Εισαγωγή στις Συναρτήσεις

Η φιλοσοφία σχεδίασης της C βασίζεται στη χρήση των συναρτήσεων. Έχουμε ήδη δει και χρησιμοποιήσει πολλές συναρτήσεις που έχει το σύστημα, όπως είναι οι printf(), scanf(),αλλά έχουμε δημιουργήσει και δικές μας συναρτήσεις. Οι πιο πολλές είχαν το όνομα main(). Τα προγράμματα της C πάντα αρχίζουν με την εκτέλεση των εντολών της συνάρτησης main() και μετά η main() καλεί άλλες συναρτήσεις.

Συνάρτηση αποκαλείται μια αυτοδύναμη μονάδα κώδικα προγράμματος που έχει σχεδιασθεί για να εκτελεί μια συγκεκριμένη εργασία. Μια συνάρτηση της C κάνει την ίδια δουλειά με τις συναρτήσεις, τις υπορουτίνες και τις διαδικασίες των άλλων γλωσσών. Μερικές συναρτήσεις, όπως η printf(), έχουν σαν αποτέλεσμα να γίνει μια ενέργεια. Μερικές άλλες συναρτήσεις υπολογίζουν μια τιμή, που χρησιμοποιείται από το πρόγραμμα, όπως η strlen(). Γενικά, μια συνάρτηση μπορεί να παράγει ενέργειες ή και να παράγει τιμές.

Χρησιμοποιούμε τις συναρτήσεις για να γλυτώσουμε από τον επαναλαμβανόμενο προγραμματισμό. Δηλαδή, αν πρέπει να γίνει μια συγκεκριμένη εργασία πολλές φορές μέσα σ' ένα πρόγραμμα, τότε γράφουμε μια φορά μια κατάλληλη συνάρτηση και μετά την χρησιμοποιούμε από το πρόγραμμα όταν και όπου την χρειαζόμαστε.

Μπορούμε να χρησιμοποιήσουμε την ίδια συνάρτηση σε διαφορετικά προγράμματα, αλλά ακόμα και αν κάνουμε μια εργασία μόνο μία φορά μέσα στο πρόγραμμα, θα είναι πιο σωστό να χρησιμοποιήσουμε μια συνάρτηση, επειδή έτσι το πρόγραμμα γίνεται πιο εύκολο στην ανάγνωση και στην τροποποίησή του και είμαστε πιο κοντά σ' αυτό που λέγεται *δομημένος προγραμματισμός*. Για παράδειγμα, αν θέλουμε να κάνουμε ένα πρόγραμμα που να διαβάζει μια λίστα αριθμών, να τους ταξινομεί, να βρίσκει τον μέσο όρο τους και να τυπώνει ένα ραβδόγραμμα, θα μπορούσαμε να δημιουργήσουμε το εξής πρόγραμμα :

```
#include <stdio.h>

#define SIZE 50

main()

{

        float list[SIZE];

        readlist(list, SIZE);

        sort(list, SIZE);

        average(list, SIZE);

        bargraph(list, SIZE);

}
```

Όταν χρησιμοποιούμε περιγραφικά ονόματα για τις συναρτήσεις, είναι ευκολότερο να καταλάβουμε τι κάνει ένα πρόγραμμα και πώς αυτό οργανώνεται. Μπορούμε ακόμα να επεξεργαστούμε ξέχωρα την κάθε συνάρτηση μέχρι να πετύχουμε την σωστή εκτέλεση μιας εργασίας. Ένα ακόμη όφελος είναι ότι αν δημιουργήσουμε τις συναρτήσεις κατά γενικό τρόπο, θα μπορούμε τότε να τις χρησιμοποιήσουμε και σ' άλλα προγράμματα.

Οι συναρτήσεις έχουν σαν σκοπό να μας βοηθήσουν να επικεντρώνουμε την προσοχή μας στην συνολική σχεδίαση ενός προγράμματος και όχι στις λεπτομέρειές του. Πριν να γράψουμε τον κώδικα μιας συνάρτησης, πρέπει να σκεφτούμε πρώτα τι δουλειά πρέπει να κάνει αυτή η συνάρτηση και πώς θα συσχετίζεται με τ' όλο πρόγραμμα.

Δημιουργία και Χρήση μιας Απλής Συνάρτησης

Στο πρόγραμμα αυτό διαβάζονται οι πλευρές ενός ορθογωνίου στο main, στη συνέχεια μεταβιβάζονται σε μια συνάρτηση η οποία υπολογίζει και ΕΠΙΣΤΡΕΦΕΙ το εμβαδόν του. Μετά μεταβιβάζονται σε μια άλλη συνάρτηση η οποία υπολογίζει και ΕΠΙΣΤΡΕΦΕΙ την περίμετρο του.

Α ΤΡΟΠΟΣ ΚΛΗΣΗΣ ΣΥΝΑΡΤΗΣΕΩΝ (ΜΕ ΕΝΤΟΛΗ ΚΑΤΑΧΩΡΙΣΗΣ)

```
#include <stdio.h>

int embadon(int a,int b)
{
int e;

e=a*b;

return e; //ή ισοδύναμα κατευθείαν return a*b χωρίς τη δήλωση του e;
}

int perimetros(int a,int b)
{
int p;

p=2*(a+b);

return p; //ή ισοδύναμα κατευθείαν return 2*(a+b) χωρίς τη δήλωση του p;
}

void main()
{
int x,y,emb,per;

printf("Dose tis plevres orthogoniou\n");
scanf("%d%d",&x,&y);

emb=embadon(x,y);
per=perimetros(x,y);
printf("Embadon orthogoniou = %d\n",emb);
printf("Perimetros orthogoniou = %d\n",per);
}
```

Στο πρόγραμμα αυτό διαβάζονται οι πλευρές ενός ορθογωνίου στο main, στη συνέχεια μεταβιβάζονται σε μια συνάρτηση η οποία υπολογίζει και ΕΠΙΣΤΡΕΦΕΙ το εμβαδόν του. Μετά μεταβιβάζονται σε μια άλλη συνάρτηση η οποία υπολογίζει και ΕΠΙΣΤΡΕΦΕΙ την περίμετρο του

Β ΤΡΟΠΟΣ ΚΛΗΣΗΣ ΣΥΝΑΡΤΗΣΕΩΝ (ΜΕΣΑ ΣΕ ΕΝΤΟΛΗ PRINTF)

```
#include <stdio.h>

int embadon(int a,int b)
{
int e;
e=a*b;
return e; //ή ισοδύναμα κατευθείαν return a*b χωρίς τη δήλωση του e;
}

int perimetros(int a,int b)
{
int p;
p=2*(a+b);
return p; //ή ισοδύναμα κατευθείαν return 2*(a+b) χωρίς τη δήλωση του p;
}

void main()
{
int x,y;

printf("Dose tis plevres orthogoniou\n");
scanf("%d%d",&x,&y);

printf("Embadon orthogoniou = %d\n",embadon(x,y));
printf("Perimetros orthogoniou = %d\n",perimetros(x,y));
}
```

Στο πρόγραμμα αυτό διαβάζονται οι πλευρές ενός ορθογωνίου στο main, στη συνέχεια μεταβιβάζονται σε μια συνάρτηση η οποία υπολογίζει και εμφανίζει το εμβαδόν του. Μετά μεταβιβάζονται σε μια άλλη συνάρτηση η οποία υπολογίζει και εμφανίζει την περίμετρο του

Γ ΤΡΟΠΟΣ ΚΛΗΣΗΣ ΤΩΝ ΣΥΝΑΡΤΗΣΕΩΝ : ΑΠΛΑ ΜΕ ΤΟ ΟΝΟΜΑ ΤΟΥΣ ΚΑΙ ΜΕ ΤΙΣ ΤΙΜΕΣ ΠΟΥ ΤΟΥΣ ΜΕΤΑΒΙΒΑΖΟΥΜΕ ΓΙΑ ΝΑ ΚΑΝΟΥΝ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΤΟΥΣ. Η ΚΑΘΕ ΣΥΝΑΡΤΗΣΗ ΥΠΟΛΟΓΙΖΕΙ ΚΑΙ ΕΜΦΑΝΙΖΕΙ ΤΟ ΑΠΟΤΕΛΕΣΜΑ ΤΗΣ ΜΕ ΤΗΝ ΕΝΤΟΛΗ PRINTF

```
#include <stdio.h>

void embadon(int a,int b)
{
        printf("Embadon orthogoniou = %d\n",a*b);

        /*Ισοδύναμα θα μπορούσαμε να γράψουμε:

        int e;
```

```
	e=a*b;
	printf("Embadon orthogoniou = %d\n",e);

	*/
}

void perimetros(int a,int b)
{
	printf("Perimetros orthogoniou = %d\n",2*(a+b));

	/*Ισοδύναμα θα μπορούσαμε να γράψουμε:

	int p;

	p=2*(a+b);
	printf("Perimetros orthogoniou = %d\n",p);

	*/
}

void main()
{
	int x,y;

	printf("Dose tis plevres orthogoniou\n");
	scanf("%d%d",&x,&y);

	embadon(x,y);
	perimetros(x,y);
}
```

ΑΛΛΟ ΠΑΡΑΔΕΙΓΜΑ ΜΕ ΧΡΗΣΗ ΣΥΝΑΡΤΗΣΕΩΝ

Να διαβάζει από το πληκτρολόγιο 10 ακεραίους αριθμούς και να τους αποθηκεύει σε ένα πίνακα (επιτρεπόμενες τιμές μεταξύ 100-500).
Στη συνέχεια ο χρήστης θα δίνει έναν αριθμό και το πρόγραμμα θα του εμφανίζει μήνυμα αν υπάρχει ή όχι στον πίνακα.
Η εύρεση του αριθμού να γίνεται με χρήση συνάρτησης όπου η επιστρεφόμενη τιμή να είναι 0 (δεν βρέθηκε) ή 1 (βρέθηκε).
Στη συνέχεια το πρόγραμμα να υπολογίζει και να εκτυπώνει (χωρίς συνάρτηση αυτή τη φορά) πόσες φορές υπάρχει ο συγκεκριμένος αριθμός μέσα στον πίνακα.

```
#include <stdio.h>
#include <stdlib.h>


#define SIZE 10

int search(int [], int, int); //Anazitisi arithmou kai epistrofi 1 h 0
```

```
main()
{
 int a[SIZE],x,result,i;
 for (i=0;i<SIZE;i++)
   do {
     printf("Dwse %do arithmo: ",i+1);
     scanf("%d",&a[i]);
   } while (a[i]<100 || a[i]>500);

 printf("Dwse arithmo pros anazitisi: \n");
 scanf("%d",&x);
 result=search(a,SIZE,x);
 if (result == 0)
   printf("O arithmos %d den vrethike\n\n",x);
 else
   {
      int count=0;
      for (i=0;i<SIZE;i++)
        if (a[i]==x) count++;
      printf("O arithmos %d yparxei %d fores.",x,count);
   }
 system("PAUSE");
}

int search(int a[], int size, int x)
{
 int i;
 for (i=0; i<size; i++)
   if (a[i]==x) return 1;
 return 0;
}
```

Η Χρήση του return

Ας δούμε τώρα και την περίπτωση που μια συνάρτηση επιστρέφει μια τιμή στο πρόγραμμα που την κάλεσε. Η επόμενη συνάρτηση imin() επιστρέφει σαν τιμή το μικρότερο από τα δύο ορίσματά της.

Ακολουθεί ένα πρόγραμμα :

```
/* που  βρίσκει τη μικρότερη από δύο τιμές */

#include <stdio.h>

main()

{
```

```
        int num1, num2;
        int imin();

        while (scanf("%d %d", &num1, &num2) == 2)
                printf("Το μικρότερο μεταξύ του %d και %d είναι το %d. \n",
                        num1, num2, imin(num1, num2));
}

int imin(n, m)
        int n, m;
{
        int min;

        if (n < m)
                min = n;
        else
                min = m;
        return min;
}
```

Η λέξη-κλειδί *return* έχει ως αποτέλεσμα η τιμή οποιασδήποτε έκφρασης που την ακολουθεί, να είναι η τιμή επιστροφής της συνάρτησης. Στη συγκεκριμένη βέβαια περίπτωση, η τιμή επιστροφής της συνάρτησης είναι η τιμή της τοπικής μεταβλητής min. Η συνάρτηση imin() είναι τύπου int, όπως είναι και η μεταβλητή min. Η τιμή επιστροφής μιας συνάρτησης μπορεί να καταχωρηθεί σε μια μεταβλητή ή να χρησιμοποιηθεί σαν μέρος μιας έκφρασης.

Ακολουθεί και μια άλλη έκδοση της συνάρτησης imin() :

```
/* δεύτερη έκδοση της συνάρτησης ελάχιστης τιμής */

imin(n, m)
        int n, m;
```

```
{
        return (n<m) ? n : m;
}
```

Η χρήση της return έχει και το αποτέλεσμα ότι τερματίζει τη συνάρτηση και ο έλεγχος επιστρέφει στην επόμενη πρόταση του προγράμματος από εκείνη που την κάλεσε. Αυτό συμβαίνει ακόμα και αν η πρόταση return δεν είναι η τελευταία της συνάρτησης.

Ακολουθεί και μια τρίτη έκδοση της συνάρτησης imin() :

```
/* τρίτη έκδοση της συνάρτησης ελάχιστης τιμής */

imin(n, m)
        int n, m;
{
        if (n<m)
                return (n);
        else
                return(m);
}
```

Η απλή πρόταση return; προκαλεί τον τερματισμό της συνάρτησης και την επιστροφή του ελέγχου στην καλούσα συνάρτηση. Αυτή η μορφή χρησιμοποιείται σε συναρτήσεις του τύπου void γιατί δεν επιστρέφει καμία τιμή.

Τύποι Συναρτήσεων

Και για τις συναρτήσεις πρέπει να δηλώνεται ο τύπος τους, ο οποίος είναι ίδιος με τον τύπο της τιμής επιστροφής της συνάρτησης. Συναρτήσεις χωρίς τιμή επιστροφής δηλώνονται σαν τύπου void και αν δεν δηλωθεί ο τύπος μιας συνάρτησης, τότε η C θεωρεί ότι είναι τύπου int. Η δήλωση του τύπου είναι μέρος του ορισμού της συνάρτησης και αναφέρεται στην τιμή επιστροφής και όχι στα ορίσματα της συνάρτησης. Μια δήλωση της συνάρτησης λέει στο πρόγραμμα τι τύπου είναι η συνάρτηση, ενώ ο ορισμός της συνάρτησης παρέχει τον πραγματικό κώδικά της. Η δήλωση μιας συνάρτησης πρέπει να γίνεται πριν από το σημείο όπου χρησιμοποιείται η συνάρτηση.

Στην ANSI C, μπορούμε στη δήλωση μιας συνάρτησης να δηλώνουμε και τον τύπο των μεταβλητών της. Το αποτέλεσμα είναι ένα *πρωτότυπο συνάρτησης*, δηλ. μια δήλωση συνάρτησης όπου καθορίζονται ο τύπος της τιμής επιστροφής, ο αριθμός των ορισμάτων και ο τύπος των ορισμάτων της συνάρτησης.

Ακολουθούν παραδείγματα :

```
int imax(int, int);

int imax(int a, int b);
```

Αλλαγή των Μεταβλητών στο Πρόγραμμα που Καλεί

Μερικές φορές θέλουμε μια συνάρτηση να προκαλεί αλλαγές στις μεταβλητές μιας άλλης συνάρτησης. Ας πάρουμε το δημοφιλές παράδειγμα της ανταλλαγής των τιμών δύο μεταβλητών x και y.

Ξέρουμε ότι αυτό γίνεται με τη χρήση μιας τρίτης βοηθητικής μεταβλητής temp, ως εξής :

```
temp = x;

x = y;

y = temp;
```

Χρησιμοποιούμε τα ονόματα x και y για τις μεταβλητές της main() και u και v για τις μεταβλητές της συνάρτησης interchange().

```
/* ανταλλαγή τιμών μεταβλητών */


#include <stdio.h>

void interchange();

main()

{

    int x = 5, y = 10;


    printf("Αρχικά x = %d και y = %d. \n", x, y);

    interchange(x, y);

    printf("Τώρα x = %d και y = %d. \n", x, y);

}
```

```
void interchange(u, v)

        int u, v;

{

        int temp;

        printf("Αρχικά u = %d και v = %d. \n", u, v);

        temp = u;

        u = v;

        v = temp;

        printf("Τώρα u = %d και v = %d. \n", u, v);

}
```

Το αποτέλεσμα του προγράμματος θα είναι το εξής :

Αρχικά x = 5 και y = 10

Αρχικά u = 5 και v = 10

Τώρα u = 10 και v = 5

Τώρα x = 5 και y = 10

Βλέπουμε ότι η συνάρτηση interchange() λειτουργεί σωστά, αφού ανταλλάσσει τις τιμές των u και v. Εκείνο που μας ενδιαφέρει, όμως, εμάς είναι η μετάδοση των τιμών αυτών πίσω στη main(). Όπως ξέρουμε, με την πρόταση return μπορούμε να μεταδώσουμε πίσω στο καλούν πρόγραμμα μόνο μία τιμή.

Τι μπορούμε, όμως, να κάνουμε;

Για να μεταδώσουμε δύο τιμές μιας συνάρτησης στη συνάρτηση που την κάλεσε, πρέπει να χρησιμοποιήσουμε τους δείκτες, που είναι συμβολικές παραστάσεις διευθύνσεων.

Εισαγωγή στους Δείκτες (Pointers)

Χρησιμοποιήσαμε προηγουμένως τον τελεστή & για να βρούμε τη διεύθυνση της μεταβλητής p. Η &p δηλαδή είναι ένας *δείκτης της p*. Η πραγματική διεύθυνση είναι ένας 16δικός αριθμός και η συμβολική παράσταση &p είναι μια *σταθερά δείκτη*. Η μεταβλητή p δεν πρόκειται να αλλάξει διεύθυνση μέσα στο πρόγραμμα, αν και μπορεί να αλλάξει τιμή. Η C έχει και *μεταβλητές δείκτη* που έχουν μια διεύθυνση ως τιμή.

Ακολουθούν παραδείγματα :

```
ptr = &p;       /* καταχωρεί τη διεύθυνση της p στην ptr */

ptr = &b;       /* ο δείκτης ptr δείχνει τώρα στη b αντί για την p */
```

Η ptr δηλαδή είναι μια μεταβλητή και η &p είναι μια σταθερά.

Για να δηλώσουμε τον τύπο μιας μεταβλητής δείκτη, πρέπει να χρησιμοποιήσουμε τον *τελεστή έμμεσης αναφοράς (*)*.

```
val = *ptr      /* βρίσκει την τιμή που δείχνει ο δείκτης ptr */
```

Πώς δηλώνουμε, όμως, τους δείκτες;

Με τη δήλωση μιας μεταβλητής ότι είναι δείκτης, πρέπει ακόμα να πούμε και τι τύπος είναι η μεταβλητή που δείχνει ο δείκτης και αυτό γιατί διαφορετικοί τύποι μεταβλητών απαιτούν διαφορετικό χώρο αποθήκευσης.

Οι δείκτες δηλώνονται ως εξής :

```
int *pi;                /* pi είναι ένας δείκτης σε μια ακέραια μεταβλητή */

char *pc;               /* pc είναι ένας δείκτης σε μια μεταβλητή χαρακτήρα */

float *pf, *pg;         /* pf, pg είναι δείκτες σε μεταβλητές float */
```

Από τα παραπάνω συμπεραίνουμε ότι pi είναι ο δείκτης και ότι η *pi είναι τύπου int. Παρόμοια, η τιμή *pc την οποία δείχνει ο pc είναι τύπου char.

Οι Δείκτες και οι Συναρτήσεις

Ακολουθεί ένα πρόγραμμα όπου γίνεται επιστροφή τιμών στη συνάρτηση που καλεί.

```
/χρήση δεικτών για την ανταλλαγή τιμών */

#include <sdtio.h>

void interchange();

main()

{

    int x = 5, y = 10;


    printf("Αρχικά x = %d και y = %d. \n", x, y);

    interchange(&x, &y);        /* στείλε τις διευθύνσεις στη συνάρτηση*/
```

```
        printf("Τώρα x = %d και y = %d. \n", x, y);
}

void interchange(u, v)
        int *u, *v;     /* οι u και v είναι τώρα δείκτες */
{
        int temp;
        temp = *u;    /* η temp παίρνει την τιμή που δείχνει η u */
        *u = *v;
        *v = temp;
}
```

Το αποτέλεσμα θα είναι :

Αρχικά x = 5 και y = 10

Τώρα x = 10 και y = 5

Ας προσέξουμε τα εξής σημεία :

1. Η κλήση της συνάρτησης γίνεται ως εξής : interchange(&x, &y);, δηλαδή, αντί να στέλνονται οι τιμές των x και y, στέλνονται οι διευθύνσεις τους. Έτσι, τα τυπικά ορίσματα u και v της συνάρτησης interchange() έχουν διευθύνσεις ως τιμές και πρέπει να δηλωθούν ως δείκτες.

2. Εφόσον οι x και y είναι ακέραιοι, δηλώνουμε τις u και v ως δείκτες σε ακέραιους.

3. Θυμηθείτε ότι η u έχει την τιμή &x, έτσι η u δείχνει τη x, δηλαδή η *u μάς δίνει την τιμή του x.

4. Δεν πρέπει να γράψουμε temp = u;, γιατί έτσι θα αποθηκευτεί η διεύθυνση της x και όχι η τιμή της.

Θέλουμε μια συνάρτηση να ανταλλάσσει τις τιμές δύο μεταβλητών. Δίνοντας στη συνάρτηση τις διευθύνσεις των μεταβλητών αυτών, αποκτάμε πρόσβαση σ' αυτές τις μεταβλητές. Χρησιμοποιώντας δείκτες και τον τελεστή *, η συνάρτηση μπορεί να εξετάσει τις τιμές που βρίσκονται σ' αυτές τις θέσεις και να τις αλλάξει.

Με τους δείκτες δηλαδή μπορούμε να μπούμε στη main() και να αλλάξουμε ό,τι είναι αποθηκευμένο εκεί.