθυνση μιας μεταβλητής.

# Οι σταθερές

- Οι σταθερές χρησιμοποιούνται όταν θέλουμε να χειριστούμε δεδομένα τα οποία δεν αλλάζουν (ή δεν πρέπει ν' αλλαχθούν) κατά την διάρκεια της εκτέλεσης του προγράμματος.
- Συνήθως καθορίζονται μια φορά και χρησιμοποιούνται όσο συχνά επιθυμούμε, κατά την διάρκεια λειτουργίας του προγράμματος.
- Με τη δήλωση της σταθεράς, πρέπει να της εκχωρηθεί και μια αρχική τιμή.
- Ένας τρόπος να ορίσουμε σταθερές στη C είναι με την μακροεντολή #define του προεπεξεργαστή:

```
#define  PI  3.14159
```

Εντολή της C

- Προσοχή: αυτή η εντολή δεν τελειώνει με το ερωτηματικό «;». Επίσης, υπάρχει κενό μεταξύ της σταθεράς PI και του 3.14159.
- Ένας άλλος τρόπος δήλωσης σταθερας είναι με την λέξη const:

```
const  int alpha = 10;
```

// Είναι εντολή της C

- Αν επιχειρήσουμε ν’ αλλάξουμε το περιεχόμενο μιας σταθεράς θα εμφανιστεί μήνυμα λάθους στη μεταγλώττιση.

# Οι μεταβλητές

- Αντίθετα με τις σταθερές, οι τιμές των μεταβλητών είναι δυνατόν να αλλάξουν κατά την διάρκεια της εκτέλεσης ενός προγράμματος.
- Όταν γίνονται αλλαγές στο περιεχόμενο μιας μεταβλητής, χρησιμοποιείται πάντα η *τελευταία τιμή* της και όχι κάποια από τις προηγούμενες τιμές της.
- Οι γλώσσες προγραμματισμού μας δίνουν την δυνατότητα να καθορίσουμε μεταβλητές και στην συνέχεια να τους δώσουμε όποια τιμή επιθυμούμε (δεδομένο), για επεξεργασία στο πρόγραμμα.
- Η δήλωση (δέσμευση χώρου) των μεταβλητών γίνεται συνήθως στην αρχή του προγράμματος (πριν χρησιμοποιηθεί μια μεταβλητή πρέπει να έχει δηλωθεί).
- Σύνταξη της δήλωσης μιας μεταβλητής:

  *τύπος  όνομα_μεταβλητής;*

- Κάθε μεταβλητή χαρακτηρίζεται από το όνομα της, τον τύπο δεδομένων και τη διεύθυνση της.

# Οι μεταβλητές

- Αντίθετα με τις σταθερές, οι τιμές των μεταβλητών είναι δυνατόν

***Παραδείγματα δήλωσης:***

***int alpha;*** *//δήλωση μεταβλητής με όνομα alpha*

*/*δέσμευση χώρου μνήμης για τη φιλοξενία των τιμών (δεδομένα) που μπορεί πάρει η μεταβλητή alpha, που πρέπει να είναι τύπου int*/*

***float delta = 5.6;*** *//δήλωση μεταβλητής με όνομα delta*

*/*δέσμευση χώρου μνήμης για τη φιλοξενία των τιμών (δεδομένα) που μπορεί πάρει η μεταβλητή delta, που πρέπει να είναι τύπου float */*

***char flag;*** *//δήλωση μεταβλητής με όνομα flag*

*/*δέσμευση χώρου μνήμης για τη φιλοξενία των τιμών (δεδομένα) που μπορεί πάρει η μεταβλητή flag, που πρέπει να είναι τύπου char */*

δεδομένων και τη διεύθυνσή της.

# Τιμές σε μεταβλητές με τον τελεστή =

- Ένας τρόπος να δώσουμε κάποια τιμή σε μια μεταβλητή είναι με χρήση του τελεστή εκχώρησης που είναι το σύμβολο της ισότητας στα μαθηματικά (**=**).
- Η γενική μορφή της εντολής εκχώρησης είναι:

*όνομα_μεταβλητής = έκφραση*

*όπου η έκφραση μπορεί να είναι μια σταθερή τιμή, μια άλλη μεταβλητή, ένας μαθηματικός τύπος, το αποτέλεσμα κλήσης μιας συνάρτησης κλπ. Φυσικά η έκφραση θα πρέπει να παράγει ένα αποτέλεσμα που να είναι συμβατό με τον τύπο της μεταβλητής.*

- Παραδείγματα:

```
index = 53; //ανάθεση της τιμής 52 στη μεταβλητή index

length = 14.56; //ανάθεση της τιμής 14.56 στη μεταβλητή length

letter = 'b'; //ανάθεση της τιμής b στη μεταβλητή letter

beta = (a-3)/(b+2); //ανάθεση της τιμής της έκφρασης στη μεταβλητή
```

# Ο τελεστής εκχώρησης

- Έχει βέβαια διαφορετική έννοια απ' ότι στα μαθηματικά.

  *Για παράδειγμα, η εντολή:*

  *var2 = 52;*

  *σημαίνει ότι δίνω την τιμή 52 στην μεταβλητή var2*

- Με την εντολή:

  *var2 = var2 +1;*

  *"αυξάνω" κατά μία μονάδα την τιμή που ήδη υπάρχει καταχωρημένη στην μεταβλητή var2, δηλαδή η νέα τιμή της var2 μετά την εκτέλεση της συγκεκριμένης εντολής θα είναι 53!*

- Μπορούμε να κάνουμε πολλαπλές εκχωρήσεις:

  *x=y=z=74;*

  *Οι αναθέσεις γίνονται από δεξιά προς τα αριστερά, δηλαδή πρώτα παίρνει την τιμή 74 η μεταβλητή z, μετά η μεταβλητή y και τέλος η x.*

***Προσοχή:*** *η εντολή x=y=74=z; θα δώσει σφάλμα στη μεταγλώττιση (ανάθεση τιμής στο 74)*

# Ονομασίες μεταβλητών

- Οι ονομασίες των μεταβλητών πρέπει να αρχίζουν με γράμμα ή με κάτω παύλα «**_**». Δεν μπορούν ν' αρχίζουν με αριθμό.
- Στην περιγραφή της ονομασίας μπορείτε να χρησιμοποιείστε και αριθμούς, αλλά δεν επιτρέπονται άλλοι ειδικοί χαρακτήρες όπως:

  *#, &, *, +, -, % κλπ,*

  οι οποίοι χρησιμοποιούνται για άλλο σκοπό.
- Τα πεζά και κεφαλαία γράμματα παίζουν ρόλο (είναι διαφορετικά!).
- Ο αριθμός χαρακτήρων που χρησιμοποιούνται για την ονομασία μεταβλητών διαφέρει από compiler σε compiler.
- Χρησιμοποιείστε εκφραστικές ονομασίες (*tasi_pigis, varos* κλπ) που περιγράφουν τα δεδομένα.
- Αν η ονομασία μιας μεταβλητής περιλαμβάνει πάνω από μια λέξη καλό είναι αυτές να χωρίζονται με τη κάτω παύλα
  πχ *bathmos_foititi, ypsos_paidiou, misthos_ypallilou* κλπ.
- Δεν θα πρέπει να χρησιμοποιείστε ονόματα που είναι δεσμευμένες λέξεις για την γλώσσα C.

# Οι δεσμευμένες λέξεις της standard C

| DATA TYPES | STORAGE CLASSES | STATMENTS |
|---|---|---|
| char | auto | break |
| double | extern | case |
| enum | register | continue |
| float | static | default |
| int | | do |
| long | | else |
| short | | for |
| struct | | goto |
| union | | if |
| unsigned | | return |
| void | | switch |
| sizeof | | while |
| typdef | | |

*Ανάλογα με τον compiler που χρησιμοποιείτε, είναι δυνατόν να υπάρχουν και άλλες δεσμευμένες λέξεις της C (cdel, const, far, fortran, huge, near, pascal, signed, volatile κ.α).*

# Οι τύποι δεδομένων στη C

- Η αναπαράσταση δεδομένων στο υπολογιστή διαφέρει, ανάλογα με την επεξεργασία που θέλουμε να κάνουμε.
- Το μέγεθος της μνήμης που δεσμεύεται για κάποιο τύπο είναι περιορισμένο. Άρα οι τιμές των δεδομένων που «φιλοξενούνται» σε κάθε τύπο είναι επίσης περιορισμένες.
- Το εύρος των διαφόρων τύπων δεδομένων μπορεί να είναι διαφορετικό από compiler σε compiler και από υπολογιστή σε υπολογιστή.
- Βασικοί τύποι δεδομένων:

  *χαρακτήρες (char), ακέραιοι (int), κινητής υποδιαστολής (float)*

- Σύνθετοι τύποι δεδομένων:

  *πίνακες, δείκτες, ενώσεις, δομές*

- Για τους βασικούς τύπους δεδομένων υπάρχουν ακόμα:
  - Χαρακτηρισμός πρόσημου: *singed / unsigned*
  - Προσδιορισμός μεγέθους: *short, long (ακέραιοι) double (κινητής υποδιαστολής).*

# Ο τύπος δεδομένων char

- Ο τύπος char χρησιμοποιείται για να αποθηκεύσει χαρακτήρες που εμφανίζονται:

  *A-Z, a-z, 0-9, !, @, #, $, %, ^, &, *, (, ), _, -, +, /, κλπ,*

  καθώς και χαρακτήρες ελέγχου που δεν εκτυπώνονται, όπως η αλλαγή γραμμής, <*enter*>, <*backspace*> κλπ.

- Εύρος*: από -128 έως 127* (με πρόσημο) ή από *0 έως 255* (1 byte, 8bits)
- Παραδείγματα καθορισμού (δήλωσης):

```
char ch1;           // Δήλωση της μεταβλητής ch1
unsigned char ch2;  //Δήλωση της μεταβλητής ch2 – χωρίς πρόσημο
char ch3='A';       //Δήλωση & αρχική τιμή
```

- Παραδείγματα χρήσης:

```
ch='D';
printf("the character is: %c", ch);
```

- Ο τύπος δεδομένων *char* είναι στη πραγματικότητα ακέραιος (int).
  - Δηλαδή, μπορείτε να κάνετε πράξεις με μεταβλητές τύπου char.

# Ο τύπος δεδομένων int

- Τύπος δεδομένων για τον χειρισμό ακεραίων αριθμών,

  με πρόσημο *(..., -2, -1, 0, 1, 2, 3, …)* ή χωρίς πρόσημο (*0, 1, 2, 3,...*)

- Παραδείγματα καθορισμού (δήλωσης):

```
int alpha;              //δήλωση μεταβλητής
short  metritis=0;  //δήλωση και ανάθεση αρχικής τιμής
unsigned int  arithmos_foititon;
```

- Εύρος *short* (2 bytes):

  *Με πρόσημο (signed): από -32.768 έως 32.767 ($2^{15}$-1),*

  *Χωρίς πρόσημο (unsigned): από 0 έως 65.535 ($2^{16}$-1).*

- Εύρος *long* (4 bytes):

  *Με πρόσημο: συνήθως από -2.147.483.648 έως 2.147.438.647 ($2^{31}$-1)*

  *Χωρίς πρόσημο: συνήθως από 0 έως 4.294.967.295 ($2^{32}$-1)*

- Εύρος *int*: συνήθως ταυτίζεται με τον short ή το long. Γενικά:

  $long \geq int \geq short$ (εξαρτάται από τον compiler).

# Ο τύπος δεδομένων int

- Για τον χειρισμό ακεραίων αριθμών με πρόσημο *(..., -2, -1, 0, 1, 2, 3, …)* ή χωρίς πρόσημο (*0, 1, 2, 3,...*)
- Π
- Π
- Π

***Παραδείγματα χρήσης:***

```
int alpha, beta, delta;  //δήλωση 3 μεταβλητών
alpha=5;  //εκχώρηση τιμής στη  μεταβλητή με όνομα alpha
/* εμφάνιση στη οθόνη της τιμής της μεταβλητής alpha
printf("the integer is:%d", alpha);
delta=alpha;  //η delta παίρνει την τιμή της alpha
```

*-1)*

- Εύρος *int*: συνήθως ταυτίζεται με τον short ή το long. Γενικά:

  *long ≥ int ≥ short* (εξαρτάται από τον compiler).

# Ο τύπος δεδομένων κινητής υποδιαστολής

- Τύπος δεδομένων *float / double*
- Χρησιμοποιείται για τον χειρισμό πραγματικών αριθμών (περιλαμβάνουν υποδιαστολή: *34.5, 23.0, 0.987*).
- Παραδείγματα καθορισμού (δήλωσης):

```
float mesos_oros;
float  tasi = 5.3;
double error_value;
double metrisi = 1.2052e+02
```

Προσοχή:
Τελεία (.), όχι κόμμα (,)

Το **E** ή **e** αναπαριστά το 10. Ο αριθμός που ακολουθεί είναι η θετική ή αρνητική δύναμη του 10.

- Εύρος *float* (4 bytes):

  συνήθως από *1.175494e-38* έως *3.402823e+38*

- Εύρος *double* (8 bytes):
  - από *2.225074e-308* έως *1.797693e+308* (8 bytes)

# Ο τύπος δεδομένων κινητής υποδιαστολής

- Τύπος δεδομένων *float / double*
- Χρησιμοποιείται για τον χειρισμό πραγματικών αριθμών (περιλαμβάνουν υποδιαστολή: *34.5, 23.0, 0.987*).
- [illegible]
  - Χρησιμοποιούμε τον τύπο ***float*** όταν η ακρίβεια των δεκαδικών ψηφίων δεν είναι τόσο σημαντική.
  - Χρησιμοποιούμε τον τύπο ***double*** όταν χρειαζόμαστε υψηλή ακρίβεια των δεκαδικών ψηφίων.
  - Σε ορισμένους compilers υπάρχει και ο προσδιοριστής ***long.***
- [illegible]
- Τύπος *double* (8 bytes):
  - από *2.225074e-308* έως *1.797693e+308* (8 bytes)

# Παρατηρήσεις (αναθέσεις τιμών)

- Αν μπροστά από μια ακέραια τιμή υπάρχει το ψηφίο 0, τότε αυτή η τιμή εκλαμβάνεται σαν οκταδικός αριθμός:

```
int alpha = 0100;  //το alpha είναι 64 και όχι 100
```

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

```
int beta = 0x10;   // το beta είναι 16 και όχι 10
```

- Η τιμή που δίνεται σε μια μεταβλητή πρέπει να συμβαδίζει με τον τύπο της μεταβλητής:

```
int alpha = 12.3;  //το alpha γίνεται 12
```

- Η τιμή που εκχωρείται σε μια μεταβλητή πρέπει να είναι μέσα στο επιτρεπτό εύρος τιμών:

```
unsigned char ch = 340;  //δεν γίνεται σωστή ανάθεση
```

- Μπορούμε να δώσουμε ακέραια τιμή σε μια float μεταβλητή:

```
float beta=50;   //συνήθως είναι το ίδιο με float beta = 50.0
```

# Ο τελεστής sizeof()

Επειδή το εύρος των διαφόρων τύπων δεδομένων είναι δυνατόν να είναι διαφορετικό από compiler σε compiler με την χρήση του τελεστή sizeof (που μπορεί να μοιάζει με συνάρτηση, αλλά δεν είναι) μπορείτε να δείτε το εύρος κάθε τύπου δεδομένων στο compiler που χρησιμοποιείτε.

```
#include <stdio.h>
  main()
  {
  printf("\nSize of type char: %d bytes", sizeof(char));
  printf("\nSize of type int:  %d bytes", sizeof(int));
  printf("\nSize of type short: %d bytes", sizeof(short));
  printf("\nSize of type long: %d bytes", sizeof(long));
  printf("\nSize of type unsigned char: %d bytes", sizeof(unsigned char));
  printf("\nSize of type unsigned int: %d bytes", sizeof(unsigned int));
  printf("\nSize of type float: %d bytes", sizeof(float));
  printf("\nSize of type double: %d bytes", sizeof(double));
  getchar();
}
```

Η *sizeof()* υπολογίζει το μέγεθος κάποιας μεταβλητής ή σταθεράς, αλλά μπορεί να πάρει σαν παραμέτρους και τύπους δεδομένων

# Η συνάρτηση printf()

- Η γενική μορφή της συνάρτησης είναι:

  *printf (ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, στοιχείο-1, στοιχείο-2,... , στοιχείο-v)*

- Τα *στοιχείο-i* (i=1,2,..v) είναι ονόματα μεταβλητών, σταθερές, τιμές, ή εκφράσεις.

- Η *ΣΕΙΡΑ_ΕΛΕΓΧΟΥ* περικλείεται μέσα σε λατινικά διπλά εισαγωγικά (") και καθορίζει τον τρόπο εμφάνισης των δεδομένων. Η ΣΕΙΡΑ_ΕΛΕΓΧΟΥ μπορεί να περιλαμβάνει:
  - *επεξηγηματικό κείμενο*, οτιδήποτε κείμενο θέλουμε που περιγράφει το αποτέλεσμα
  - *προσδιοριστές*, που μπαίνουν απαραίτητα μέσα στην ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, μόνο όταν θέλουμε να εμφανίσουμε κάποιο στοιχείο (μεταβλητές, σταθερές, τιμές ή εκφράσεις) και
  - *χαρακτήρες ελέγχου*, που τους χρησιμοποιούμε για την διαμόρφωση της εμφάνισης.

- Η *printf()* σαν συνάρτηση, επιστρέφει το αριθμό των χαρακτήρων που τυπώνονται.

# Η ΣΕΙΡΑ ΕΛΕΓΧΟΥ της printf()

- Το τι περιλαμβάνει η *ΣΕΙΡΑ_ΕΛΕΓΧΟΥ* εξαρτάται από το τι θέλουμε να κάνουμε με την printf().
- Αν θέλουμε απλά να εμφανίσουμε ένα μήνυμα, η *ΣΕΙΡΑ_ΕΛΕΓΧΟΥ* θα περιλαμβάνει μόνο επεξηγηματικό κείμενο:

```
printf ("Hello World");
```

- Αν θέλουμε να παρουσιάσουμε το περιεχόμενο κάποιων μεταβλητών μόνο, τότε η ΣΕΙΡΑ_ΕΛΕΓΧΟΥ θα πρέπει περιλαμβάνει ένα προσδιοριστή για κάθε μεταβλητή που πρόκειται να παρουσιάσουμε:

```
printf("%d , %d ", x, y);   // δυο μεταβλητές, δυο προσδιοριστές
```

- Αν θέλουμε να συνοδέψουμε την εμφάνιση των μεταβλητών με κάποιο κείμενο, τότε η ΣΕΙΡΑ_ΕΛΕΓΧΟΥ θα πρέπει να περιλαμβάνει και τους προσδιοριστές και το κείμενο:

```
printf("The value of alpha is: %f", alpha);
```

- Αν επιπλέον επιθυμούμε την διαμόρφωση της εμφάνισης αυτών που πρόκειται να παρουσιάσουμε, τότε η ΣΕΙΡΑ_ΕΛΕΓΧΟΥ, θα πρέπει να περιλαμβάνει και κάποιους χαρακτήρες ελέγχου:

```
printf("x=%d /n y=%f \n ", x, y);  //  το \n είναι χαρακτήρας ελέγχου
```

# Οι προσδιοριστές στη ΣΕΙΡΑ ΕΛΕΓΧΟΥ

- Οι προσδιοριστές των στοιχείων έχουν σχέση με τον τύπο των δεδομένων που εμφανίζονται στην printf() και αρχίζουν με τον χαρακτήρα «%».
- Ο μεταγλωττιστής αντιστοιχίζει ένα-προς-ένα, από αριστερά προς τα δεξιά, τα ονόματα των μεταβλητών (ή σταθερών, τιμών, εκφράσεων) με τους προσδιοριστές.

```
printf ("An integer: %d a float: %f and a char%c\n", 1977, 3.1416, 'a');
```

- Αν οι προσδιοριστές είναι περισσότεροι από τις μεταβλητές, τότε για τα πρόσθετα προσδιοριστικά εμφανίζονται τυχαίες τιμές.
- Αν οι προσδιοριστές λιγότεροι από τις μεταβλητές, τότε δεν εμφανίζονται οι τιμές των πρόσθετων μεταβλητών.

# Πίνακας των προσδιοριστών της printf

| Προσδιοριστής | Περιγραφή |
|---|---|
| *%d ή %i* | Ακέραιος /int |
| *%c* | Χαρακτήρας/char (1 μόνο) |
| *%s* | Σειρά χαρακτήρων (συμβολοσειρά) |
| *%f* | Κινητής υποδιαστολής, δηλαδή float ή double |
| *%e ή %E* | Κινητής υποδιαστολής, δηλαδή float ή double σε εκθετική μορφή |
| *%g ή %G* | Κινητής υποδιαστολής όπως το %e ή %f (όποιο είναι μικρότερο) |
| *%u* | Ακέραιος χωρίς πρόσημο (unsigned int) |
| *%p* | Χρησιμοποιείται για την εμφάνιση διέθυνσης μνήμης |
| *%o* | Ακέραιος σε οκταδικό σύστημα (χωρίς πρόσημο) |
| *%x ή %X* | Ακέραιος σε δεκαεξαδικό σύστημα (χωρίς πρόσημο) |

# Δυνατότητες εμφάνισης με τους προσδιοριστές

Στην εμφάνιση μιας μεταβλητής μπορούμε να καθορίσουμε το συνολικό πλήθος των χαρακτήρων μαζί με τα ψηφία ακρίβειας και την υποδιαστολή.

- Για παράδειγμα το *%6d*, σημαίνει ότι το δεδομένο είναι ακέραιος και ότι το εύρος του είναι 6 ψηφία (χαρακτήρες).
- Εξ' ορισμού (by default) εμφανίζονται *6* δεκαδικά ψηφία μετά την υποδιαστολή. Μπορούμε όμως να προσδιορίσουμε πόσα ψηφία θέλουμε.
- Το *%7.2f*, σημαίνει ότι το δεδομένο είναι κινητής υποδιαστολής και έχει εύρος 7 χαρακτήρες με 2 ψηφία μετά την υποδιαστολή (4 χαρακτήρες για το ακέραιο μέρος, 1 χαρακτήρας για την υποδιαστολή και 2 χαρακτήρες για την ακρίβεια).
- Η τιμή μιας πραγματικής μεταβλητής στρογγυλοποιείται προς τα πάνω ή προς τα κάτω.
- Ο χαρακτήρας «**+**» μπροστά από τον προσδιοριστή (για παράδειγμα *%+d*) εμφανίζει το πρόσημο.
- ο χαρακτήρας «**-**» (πχ *%-d*) στοιχίζει αριστερά το αποτέλεσμα, ενώ το μηδέν (πχ *%08d*) γεμίζει με μηδέν τα κενά που είναι άδεια μπροστά από το αποτέλεσμα.
- Λόγω της ειδικής σημασίας του χαρακτήρα %, για την εμφάνιση του πρέπει να γραφούν δύο χαρακτήρες % (*%%*).

# Οι χαρακτήρες ελέγχου (πίνακας)

| char | Περιγραφή |
|---|---|
| \a | Ηχητικό σήμα (<BELL>) |
| \b | Ο χαρακτήρας <BACKSPACE> (διάστημα πίσω) |
| \f | Ο χαρακτήρας νέας σελίδας (<FORM FEED>) |
| \n | Νέας γραμμής (<LINE FEED>) |
| \r | Επιστροφής (<CR>) |
| \t | Οριζοντίου προκαθορισμένου διαστήματος (<TAB>) |
| \v | Κατακορύφου διαστήματος( <VTAB>) |
| \' | Εμφάνιση του απλού εισαγωγικού |
| \" | Εμφάνιση του διπλόυ εισαγωγικού |
| \\ | Εμφάνιση της ανάποδης πλαγίας καθέτου |
| \? | Εμφάνιση του λατινικού ερωτηματικού |
| \xhhh | Εμφάνιση του χαρακτήρα hhh σε δεκαεξαδικό αριθμό |
| \ooo | Εμφάνιση του χαρακτήρα ooo σε δεκαεξαδικό αριθμό |

Όποιος χαρακτήρας ακολουθεί τον χαρακτήρα "\", τότε επισημαίνεται στο compiler ότι αυτός ο δεύτερος χαρακτήρας έχει ειδική σημασία. Οι συνδυασμοί αυτών των δύο χαρακτήρων λαμβάνονται υπόψη σαν ένας χαρακτήρας και χρησιμοποιούνται για τον συμβολισμό των ειδικών χαρακτήρων.

# Παραδείγματα με την printf()

| ΕΝΤΟΛΕΣ ΤΗΣ C | ΑΠΟΤΕΛΕΣΜΑΤΑ |
|---|---|
| *index=5;*<br>*printf(" %d", index);* | 5 |
| *index=5;*<br>*printf("The value of variable index is: %d", index);* | The value of variable index is: 5 |
| *index=5;*<br>*printf("The value of variable index is:\n %d", index);* | The value of variable index is:<br>5 |
| *x=5;*<br>*y=3;*<br>*printf("%d + %d = %d", x, y, (x+y));* | 5+3 = 8 |
| *x=5;*<br>*y=3;*<br>*printf("x=%d \n y=%d \n x+y= %d", x, y, (x+y));* | x=5<br>y=3<br>x+y=8 |
| *x=5;*<br>*y=3;*<br>*printf(“adding x=%d, and y=%d, we get:\n sum= %d", x, y, (x+y));* | x=5<br>adding x=5 and y=3, we get:<br>sum=8 |

# Παράδειγμα με προσδιοριστές

```
#include <stdio.h>
main()
{
        int int_num;
        double flt_num;
        int_num = 123;
        flt_num = 123.456789;
        printf(“Integer format:    %d\n", int_num);
        printf("With precision specifier:  %08d\n", int_num);
        printf(“Float format:      %f\n", flt_num);
        printf("With precision specifier:  %-10.2f\n", flt_num);
}
```

**Αποτέλεσμα εκτέλεσης**

```
Integer format:     123
With precision specifier:  00000123
Float format:      123.456789
With precision specifier:  123.46
```

# Παίζοντας με την printf ()

```
#include <stdio.h>
 main ()
{
   int i = 123;
   double f = 3.1415926535;
   printf ("i = %i\n", i);
   printf ("i = %o\n", i);
   printf ("i = %x\n", i);
   printf ("i = %X\n", i);
   printf ("i = %+i\n", i);
   printf ("i = %8i\n", i);
   printf ("i = %08i\n", i);
   printf ("i = %+08i\n", i);
   printf ("f = %f\n", f);
   printf ("f = %10.3f\n", f);
   printf ("f = %+10.3f\n", f);
   printf ("f = %g\n", f);
   printf ("f = %10.6g\n", f);
   printf ("f = %10.6e\n", f);
}
```

**Αποτέλεσμα εκτέλεσης**

```
i = 123
i = 173
i = 7b
i = 7B
i = +123
i =      123
i = 00000123
i = +0000123
f = 3.141593
f =      3.142
f =     +3.142
f = 3.14159
f =    3.14159
f = 3.141593e+000
```

# Παίζοντας (ξανά) με την printf ()

```
#include <stdio.h>
main()
{
    int num1, num2, num3, num4, num5;
    num1 = 1;
    num2 = 12;
    num3 = 123;
    num4 = 1234;
    num5 = 12345;
    printf("%8d  %-8d\n", num1, num1);
    printf("%8d  %-8d\n", num2, num2);
    printf("%8d  %-8d\n", num3, num3);
    printf("%8d  %-8d\n", num4, num4);
    printf("%8d  %-8d\n", num5, num5);
}
```

**Αποτέλεσμα εκτέλεσης**

```
       1  1
      12  12
     123  123
    1234  1234
   12345  12345
```

# Ο τύπος char που είναι int

```
#include <stdio.h>
main()
{
    char ch1;

    ch1='A';
    printf("o ch1: %c\n", ch1);
    printf("o ch1: %d\n", ch1);
}
```

```
#include <stdio.h>
main()
{
    char ch1;

    ch1=65;
    printf("o ch1: %c\n", ch1);
    printf("o ch1: %d\n", ch1);
}
```

Αποτέλεσμα (και των δύο προγραμμάτων):

A

65

# Οι αριθμητικοί τελεστές

- Χρησιμοποιούνται για την εκτέλεση αριθμητικών πράξεων.
- Για κάθε τελεστή απαιτούνται 2 όροι (εκτός από τον τελεστή «**-**», που είναι δυνατόν να χρησιμοποιηθεί και σαν πρόσημο), οι οποίοι μπορεί να είναι σταθερές ή μεταβλητές.
- Τα κενά διαστήματα δεν παίζουν ρόλο και μπορείτε να χρησιμοποιήστε παρενθέσεις για μαθηματικές παραστάσεις.
- Ο τελεστής «**%**» μας δίνει το υπόλοιπο (καλείται «modulo») και εφαρμόζεται μόνο σε ακεραίους.
- Όλοι οι αριθμητικοί τελεστές είναι δυαδικοί τελεστές (δύο όροι).
- Τα σύμβολα της πρόσθεσης (+) και της αφαίρεσης (-) μπορεί να είναι και μοναδιαίοι τελεστές και εκφράζουν το πρόσημο.

| | |
|---|---|
| **+** | Πρόσθεση |
| **-** | Αφαίρεση |
| ***** | Πολαπλασιασμός |
| **/** | Διαίρεση |
| **%** | Υπόλοιπο |

```
ilikia=etos1 - etos2 ;
printf("%d", 95-55) ;
celsious=(-25);
A= 2*3+4;
b= (5-c)/d;
result = -(b+2)*a + (c +3*(a-b));
```

# Παρατηρήσεις για τους αριθμητικούς τελεστές

- Η C ακολουθεί την *αλγεβρική προτεραιότητα* στους αριθμητικούς τελεστές.

  Δηλαδή μεγαλύτερη προτεραιότητα έχουν οι παρενθέσεις, μετά οι τελεστές πρόσημου (ίδια προτεραιότητα), ακολουθούν οι *, / και % (ίδια προτεραιότητα) και τέλος οι + και – (ίδια προτεραιότητα):

  η εντολή *alpha = 2*3+4* θα δώσει στην *alpha* την τιμή *10*

- Με τις παρενθέσεις μπορούμε να αλλάξουμε τον τρόπο υπολογισμού μιας έκφρασης:

  η εντολή *alpha = 2*(3+4)* θα δώσει στην alpha την τιμή *14*

- Όταν έχουμε δυαδικούς τελεστές με την ίδια προτεραιότητα, η φορά των πράξεων είναι από αριστερά προς δεξιά:

  η εντολή *B = 12/4*3* θα δώσει την τιμή *9* στη B και όχι *1*

- Στους μοναδιαίους τελεστές πρόσημου, η φορά είναι από τα δεξιά προς αριστερά.

- Ο τελεστής εκχώρησης «=» έχει μικρότερη προτεραιότητα από τους αριθμητικούς τελεστές.

# Ο τελεστής %

Μας δίνει το υπόλοιπο μιας διαίρεσης:

- Η εντολή *alpha = 20 % 4* δίνει στην alpha την τιμή *0*
- Η εντολή *alpha = 20 % 3* δίνει στην alpha την τιμή *2*
- Η εντολή *alpha = 8 % 7* δίνει στην alpha την τιμή *1*
- Η εντολή *alpha = 8 % 8* δίνει στην alpha την τιμή *0*
- Η εντολή *alpha = 4 % 5* δίνει στην alpha την τιμή *4*

*Παράδειγμα:* Εξετάζοντας αν το περιεχόμενο της ακέραιας μεταβλητής alpha είναι άρτιος ή περιττός:

```
if (alpha % 2 = = 0)
        printf(" O arithmos einai artios\ n");
else
        printf(" O arithmos einai peritos\ n");
```

# Από τη βιβλιοθήκη math.h

```
#include <math.h>
```

| Συνάρτηση | Περιγραφή |
|---|---|
| *sin(x)* | Ημίτονο του x |
| *cos(x)* | Συνημίτονο του x |
| *tan(x)* | Εφαπτόμενη του x |
| *asin(x)* | Τέμνουσα του x ($\sin^{-1}$) με αποτέλεσμα στο [-π/2 έως π/2] με x να ανήκει στο διάστημα [-1,1] |
| *exp(x)* | Η εκθετική συνάρτηση – $e^x$ |
| *log(x)* | Ο φυσικός λογάριθμος - ln(x), x>0 |
| *log10(x)* | Ο δεκαδικός λογάριθμος - $\log_{10}(x)$, x>0 |
| *pow(x,y)* | Επιστρέφει το $x^y$ (δύναμη του x στην y) |
| *sqrt(x)* | Η τετραγωνική ρίζα του x (x>=0) |
| *ceil(x)* | Ο μικρότερος ακέραιος μεγαλύτερος του x (x:double) |
| *floor(x)* | Ο μεγαλύτερος ακέραιος, μικρότερος του x (x:double) |
| *fabs(x)* | Απόλυτη τιμή του x ( \|x\| ) |

# Ρητή μετατροπή τύπων (casting)

- Όταν κάνουμε πράξεις όπου εμπλέκονται διαφορετικού τύπου μεταβλητές ή όταν καταχωρούμε αποτελέσματα σε διαφορετικού τύπου μεταβλητές, πρέπει να είμαστε προσεκτικοί.
- Πολλές φορές κάποιος τύπος δεδομένων πρέπει να μετατραπεί προσωρινά σε κάποιο άλλο τύπο.
- Στη C είναι δυνατόν να επιβληθούν τέτοιου είδους μετατροπές.
- Αυτό γίνεται βάζοντας τον τύπο δεδομένων μέσα σε παρενθέσεις μπροστά από τη μεταβλητή (που έχει δηλωθεί διαφορετικά):

  (τύπος_δεδομένων) έκφραση ή μεταβλητή ή τιμή ή παράσταση

- Παράδειγμα:

  Αν η *alpha* έχει δηλωθεί σαν ακέραιος (*int alpha*;) τότε η εντολή

  *(float) alpha;*

  μετατρέπει προσωρινά την *alpha* από int σε *float*

- Αυτό το κάνουμε κυρίως για την αποφυγή απωλειών κλασματικών μερών.

# Παράδειγμα casting

**Λάθος αποτέλεσμα**

```
#include <stdio.h>
main()
{
    int x=20;
    int y=3;
    float z;
    z = x / y;
    printf("z=%f\n",z);
}
```

**Σωστό αποτέλεσμα**

```
#include <stdio.h>
main()
{
    int x=20;
    int y=3;
    float z;
    z = (float) x / y;
    printf("z=%f\n",z);
}
```

# Άμεση μετατροπή τύπων στις εκφράσεις

- Όταν οι μεταβλητές/σταθερές που εμπλέκονται σε μια έκφραση, είναι του ιδίου τύπου το αποτέλεσμα είναι αυτού του τύπου.
- Όμως συχνά σε μια έκφραση εμπλέκονται διαφορετικού τύπου μεταβλητές, τιμές ή σταθερές (μικτή έκφραση).
- Γενικά, ο κανόνας είναι να μετατρέπεται ο τύπος με το μικρότερο εύρος στον τύπο με το μεγαλύτερο εύρος, έτσι ώστε να μη χάνεται πληροφορία. Για παράδειγμα:

  *εάν η* alpha *είναι ακέραια μεταβλητή (integer) η* delta *είναι κινητής υποδιαστολής (float) τότε το αποτέλεσμα της έκφρασης* alpha+delta *είναι κινητής υποδιαστολής*.

- Η έκφραση υπολογίζεται στον τύπο με το μεγαλύτερο εύρος και το αποτέλεσμα είναι ίδιου τύπου.
- Εκφράσεις που αναθέτουν ένα τύπο μεγαλύτερου εύρους σε ένα μικρότερο (π.χ. i = f), συνήθως δημιουργούν μια προειδοποίηση (warning) από τον μεταφραστή (compiler) και γενικά πρέπει να αποφεύγονται.
- Γενικά ισχύει:

  *int < unsigned < long < unsigned long < float < double*

# Πίνακας μετατροπής τύπων

| Τύπος | char | int | long | float | double | long double |
|---|---|---|---|---|---|---|
| char | *char* | *int* | *long* | *float* | *double* | *long double* |
| int | *int* | *int* | *long* | *float* | *double* | *long double* |
| long | *long* | *long* | *long* | *float* | *double* | *long double* |
| float | *float* | *float* | *float* | *float* | *double* | *long double* |
| double | *double* | *double* | *double* | *double* | *double* | *long double* |
| long double | *long double* | *long double* | *long double* | *long double* | *long double* | *long double* |

# Για εξάσκηση…

- Μεταγλωττίστε και εκτελέστε τα προγράμματα που παρουσιάστηκαν.
- Ένα σώμα αφήνεται να πέσει ελεύθερα από ηρεμία. Γράψτε ένα πρόγραμμα το οποίο να υπολογίζει την απόσταση που διανύει το σώμα και την ταχύτητά του μετά από *2 sec*. Εμφανίστε ακρίβεια *2* δεκαδικών ψηφίων τα αποτελέσματα σας σας. (Δίνονται $s=1/2*(g*t^2)$, $v=g*t$, $g=9.81\ m/sec^2$).

```
#include <stdio.h>
#include <math.h>
main()
{
   double g,t,s,v;
    g=9.81;  // επιτάχυνση βαρύτητας
    t=2;       // χρόνος 2 sec
    s=(g*pow(t,2.0))/2.0;  // διάστημα
    v=g*t;          // ταχύτητα
    printf("Apostasi meta 2s:%8.2f\n",s);
    printf("Taxytita meta 2s:%8.2f\n\n",v);
    getchar();
}
```

Κάνουμε χρήση της συνάρτησης pow() που βρίσκεται στη βιβλιοθήκη math.h (#include <math.h>)

Σύνταξη της pow:

*pow (x,y)*

Υψώνει την x εις την y.

Άρα το $t^2$ συντάσσεται: pow(t,2.0)

# Περισσότερη εξάσκηση

- Λύστε τη σειρά των ασκήσεων που έχουν αναρτηθεί στο δικτυακό τόπο του μαθήματος (εργαστήριο).