Τι είναι συνάρτηση

Συνάρτηση (Function) είναι ένα μικρό κομμάτι προγράμματος (υποπρόγραμμα ή υπορουτίνα) που περιέχει μία ή περισσότερες εντολές σε γλώσσα C και εκτελεί μία ή περισσότερες δουλειές.

Σε ένα καλογραμμένο πρόγραμμα της γλώσσας C, κάθε συνάρτηση εκτελεί μία μόνο δουλειά !

Συνάρτηση main

Η συνάρτηση main( ) είναι η πρώτη συνάρτηση που εκτελείται όταν το πρόγραμμα αρχίζει να τρέχει (run) στον υπολογιστή και έχει τη μορφή :

```
main ( )
{
    Εντολές
}
```

int main ()
int main (void)
void main ()
void main (args)

Ο γενικός τύπος μιας συνάρτησης είναι:

όνομα_συνάρτησης (λίστα παραμέτρων)

δήλωση του τύπου των παραμέτρων της λίστας ;

{ εναρκτήριο άγκιστρο με το οποίο αρχίζει η συνάρτηση

.....

το σώμα της συνάρτησης ;

.....

} τελικό άγκιστρο με το οποίο τελειώνει η συνάρτηση

Παράμετροι συναρτήσεων

Παράμετρος είναι μια τιμή η οποία μεταφέρεται (περνά) μέσα στη συνάρτηση, Π.χ.

```
int x,y;
mul (x,y)
   {
     return(x*y);
   }
```

Η εντολή return αναγκάζει μια επιστροφή, δηλαδή μία και μόνο μία τιμή να επιστραφεί από τη συνάρτηση

Ένα απλό παράδειγμα

```
main()
{
  int x,y,j,k,p;
    x=10;
    y=15;
        p=mul (x,y) ;      /*  1η κλήση της συνάρτησης  */
        printf("%d",p);  /* εμφάνιση του αποτελέσματος
                            σε μορφή ακεραίου αριθμού */
        j=25;
        k=30;
        p=mul(k,j);         /*  2η κλήση της συνάρτησης  */
        printf("%d",p);   / εμφάνιση του αποτελέσματος
                            σε μορφή ακεραίου */
}
```

... συνέχεια

```
mul(x,y)
  int x,y;
 {
        return(x*y);   /* επιστρέφει το αποτέλεσμα του γινομένου */
  }
```

Ή εναλλακτικά

```
mul(int x, int y)
 {
        return(x*y);
  }
```

Παράδειγμα

```
/* Η συνάρτηση mul υπολογίζει το γινόμενο δύο αριθμών */

    mul(x, y)
    int x, y;       // Δήλωση του τύπου των παραμέτρων
    {
       return(x*y);          /* επιστρέφει το αποτέλεσμα
                                του γινομένου των δύο
                             παραμέτρων της συνάρτησης */
     }  /* Τέλος της συνάρτησης */
```

Η συνάρτηση printf ( )

Επιτρέπει την εμφάνιση στην οθόνη των αποτελεσμάτων της επεξεργασίας του προγράμματος.

```
main( )
 {
    printf("%d",123);
 }
```

Χαρακτήρες ελέγχου της συνάρτησης printf( )

Χαρακτήρας	*Αποτέλεσμα*
c	απλός χαρακτήρας
d	Ακέραιος
e	επιστημονικός συμβολισμός
f	κινητής υποδιαστολής-δεκαδικός
g	χρήση των %e ή %f (το συντομότερο)
s	σειρά χαρακτήρων
u	ακέραιος χωρίς πρόσημο
x	εμφάνιση στο δεκαεξαδικό σύστημα
o	εμφάνιση στο οκταδικό σύστημα

Παραδείγματα

printf("%s %d","this is a string", 100);

this is a string 100

printf("this is a string %d", 100);

this is a string 100

printf("number %d is decimal,%f is float.",10,110.789);

number 10 is decimal, 110.789 is float.

printf("%c %s %d-%x",'a',"number in decimal and hex:",10,10);

a number in decimal and hex: 10-A

printf("%s","HELLO ";

HELLO

Βασικά Στοιχεία της C

Οι *μεταβλητές*, οι *σταθερές* και οι *τελεστές*
Χρησιμοποιούνται για να σχηματίσουν τις
Εκφράσεις

Λέμε Έκφραση μια ΠΡΟΤΑΣΗ ή μια
ΕΝΤΟΛΗ

Οι μεταβλητές στη γλώσσα C

Αντίθετα απ' ότι ισχύει σε πολλές άλλες γλώσσες προγραμματισμού, τα ΚΕΦΑΛΑΙΑ ΓΡΑΜΜΑΤΑ διαφέρουν από τα μικρά γράμματα όταν χρησιμοποιούνται στη γλώσσα C

Η γλώσσα C είναι case sensitive, δηλαδή,

(Ευαίσθητη στους χαρακτήρες)

Τα όνομα μιας μεταβλητής μπορεί να αρχίζει με οποιοδήποτε γράμμα της λατινικής αλφαβήτου ή με ένα χαρακτήρα υπογράμμισης, το _ (underscore)

Όλες οι μεταβλητές του προγράμματος πρέπει να δηλωθούν πριν να χρησιμοποιηθούν !

Τύποι δεδομένων	*Η αντιστοιχία στη C*
χαρακτήρας	char
ακέραιος	int
βραχύς ακέραιος	short int
μακρύς ακέραιος	long int
ακέραιος χωρίς πρόσημο	unsigned int
κινητής υποδιαστολής, δεκαδικός	float
διπλής ακρίβειας κινητής υποδιαστολής	double

Παραδείγματα δηλώσεων

```
sample ( )
{
  int count;
  char c;
  float value;
  short int top;
  long int counter;
  double pay_out;
  unsigned int unit;
     .
     .      /*  Οι εντολές του προγράμματος  */
     .
}
```

Μέγεθος και εύρος των μεταβλητών

Τύπος	*Εύρος σε bit*	*Μέγεθος (τιμή)*
char	8	0 μέχρι 255
int	16 ή 32	-32768 μέχρι 32761
short int	8	-128 μέχρι 127
unsigned int	16	0 μέχρι 65535
long int	32	-4294967296 μέχρι 4294967295
float	32	ακρίβεια περίπου 6 δεκαδικών
double	64	ακρίβεια περίπου 12 δεκαδικών

Μεταβλητές

Λέμε *μεταβλητή* στον προγραμματισμό των υπολογιστών, μια θέση της μνήμης του υπολογιστή η οποία μπορεί να φιλοξενήσει μια τιμή

Προσοχή. Το όνομα μιας μεταβλητής δεν έχει καμία σχέση με τον τύπο της

Π.χ.

```
int  pragmatikos_ar_123; float akeraios15, integer;
```

- Στη γλώσσα C ένας χαρακτήρας είναι ισοδύναμος με ένα (1) byte. Δηλαδή, κάθε χαρακτήρας αποθηκεύεται σε ένα (1) byte
- Το αριστερότερο bit (bit υψηλής τάξης) ενός ακέραιου αριθμού θεωρείται σαν ο δείκτης πρόσημου (sign flag). Αν ο δείκτης του πρόσημου είναι 0 τότε ο αριθμός είναι θετικός, αν είναι 1 τότε ο αριθμός είναι αρνητικός
- Π.χ. ο αριθμός 127 στο δυαδικό σύστημα αρίθμησης (στη μνήμη), θα είναι ο αριθμός:
-

0 0000000 01111111

Παράσταση των ακεραίων αριθμών στη μνήμη

Ο ακέραιος αριθμός καταλαμβάνει 2 συνεχόμενα bytes στη μνήμη

Θέση του Bit: 15 | 14 ..... 0 |

Η γλώσσα C επιτρέπει να δηλώσουμε τους ακέραιους αριθμούς σαν αριθμούς χωρίς πρόσημο, χρησιμοποιώντας το bit του πρόσημου σαν μέρος του αριθμού και όχι σαν ένα δείκτη πρόσημου. Αυτή η τεχνική διπλασιάζει το μέγεθος του μέγιστου αριθμού

Παράσταση των πραγματικών αριθμών στη μνήμη

Λέγεται παράσταση *κινητής υποδιαστολής* ή *επιπλέουσας υποδιαστολής* (floating point)

Για την παράσταση αυτή ο πραγματικός αριθμός καταλαμβάνει τέσσερα (4) συνεχόμενα bytes στη μνήμη (32 bits).

Θέση του Bit: ***31 \| 30 .. 23 \| 22 ..... 0 \|***

Παράσταση *κινητής υποδιαστολής*

Στην παράσταση κινητής υποδιαστολής ένας πραγματικός αριθμός X εκφράζεται σε *εκθετική μορφή* (exponential representation) με τον ακόλουθο τρόπο:

$$X = s * m * b^e$$

όπου:

- s, το πρόσημο του αριθμού (s = + ή s = -)
- m, ο συντελεστής (mantissa)
- b, η βάση του συστήματος αρίθμησης (συνήθως 2 ή 8 ή 10 ή 16) και
- e, ο εκθέτης του X

Π.χ. ο αριθμός $16,1_{(10)}$ ή 10000,001

παριστάνεται ως: $0,10000001 * 2^5$

Μιγαδικοί αριθμοί

Με την έκδοση C99 έχει εισαχθεί ένας νέος τύπος πραγματικών αριθμών κινητής υποδιαστολής για την παράσταση των μιγαδικών αριθμών (complex number)

- **float_Complex,**
- **double_Complex,**
- **long double_Complex**

Δήλωση και χρήση των μεταβλητών

Υπάρχουν τρεις διαφορετικές θέσεις σ' ένα πρόγραμμα σε γλώσσα C όπου μπορούν να δηλωθούν οι μεταβλητές:

1. **μέσα στις συναρτήσεις**
2. **στον ορισμό των παραμέτρων των συναρτήσεων**
3. **έξω από όλες τις συναρτήσεις**

Τότε οι μεταβλητές λέγονται αντίστοιχα :

1. **τοπικές μεταβλητές (local variables),**
2. **τυπικές παράμετροι (formal parameters)**
3. **γενικές μεταβλητές (global variables)**

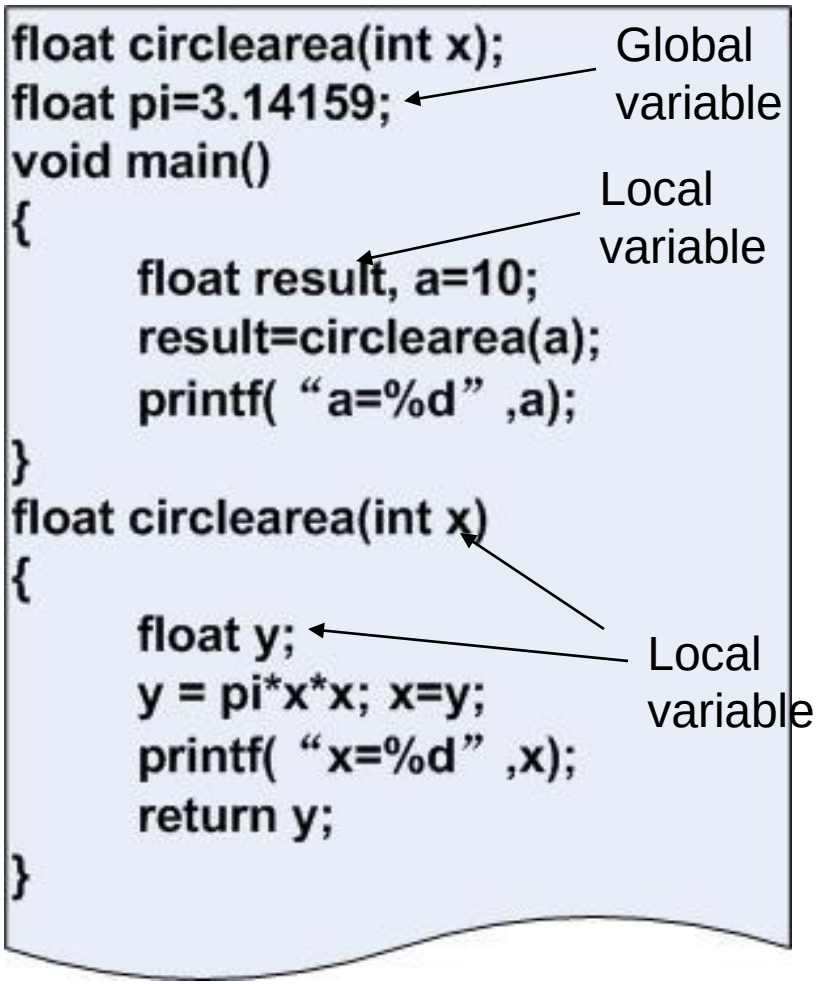
Παράδειγμα χρήσης των μεταβλητών

```
int count;    /* Η μεταβλητή count είναι γενική */
main( )
{   count=mul(10,123);
    printf("%d",count);
    printf("%d",func2());
 }

func2( )
{
  int count;    /* Η μεταβλητή
                   count είναι τοπική */
   count =10;
   return(count);
 }

mul(x,y)
int x,y;    /* ορισμός
        των παραμέτρων  */
 {
        return(x*y);
 }
```

Παράδειγμα ΙΙ



Συνήθως, οι γενικές μεταβλητές είναι χρήσιμες αλλά θα πρέπει να φροντίζουμε να χρησιμοποιούμε τις γενικές μεταβλητές με φειδώ και προσοχή και να τις αποφεύγουμε, όταν δεν είναι απαραίτητο, για τρεις κυρίως λόγους:

- (1) απασχολούν μόνιμα τη μνήμη του υπολογιστή σε όλη τη διάρκεια της εκτέλεσης του προγράμματος και όχι μόνο όταν χρειάζονται
- (2) η χρησιμοποίηση μιας γενικής μεταβλητής στη θέση μιας τοπικής μεταβλητής κάνει τη συνάρτησή εξαρτώμενη από εξωτερικές τιμές οι οποίες δεν είναι εύκολο να ελεγχθούν
- (3) η χρησιμοποίηση μεγάλου αριθμού γενικών μεταβλητών μπορεί να οδηγήσει το πρόγραμμα σε σφάλματα, εξαιτίας αγνώστων κι ανεπιθύμητων δευτερευόντων αποτελεσμάτων

Μεταβλητές καταχωρητών (register)

Η δηλωτική εντολή register εφαρμόζεται μόνο στους τύπους int και char και αναγκάζει το μεταγλωττιστή της γλώσσας C να αποθηκεύσει τις τιμές των μεταβλητών οι οποίες έχουν δηλωθεί με αυτό τον τρόπο στους ειδικούς καταχωρητές (register) της Κεντρικής Μονάδας Επεξεργασίας αντί να τοποθετήσει αυτές τις τιμές στη μνήμη RAM

Π.χ. γράφουμε:

```
register int s, t;
```

Πίνακες (Arrays)

Τύπος όνομα_μεταβλητής [αριθμός στοιχείων];

```
main ( )
{
  char string[ 80 ];

  gets (string);
  printf (string);
  printf ("%c %c %c",string[0],
                    string[1], string[2]);
}
```

Εντολή αντικατάστασης

Η γενική μορφή μιας εντολής αντικατάστασης είναι :

όνομα_μεταβλητής = έκφραση;

Η τιμή της δεξιάς πλευράς μετατρέπεται στον τύπο της αριστερής πλευράς

Σημ. Μερικοί μεταγλωττιστές της γλώσσας C (και μερικοί επεξεργαστές) μεταχειρίζονται τη μεταβλητή char σαν θετικό αριθμό, ανεξάρτητα από την τιμή που είχε όταν μετατρεπόταν σε int ή σε float.

Άλλοι μεταγλωττιστές τη στιγμή της μετατροπής θεωρούν τις τιμές των μεταβλητών char που είναι μεγαλύτερες από το 127 σαν αρνητικούς αριθμούς.

Παραδείγματα

```
price = 12.345 ;   // απλή αντικατάσταση

tab [ 0 ] = 10;      // απλή αντικατάσταση

x = (y + z) * (k +3.14);   // σύνθετη έκφραση
```

Κανόνες μετατροπής τύπου σε μία αντικατάσταση

Τύπος του στόχου	*Τύπος της έκφρασης*	*Πιθανή απώλεια πληροφορίας*
char	short int	Το πρόσημο
char	int	8 bits υψηλής τάξης
char	long int	24 bits υψηλής τάξης
short int	int	8 bits υψηλής τάξης
short int	long int	24 bits υψηλής τάξης
int	long int	16 bits υψηλής τάξης
int	float	Το δεκαδικό μέρος
float	double	Η ακρίβεια, το αποτέλεσμα στρογγυλοποιείται

Παραδείγματα

```
short  int x;
char ch;
 x= 65;
 ch=x;

printf("x=%d  ch(ακέραιος)=%d
          ch(χαρακτήρας)=%c\n",  x, ch, ch);
```

x=65 ch(ακέραιος)=65 ch(χαρακτήρας)=A

Παραδείγματα

```
int x;
   float  f;
     f=3.14;
          x=f;

          printf("x=%d  f=%f \n", x, f);
```

x=3 f=3.140000

Σταθερές (Constants)

Σταθερές στη γλώσσα C είναι οι σταθερές τιμές που δεν μπορεί ένα πρόγραμμα να τις αλλάξει.

Η γλώσσα C υποστηρίζει κι ακόμα ένα τύπο.

Αυτός είναι η αλφαριθμητική σταθερά.

Όλες οι αλφαριθμητικές σταθερές περικλείονται σε διπλά εισαγωγικά, π.χ. "this is a test"

Η C υποστηρίζει και την εντολή #define η οποία χρησιμοποιείται για να ορίσει μία σειρά χαρακτήρων σαν μια σταθερή

π.χ. #define MAX_NUM 100

ή ακόμη και

#define err "Syntax error"

Σταθερές (Constants)

Τύπος	*Παραδείγματα*
char	'a' ' \n' '9'
int	11 232 1000 -234
long int	85000 -345
short int	10 -12 90
Unsigned int	10000 987
float	123.45 356.61
double	123.44 0.4567893

Αρχικές τιμές μεταβλητών

Μπορούμε να δώσουμε στις περισσότερες μεταβλητές της γλώσσας C μια τιμή τη στιγμή που τις δηλώνουμε με την προσθήκη του σήματος της ισότητας και μία σταθερή τιμή μετά το όνομα της μεταβλητής.

Ο γενικός τύπος του προσδιορισμού της αρχικής τιμής είναι:

Τύπος όνομα_μεταβλητής = σταθερή_τιμή ;

Παραδείγματα:

```
char ch = 'a';
int first = 15;
float balance = 123.23;
```

Αλφαριθμητική μεταβλητή

Λέμε Αλφαριθμητική μεταβλητή (string) μία σειρά χαρακτήρων που τελειώνει μ' ένα κενό χαρακτήρα

Ο Κενός χαρακτήρας (null) στη γλώσσα C συνήθως είναι το 0 (μηδέν)

Για να δώσουμε αρχική τιμή στον πίνακα str με την έκφραση "Hi there" μπορούμε να γράψουμε :

```
char str[9] = "Hi there" ;
```

Οι πίνακες επίσης μπορούν να πάρουν αρχικές τιμές με την απαρίθμηση των στοιχείων τους και να βρίσκονται μέσα σε άγκιστρα. Π.χ. το "Hi there", μ' αυτή τη μέθοδο γράφεται:

```
char str[9] = {'H', 'i', ' ' , 't', 'h', 'e', 'r', 'e', '\0'};
```

Ειδικές σταθερές χαρακτήρων (backslash)

- Υπάρχουν ορισμένοι χαρακτήρες, όπως ο χαρακτήρας **επιστροφής** (carriage return ή Enter) οι οποίοι είναι αδύνατον να δοθούν από το πληκτρολόγιο.
- Για το λόγο αυτό, η γλώσσα C έχει δημιουργήσει τις ειδικές σταθερές χαρακτήρων στις οποίες αποδίδεται ο όρος **backslash.**

\n Αλλαγή γραμμής (*new line*)

\0 Μηδέν (Τερματικός χαρακτήρας)

Τελεστές (operators)

Τελεστής	*Πράξη*
-	αφαίρεση
+	πρόσθεση
*	πολλαπλασιασμός
/	διαίρεση
%	διαίρεση modulo
--	μείωση
+ +	αύξηση

Ένα παράδειγμα χρήσης του τελεστή %

```
main()
{
 int x, y ;
  x=10;
  y=3;
  printf("%d",x/y);      /* θα εμφανιστεί ο αριθμός 3 */
  printf("%d",x%y);   /* θα εμφανιστεί ο αριθμός 1 */
  x=1;
  y=2;
  printf("%d  %d", x/y, x%y);  /* θα εμφανιστούν οι αριθμοί  0 και 1
*/
}
```

Άσκηση

Να γραφτεί ένα πρόγραμμα για τον υπολογισμό των εκφράσεων:

$$X = 15 * 2 + (255 – 123) * (321 / 7 – 12) + 1234 / (7 – 4)$$

$$ZZ = (14 - 2)^2 / (13 + 5)^3 – 12$$

$$AAA = X - ZZ*2 + 125/6$$

Οι τελεστές ++ αυξητικός και -- μειωτικός

Ο τελεστής ++ προσθέτει τη μονάδα στο τελεστέο του ενώ ο τελεστής -- αφαιρεί τη μονάδα.

Οι παρακάτω πράξεις είναι ισοδύναμες:

x = x + 1; είναι το ίδιο με ++x;

x = x - 1; είναι το ίδιο με --x;

Τόσο ο αυξητικός όσο και ο μειωτικός τελεστής μπορούν ή να προηγηθούν του τελεστέου ή να τον ακολουθήσουν

Όταν ο αυξητικός ή ο μειωτικός τελεστής προηγείται του τελεστέου του, η C θα εκτελέσει την πράξη πριν να χρησιμοποιήσει την τιμή του τελεστέου.

Αν ο τελεστής ακολουθεί τον τελεστέο του, η C θα χρησιμοποιήσει την τιμή του τελεστέου πριν την αύξηση ή τη μείωση.

Παραδείγματα

```
int x, y;
x = 10;
y = ++x;
printf("%d %d", x, y );
```

Τα x, y θα είναι ίσα με 11

Αν όμως είχε γραφεί σαν :

```
int x, y;
x = 10;
y = x++;
printf("%d %d", x, y );
```

Σ' αυτή την περίπτωση το y θα είναι ίσο με 10 και το x με 11

Και στις δυο περιπτώσεις το x γίνεται 11

Άσκηση

Τι θα εμφανίσει το πρόγραμμα:

```
int x, y;
x = 105;
y = ++x + 10;
printf("X = %d ",  y );
```

Και τι το πρόγραμμα:

```
float x, y;
x = - 14;
y = x++ – 2;
printf("X = %f ",  y );
```

Σχετικοί και Λογικοί Τελεστές

Σχετικός Τελεστής	*Πράξη*
>	μεγαλύτερο
>=	μεγαλύτερο ή ίσο
<	μικρότερο
<=	μικρότερο ή ίσο
= =	ίσο
!=	άνισο

Σχετικοί και Λογικοί Τελεστές

Λογικός Τελεστής	*Πράξη*
&&	AND (Τομή)
||	OR (Ένωση)
!	NOT (Άντίθετο)

Όλες οι σχετικές και λογικές εκφράσεις παράγουν αποτέλεσμα 0 ή 1

Μηδέν (0) σημαίνει ψευδές

Κάθε τιμή διαφορετική του μηδενός σημαίνει ΑΛΗΘΕΣ

Έτσι, το απόσπασμα του προγράμματος που ακολουθεί δεν είναι απλώς σωστό, αλλά θα εμφανίσει και τον αριθμό 1 στην οθόνη:

```
int x;
x=100;
printf ("%d",    x >10);
```

Τελεστές Bitwise

Ο όρος πράξεις bitwise αναφέρεται στον έλεγχο ή την μετακίνηση των bits σε ακέραιους ή μεταβλητές χαρακτήρων

Αυτές οι πράξεις δεν χρησιμοποιούνται στους πραγματικούς float και double

Οι πράξεις bitwise βρίσκουν συχνά εφαρμογή στους οδηγούς (drivers) μηχανισμών και συσκευών, καθώς επίσης και προγραμμάτων ελέγχου modem και εκτυπωτών, επειδή μπορούν να κάνουν πράξεις μεταξύ bits π.χ. για να κρύψουν ορισμένα bits όπως το bit ισοτιμίας

Τελεστές Bitwise

Τελεστής	*Πράξη*
&	AND
|	OR
^	αποκλειστικό OR
~	συμπλήρωμα του 1
>>	μετακίνηση δεξιά
<<	μετακίνηση αριστερά

Παράδειγμα Τελεστή Bitwise

char x;	*Κατάσταση του x*	*Τιμές του x*
x=7	00000111	7
x << 1;	00001110	14
x<<3;	01110000	112
x<<2;	11000000	192
x>>1;	01100000	96
x>>2;	00011000	24

πράξεις bitwise

Θεωρούμε ότι η μεταβλητή ch έχει τιμή το χαρακτήρα 'A' και έχει bit ισοτιμίας στο 1

1 1 0 0 0 0 0 1 Το bit ισοτιμίας της ch στο 1 (χαρακτήρας A)

&

0 1 1 1 1 1 1 1 Ο αριθμός 127 στο δυαδικό σύστημα αρίθμησης

---

0 1 0 0 0 0 0 1 Το 'A' χωρίς ισοτιμία

Το bitwise **OR** χρησιμοποιείται για να ανάψει τα bits. Κάθε bit που είναι μονάδα σ' οποιονδήποτε από τους δύο τελεστέους θα προκαλέσει ώστε το αντίστοιχο bit της μεταβλητής να γίνει μονάδα.

Για παράδειγμα, το **128 | 3** θα δώσει

```
  1 0 0 0 0 0 0 0      128 σε δυαδικό
|
  0 0 0 0 0 0 1 1        3 σε δυαδικό
 -------------------
  1 0 0 0 0 0 1 1      αποτέλεσμα
```

Ο Τελεστής ?

Μπορεί να χρησιμοποιηθεί για ν' αντικαταστήσει εντολές του τύπου

if then else

Το τρίτιμο ζεύγος τελεστών ? έχει τη γενική μορφή :

Exp1 ? Exp2 : Exp3

Ο τελεστής ? λειτουργεί ως εξής:

Η Exp1 υπολογίζεται. Αν είναι αληθής, η Exp2 υπολογίζεται και γίνεται τιμή της έκφρασης.

Αν η Exp1 είναι ψευδής η Exp3 υπολογίζεται και γίνεται η τιμή της έκφρασης. Π.χ.

```
x=10;
y = x>9 ? 100 : 200;
```

Εδώ το y παίρνει την τιμή 100.

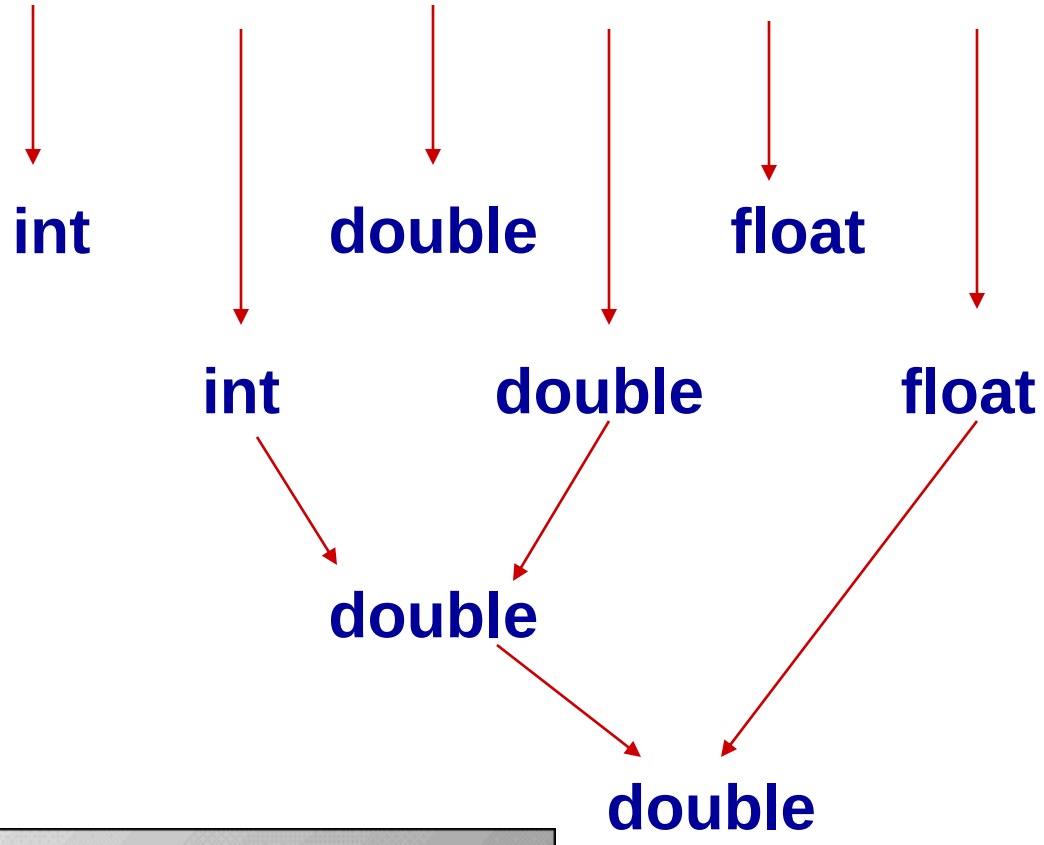
Αν το x ήταν μικρότερο από το 9, τότε το y θα έπαιρνε την τιμή 200

Προτεραιότητα των τελεστών της γλώσσας C

Υψηλότερη	( ) [ ]
	! ~ ++ -- - (type) * & sizeof
	*/%
	+ -
	<< >>
	> >= < <=
	= = ! =
	&
	^
	|
	&&
	||
	? :
	= += - = * = / =
Χαμηλότερη	,

```
char ch;
int i;
float f;
double d;
```

```
result = ( ch / i ) + ( f * d ) - ( i + f );
```



Τυποποιήσεις (casts)

Είναι δυνατόν να αναγκάσουμε μία έκφραση να γίνει κάποιου συγκεκριμένου τύπου με την τυποποίηση.

Ο γενικός τύπος της τυποποίησης είναι:

(τύπος) έκφραση

όπου τύπος είναι ένας από τους σταθερούς τύπους δεδομένων της γλώσσας C και έκφραση μια οποιαδήποτε έκφραση αποδεκτή από τη C

Αν θέλουμε να βεβαιωθούμε ότι η έκφραση x/2 υπολογίσθηκε σωστά και το αποτέλεσμα θα είναι τύπου float, πρέπει να τη γράψουμε ως εξής :

(float) x/2

Στενογραφία στη C

Η γλώσσα C έχει μια ιδιαίτερη στενογραφία, η οποία δεν συναντάται σε άλλες γλώσσες προγραμματισμού και η οποία απλοποιεί την κωδικοποίηση ενός συγκεκριμένου τύπου εντολών αντικατάστασης.

Ο γενικός τύπος αντικατάστασης συνήθως είναι :

μεταβλητή = μεταβλητή τελεστής έκφραση

Αυτό είναι το ίδιο με τον τύπο :

μεταβλητή τελεστής = έκφραση

Π.χ. η απλή εντολή : x = x + 10;

μπορεί να γραφτεί στη στενογραφία της γλώσσας C

x + = 10;

Οι τελεστές δεικτών & και *

- Λέμε *δείκτη* (pointer) τη διεύθυνση στη μνήμη μιας μεταβλητής.

Οι δείκτες έχουν διπλή λειτουργία στη γλώσσα C

- είναι ένας πολύ σύντομος τρόπος αναφοράς στα στοιχεία πινάκων και

- επιτρέπουν στις συναρτήσεις της γλώσσας C να τροποποιήσουν τις παραμέτρους κλήσης τους

- Ο πρώτος τελεστής είναι ο χαρακτήρας &.

Είναι ένας μοναδιαίος τελεστής ο οποίος επιστρέφει τη διεύθυνση μνήμης του τελεστέου του.

Αν γράψουμε:

```
int count ;
count = 562 ;
mem = &count;
```

τότε στη μεταβλητή mem αποθηκεύεται η διεύθυνση μνήμης της μεταβλητής count και όχι η τιμή 562.

- Αυτή η διεύθυνση είναι η εσωτερική θέση της μεταβλητής στη μνήμη RAM.
- Η διεύθυνση μνήμης της μεταβλητής count (δείκτης) δεν έχει καμιά σχέση με την τιμή της μεταβλητής count

- Ο δεύτερος τελεστής είναι ο χαρακτήρας *.

Είναι ένας μοναδιαίος τελεστής ο οποίος επιστρέφει την τιμή της μεταβλητής η οποία βρίσκεται στη διεύθυνση την οποία προσδιορίζει ο δείκτης.

Έτσι, μετά την εκτέλεση της εντολής:

```
next = *mem ;
```

μεταφέρεται στη μεταβλητή next η τιμή της μεταβλητής η οποία βρίσκεται στη διεύθυνση μνήμης την οποία προσδιορίζει ο δείκτης mem

Ο τύπος void

Η γλώσσα C, για να προσαρμοστεί στις ανάγκες του δομημένου προγραμματισμού έχει δημιουργήσει και ένα επιπλέον προσδιοριστή τύπου ο οποίος απλά δείχνει ότι δεν θα είναι διαθέσιμη καμιά τιμή.

Ο τύπος void χρησιμοποιείται σε τρεις περιπτώσεις:

- Στις εκφράσεις, οι οποίες δεν θα επιστρέφουν τιμή
- Στις συναρτήσεις, οι οποίες δεν θα έχουν παραμέτρους
- Στους δείκτες, οι οποίοι θα δείχνουν σε διαφορετικούς τύπους

Π.χ. μπορούμε να γράψουμε: `void list ( ) ;`

Συναρτήσεις εισόδου/εξόδου

Οι απλούστερες από τις συναρτήσεις εισόδου/εξόδου της γλώσσας C, είναι:

- **getchar( ),** η οποία διαβάζει ένα χαρακτήρα από την είσοδο (συνήθως το πληκτρολόγιο), και

- **putchar( ),** η οποία τυπώνει/εμφανίζει ένα χαρακτήρα στην έξοδο (συνήθως στην οθόνη).

Π.χ.
```
char ch;
ch=getchar( );      // Είσοδος
putchar(ch);        // Έξοδος
```

Οι συναρτήσεις gets( ) και puts( )

- Οι συναρτήσεις gets( ) και puts( ) επιτρέπουν να διαβάσουμε και να γράψουμε όχι μόνο απλούς χαρακτήρες αλλά σειρές χαρακτήρων.

- Η συνάρτηση gets( ) επιτρέπει να διορθώσουμε τα πιθανά λάθη κατά την πληκτρολόγηση, χρησιμοποιώντας το πλήκτρο BACKSPACE πριν πατηθεί το πλήκτρο ENTER.

- Η συνάρτηση puts( ) εμφανίζει την παράμετρό της δηλαδή, μια σειρά χαρακτήρων στην οθόνη.

Προδιαγραφές της συνάρτησης printf( )

Η γενική μορφή των προδιαγραφών οι οποίες χρησιμοποιούνται μέσα σε μια συνάρτηση printf( ) είναι:

% [δείκτης] [εύρος] [. Πλήθος σημαντικών ψηφίων] χαρακτήρας ελέγχου

- Ο δείκτης μπορεί να είναι ένας συγκεκριμένος χαρακτήρας (+ , κενό διάστημα , - , 0) με τις ακόλουθες ενέργειες:

+ , τοποθετεί το θετικό πρόσημο μπροστά από ένα θετικό αριθμό

κενό διάστημα (blanc), τοποθετεί το κενό διάστημα μπροστά από ένα θετικό αριθμό

- , τοποθετεί τον αριθμό στοιχισμένο αριστερά στο πεδίο που ορίζεται από το εύρος

0 , συμπληρώνει με μηδενικά, όσα χρειάζονται και πριν από τον αριθμό, στο πεδίο το οποίο καθορίζεται από το εύρος

Παραδείγματα εξόδου

Εντολή printf( )	*έξοδος*
("%-5.2f",123.234)	123.23
("%5.2f",3.238)	3.24
("%l0s" "Σήμερα")	Σήμερα
("%-l0s"," Σήμερα ")	Σήμερα
("%5.7s","123456789")	1234567

Η συνάρτηση scanf( )

Η συνάρτηση scanf( ) επιτρέπει να διαβάσουμε δεδομένα και αυτόματα μετατρέπει αριθμητικές πληροφορίες σε ακεραίους ή σε πραγματικούς.

Ο γενικός τύπος της συνάρτησης scanf( ) είναι:

```
scanf ("σειρά χαρακτήρων ελέγχου", ονόματα μεταβλητών);
```

Π.χ.

```
# include <stdio.h>
   main ( )
    {     int number;
        scanf("%d", &number );    // Ανάγνωση  ακέραιου
//    Υπολογισμός και εμφάνιση του αποτελέσματος
 printf("\n %d", number * number * number * number );
    }
```

Κωδικοποίηση χαρακτήρων

- Οι υπολογιστές αποθηκεύουν τους χαρακτήρες αντιστοιχώντας στο καθένα τους από ένα διαφορετικό και μοναδικό αριθμό.
- Αυτή η αντιστοιχία ονομάζεται ***κωδικοσελίδα***.
- Λόγω του περιορισμού του μεγέθους των κωδικοσελίδων σε καμία κωδικοσελίδα δεν είναι δυνατόν να υπάρχει αντιστοιχία όλων των γνωστών χαρακτήρων των υπαρχουσών γραπτών γλωσσών
- Οι πιο βασικές κωδικοσελίδες οι οποίες χρησιμοποιούνται από όλους τους υπολογιστές στην Ελλάδα είναι:

DOS 437/737 για το λειτουργικό σύστημα MS-DOS

Windows 1253 για όλες τις εκδόσεις των Windows

ISO 8859-7 γνωστή και ως ΕΛΟΤ 928

- Και οι τρεις κωδικοσελίδες για τις 128 πρώτες θέσεις (7 bits) χρησιμοποιούν τον κώδικα ASCIII γι αυτό και συμφωνούν απολύτως.
- Τις επόμενες 128 θέσεις δεν τις χρησιμοποιούν με τον ίδιο τρόπο. Για το λόγο αυτό το ίδιο πρόγραμμα όταν περιέχει ελληνικά γράμματα στις εντολές εξόδου, εμφανίζει με διαφορετικό τρόπο τα αποτελέσματα, ανάλογα με το περιβάλλον εκτέλεσης του προγράμματος.
- Μια εναλλακτική πρόταση για την επίλυση αυτού του προβλήματος αποτελεί η χρησιμοποίηση των greeklish δηλαδή, η χρήση λατινικών γραμμάτων αντί για τα ελληνικά γράμματα.
- Με τον τρόπο αυτό μπορούμε να χρησιμοποιήσουμε στις εντολές εξόδου τα greeklish έτσι ώστε ανεξάρτητα του περιβάλλοντος εργασίας τα μηνύματα προς το χρήστη καθώς και όλες οι εμφανιζόμενες πληροφορίες οι οποίες θα περιέχουν ελληνικά γράμματα να είναι εν μέρει κατανοητές.

- Π.χ. για την πρόταση:
- Αποτελέσματα του προγράμματος
-
- μπορούμε να γράψουμε :
- Apotelesmata tou programmatos

- Τα τελευταία χρόνια το Unicode, αλλάζει αυτή την κατάσταση γιατί κάνει χρήση 16 bits για την αναπαράσταση κάθε χαρακτήρα

  $2^{16} = 65536$ χαρακτήρες
- Το Unicode υποστηρίζεται από πολλά λειτουργικά συστήματα και όλους τους σύγχρονους φυλλομετρητές του Διαδικτύου (browsers) αλλά δυστυχώς, δεν υποστηρίζεται ακόμη αυτόματα από όλα τα περιβάλλοντα ανάπτυξης και εκτέλεσης προγραμμάτων της γλώσσας C.

- Σε όλες τις εκδόσεις του λειτουργικού συστήματος των Windows της Microsoft εφαρμόζεται η κωδικοσελίδα Windows 1253,
- Στο προηγούμενο λειτουργικό σύστημα της Microsoft το MS-DOS, χρησιμοποιείται η κωδικοσελίδα 437/737.
- Έτσι, ότι παράγεται ή εκτελείται στο γραφικό περιβάλλον των Windows ακολουθεί τη κωδικοσελίδα Windows 1253, ενώ ότι εκτελείται στο μη γραφικό περιβάλλον του MS-DOS (μαύρη οθόνη), ακολουθεί τη κωδικοσελίδα 437/737.
- Για την αλλαγή (CHange Code Page) της κωδικοσελίδας εμφάνισης των αποτελεσμάτων σε Windows 1253 πρέπει να προσθέσουμε στην αρχή του πηγαίου κώδικα την εντολή:
- system("chcp 1253");

Βελτιωμένη έκδοση

```
/*  Βελτιωμένη έκδοση εισόδου/εξόδου τιμών */
   # include <stdio.h>
      main ( )
       {
         int number;
            system("chcp 1253");
printf("Πρόγραμμα υπολογισμού της τέταρτης δύναμης ενός αριθμού\n");
        printf("Πληκτρολογήστε τον αριθμό : ");
   scanf("%d", &number );     // Ανάγνωση του ακέραιου αριθμού
           printf("\n Η τέταρτη δύναμη του %d είναι : %d \n",
                 number, number * number * number * number );
              system("PAUSE");
       }
```

Παραδείγματα

- Να γράψετε μια συνάρτηση με όνομα **add( )** η οποία να έχει δύο ακέραιες παραμέτρους.
- Η συνάρτηση add( ) να επιστρέφει την τιμή του αθροίσματος των δύο παραμέτρων της.
- Να γραφτεί και η συνάρτηση **main( )** η οποία θα καλεί τη συνάρτηση **add( )** και με τις κατάλληλες εντολές εισόδου/εξόδου θα επιτρέπει την ασφαλή εισαγωγή των τιμών και την επεξηγηματική εμφάνιση του αποτελέσματος.

```
# include <stdio.h>
  int add(x,y)
 int x, y;
 {
  return(x+y);
 }

    main( )
  {   system("chcp 1253");
      int t1,t2;
printf("Πρόγραμμα υπολογισμού του αθροίσματος 2 αριθμών \n");
      printf("Πληκτρολογήστε τον πρώτο αριθμό : ");
        scanf("%d", &t1 );     // Ανάγνωση του πρώτου αριθμού
      printf("Πληκτρολογήστε το δεύτερο αριθμό : ");
        scanf("%d", &t2 );     // Ανάγνωση του δεύτερου αριθμού
   printf("Το άθροισμα των 2 αριθμών είναι : %d \n", add(t1,t2) );
   }
```

Δοκιμάσετε να εφαρμόσετε τους τρεις τελεστές bitwise:

& | και ^

Α. Στις τιμές : x1='A' και x2='B'

Β. Στις τιμές : y1= 12 και y2= 7

Γ. Στις τιμές : z1= 67 και z2= 'C'

και να εμφανίσετε στην οθόνη τα αποτελέσματα.

```
# include <stdio.h>

    main( )
    {
        int t1=67;
        char t2='C’;
        printf("Το αποτέλεσμα είναι : %d \n", (t1 & t2) );
    }
```