ΣΥΝΑΡΤΗΣΕΙΣ

1. Οι συναρτήσεις στη C

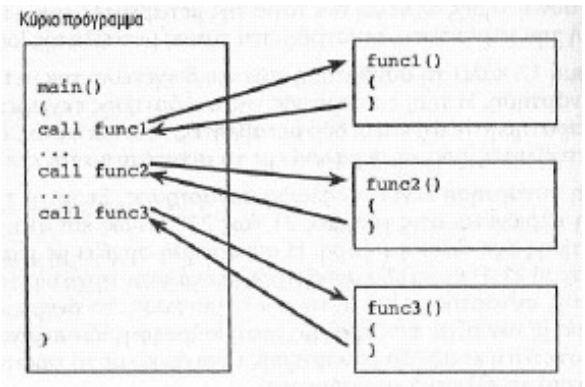
1.1 Ορισμός

Μια συνάρτηση είναι ένα ανεξάρτητο τμήμα κώδικα της C που εκτελεί μια συγκεκριμένη εργασία και επιστρέφει προαιρετικά μια τιμή στο πρόγραμμα που την κάλεσε. Τα κύρια χαρακτηριστικά μιας συνάρτησης στην γλώσσα 'C' είναι τα ακόλουθα:

- **Μοναδικότητα Ονόματος:** Κάθε συνάρτηση έχει ένα μοναδικό όνομα. Χρησιμοποιώντας αυτό το όνομα σε ένα άλλο μέρος του προγράμματος, μπορεί κανείς να εκτελέσει τις προτάσεις που περιέχονται στην συνάρτηση. Αυτό ονομάζεται κλήση της συνάρτησης. Τα ονόματα των μεταβλητών εκχωρούνται από τον προγραμματιστή σύμφωνα με τους κανόνες ονομασίας των μεταβλητών
- **Ανεξαρτησία:** Μια συνάρτηση μπορεί να εκτελεί συγκεκριμένες λειτουργίες χωρίς να αλληλεπιδρά με άλλα μέρη του προγράμματος
- **Λειτουργία:** Μια συνάρτηση εκτελεί μια συγκεκριμένη λειτουργία σαν υποσύνολο του προγράμματος που έχετε υλοποιήσει.
- **Επιστροφή τιμής:** Μια συνάρτηση μπορεί να επιστρέφει μια τιμή στον πρόγραμμα που την καλεί. Ο τύπος επιστροφής μπορεί να οποιοσδήποτε από τους τύπους δεδομένων της 'C': char, int, long, float, double κτλ.

1.2 Λειτουργία Συναρτήσεων

Ένα πρόγραμμα της 'C' δεν εκτελεί τις προτάσεις σε μια συνάρτηση έως ότου κληθεί η συνάρτηση από ένα άλλο τμήμα του προγράμματος. Όταν καλείται μία συνάρτηση το πρόγραμμα στέλνει τις πληροφορίες της συνάρτησης υπό μορφή ορίσματος. Ένα όρισμα είναι δεδομένα προγράμματος που απαιτούνται από την συνάρτηση για εκτελέσει την εργασία της. Για κάθε όρισμα που διοχετεύεται στην συνάρτηση, η λίστα των παραμέτρων πρέπει να περιέχει μια καταχώρηση. Αυτή καταχώρηση καθορίζει τον τύπο των δεδομένων και το όνομα της παραμέτρου. Οι προτάσεις στην συνάρτηση εκτελούν τις εργασίες που έχουν σχεδιαστεί να κάνουν. Όταν τελειώσουν οι προτάσεις της συνάρτησης, η εκτέλεση περνά ξανά στην ίδια θέση στο πρόγραμμα, από όπου κλήθηκε η συνάρτηση. Αυτό φαίνεται και σχηματικά στο παρακάτω Σχήμα.



Το Σχήμα δείχνει ένα πρόγραμμα με τρεις συναρτήσεις, καθεμία από τις οποίες καλείται μια φορά. Κάθε φορά που καλείται μία συνάρτηση, η εκτέλεση περνά σε αυτήν τη συνάρτηση. Όταν τελειώσει η συνάρτηση, η εκτέλεση

περνά στη θέση από όπου κλήθηκε. Μια συνάρτηση μπορεί να κληθεί όσες φορές χρειάζεται και οι συναρτήσεις μπορούν να καλούνται με οποιαδήποτε σειρά.

1.3 Σύνταξη της Συνάρτησης

Η σύνταξη της συνάρτησης αποτελείται από το πρωτότυπο της και τη δήλωση της, όπως περιγράφονται παρακάτω.

Το πρωτότυπο της συνάρτησης παρέχει στο μεταγλωττιστή (compiler) μία περιγραφή μιας συνάρτησης η οποία θα οριστεί αργότερα στο πρόγραμμα. Το πρωτότυπο περιλαμβάνει ένα τύπο επιστροφής που δηλώνει τον τύπο της μεταβλητής που θα επιστρέψει η συνάρτηση. Επίσης περιλαμβάνει το όνομα της συνάρτησης που θα πρέπει να περιγράφει τι κάνει η συνάρτηση και τους τύπους των ορισμάτων που θα διοχετευθούν στη συνάρτηση.

```
Typos_Epistrofis  onoma_synartisis  (typos_orismatos  1,…,typos_orismatos_n);
```

Μια δήλωση συνάρτησης περιέχει τις προτάσεις της C που εκτελείται. Η πρώτη γραμμή μις δήλωσης μιας συνάρτησης που λέγεται κεφαλίδα συνάρτησης θα πρέπει να είναι ίδια με του πρωτότυπου της συνάρτησης, με την εξαίρεση της απουσία του ερωτηματικού. Στην συνέχεια ακολουθεί σε άγκιστρα το σώμα της συνάρτησης. Εάν η συνάρτηση δεν επιστρέφει την τιμή μιας μεταβλητή είναι τύπου void. Διαφορετικά η τιμή που επιστρέφεται δια μέσου της πρότασης return πρέπει να συμβαδίζει με τον τύπο επιστροφής στο πρωτότυπο της συνάρτησης

ΠΑΡΑΔΕΙΓΜΑ

```
/* Synartisi ypologismoy tis dynamis enos akeraioy */
#include <stdio.h>

int power(int, int); /* Prwtotipo Synartisis */

main()
{
    int a; int b; int c;

    printf("Dose timi gia to a: "); scanf("%d",&a);
    printf("Dose timi gia to b: "); scanf("%d",&b);

    c=power(a,b);  /* Klisi tis synartisis */

    printf("H dynami toy %d  stin %d  isoytai me  %d\n",a,b,c);
}

int power(int x,int y)   /* Kefalida tis synartisis */
{   /* Soma tis synartisis */
    int var,i;

    var=1;
    for(i=1;i<=y;i++)
       var*=x;

    return var;
}
```

Με τα εξής αποτελέσματα :

```
Dose timi gia to a: 5
Dose timi gia to b: 3
H  dynami toy 5 stin 3 isoytai me  125
```

Μερικές φορές προκύπτει σύγχυση για τη διάκριση ανάμεσα στις παραμέτρους και τα ορίσματα. Μια παράμετρος είναι μια καταχώριση σε μια κεφαλίδα συνάρτησης. Λειτουργεί ως ‘κράτηση θέσης’ για ένα όρισμα.

Οι παράμετροι μιας συνάρτησης είναι σταθερές. Δεν αλλάζουν κατά την εκτέλεση του προγράμματος. Από την άλλη πλευρά ένα όρισμα είναι μια πραγματική τιμή που διοχετεύεται στη συνάρτηση. Κάθε φορά που καλείται μια συνάρτηση μπορεί να δεχθεί διαφορετικά ορίσματα. Κάθε φορά που καλείται μια συνάρτηση πρέπει να διοχετεύεται ο ίδιος αριθμός και τύπος ορισμάτων.

ΠΑΡΑΔΕΙΓΜΑ

```
/* Apeikonizei tin diafora meta3y orismatos kai parametrou */
#include <stdio.h>

float half_value(float); /* Prwtotipo Synartisis */

void main ()
{
    float value1,value2; float r_val1,r_val2;

    value1=30.55;
    r_val1=half_value(value1);/* H  value1 einai orisma tis half_value */
    printf("The value of r_val1 %f\n",r_val1);

    value2=180.32;
    r_val2=half_value(value2);/* H  value2 einai orisma tis half_value */
    printf("The value of r_val2 %f\n",r_val2);

}

float half_value(float a)    /* Kefalida tis synartisis */
{   /* a einai i parametros  */
    return a/2;
}
```

1.4 Τοπικές Μεταβλητές

Μπορείτε να δηλώσετε μεταβλητές μέσα στο σώμα μιας συνάρτησης. Οι μεταβλητές που δηλώνονται σε μια συνάρτηση ονομάζονται τοπικές μεταβλητές. Ο όρος τοπικές μεταβλητές σημαίνει ότι οι μεταβλητές είναι διαφορετικές από οποιεσδήποτε άλλες μεταβλητές με το ίδιο όνομα που δηλώνονται σε άλλες συναρτήσεις.

ΠΑΡΑΔΕΙΓΜΑ

```
/* Xrisi twn topikwn metablitwn */
#include <stdio.h>

void demo(void); /* Prwtotipo Synartisis */

main ()
{
    int x,y;
    x=1; y=2;
    printf("Before calling demo(),x=%d and y=%d\n",x,y);
    demo();
    printf("\nAfter calling demo(),x=%d and y=%d\n",x,y);
}

void demo(void)   /*  Kefalida  tis synartisis  */
{
    int x,y;
    x=144; y=591;
    printf("\nWithin  demo(),  x=%d  and  y=%d\n",x,y);
}
```

Στην οθόνη του υπολογιστή εμφανίζεται το παρακάτω αποτέλεσμα:

```
Before calling demo(),x=1 and y=2
Within demo(),x=144 and y=591
After calling demo(),x=1  and y=2
```

1.5 Κλήση Συναρτήσεων

Υπάρχουν δύο τρόποι για να κληθεί μια συνάρτηση. Οποιασδήποτε συνάρτηση μπορεί να κληθεί χρησιμοποιώντας απλώς το όνομα της και τη λίστα ορισμάτων μόνο σε μια πρόταση. Η δεύτερη μέθοδος μπορεί να χρησιμοποιηθεί μόνο με συναρτήσεις που έχουν τιμή επιστροφής. Επειδή αυτές οι συναρτήσεις εκτιμώνται σε μια τιμή, μπορούν να χρησιμοποιηθούν οπουδήποτε μπορεί να χρησιμοποιηθεί μια έκφραση της C.

ΠΑΡΑΔΕΙΓΜΑ

```
/* Klisi Synartisewn */
#include <stdio.h>

void demo(void);  /* Ektipwsi enos minimatos */
int factorial(int); /* Ypologismos Paragontikou   */

main ()
{
    int x=5,y;
    demo();
    y=factorial(x);
    printf("\nThe factorial of x=%d is y=%d\n",x,y);
}

void demo(void)
{
    printf("\nYpologismos paragontikoy\n");
}

int factorial(int a)
{
    int product,i;

    product=1;
    for(i=2;i<=a;i++)
      product = product * i;

    return product;
}
```

Στην οθόνη του υπολογιστή εμφανίζεται το παρακάτω αποτέλεσμα:

```
Ypologismos paragontikoy

The factorial of x=5 is y=120
```

1.6 Αναδρομή

Ο όρος αναδρομή αναφέρεται σε μια κατάσταση στην οποία μία συνάρτηση καλεί τον εαυτό της είτε άμεσα είτε έμμεσα. Η έμμεση αναδρομή λαμβάνει χώρα όταν μια συνάρτηση καλεί μία άλλη συνάρτηση η οποία καλεί την πρώτη συνάρτηση. Η C επιτρέπει τις αναδρομικές συναρτήσεις οι οποίες μπορεί να είναι χρήσιμες σε μερικές περιπτώσεις.

ΠΑΡΑΔΕΙΓΜΑ

```
/* Paradeigma Anadromis */
#include <stdio.h>

unsigned int factorial(int) /* Ypologismos Paragontikou  */
main ()
{
    unsigned int x,f;
    scanf("%d",&x);
    f=factorial(x);
    printf("\nThe factorial of x=%d is f=%d\n",x,f);
}
```

```
unsigned int  factorial(unsigned int a) /* Ypologismos Paragontikou*/
{
    if(a==1) return  1;
    else
    {
        a*=factorial(a-1);
        return a;
    }
}
```

1.7 Διοχέτευση Πινάκων σε Συναρτήσεις

Για να διοχετεύσετε ένα όρισμα σε μια συνάρτηση πρέπει να καθορίσετε το όνομα του πίνακα χωρίς αγκύλες. Η C αυτομάτως διοχετεύει πίνακες σε συναρτήσεις χρησιμοποιώντας 'εικονική' κλήση μέσω αναφοράς - οι κληθείσες συναρτήσεις μπορούν να τροποποιήσουν τις τιμές των στοιχείων στους αρχικούς πίνακες της συνάρτησης που τις καλεί. Το όνομα του πίνακα είναι στην πραγματικότητα η διεύθυνση του πρώτου στοιχείου του πίνακα. Ως εκ τούτου όταν η κληθείς συνάρτηση τροποποιεί τα στοιχεία του πίνακα στο σώμα της, τότε τροποποιούνται και τα στοιχεία του πίνακα που περάστηκε ως όρισμα.

ΠΑΡΑΔΕΙΓΜΑ

```
/* Klisi synartisis me orisma pinaka */
#include <stdio.h>
#define  SIZE  5
void modifyArray(int [], int);

void main()
{
    int a[SIZE]={0,1,2,3,4};
    int i;

    printf("before  calling  the  function  modifyArray\n");
    for(i=0;i<SIZE;i++)
        printf("value=%d\n",a[i]);

    modifyArray(a,SIZE);

    printf("after call the function modifyArray\n");
    for(i=0;i<SIZE;i++)
    printf("value=%d\n",a[i]);
}

void modifyArray(int b[], int size)
{
    int j;
    for(j=0;j<size;j++)
        b[j]*=2;
}
```

Στην οθόνη του υπολογιστή εμφανίζεται το παρακάτω αποτέλεσμα:

```
before  calling  the  function  modifyArray
value=0
value=1
value=2
value=3
value=4
after call the function modifyArray
value=0
value=2
value=4
value=6
value=8
```

Μπορεί να υπάρχουν περιπτώσεις στα προγράμματα σας όπου μια συνάρτηση δεν θα επιτρέπεται να τροποποιεί τα στοιχεία ενός πίνακα. Επειδή οι πίνακες διοχετεύονται πάντοτε στην εικονική κλήση μέσω αναφοράς, η τροποποίηση των τιμών σε ένα πίνακα είναι δύσκολο να ελεγχθεί. Η 'C' παρέχει το προσδιορισμό τύπου 'const' για να αποτρέπει την τροποποίηση των τιμών του πίνακα σε μία συνάρτηση. Όταν μια παράμετρος πίνακα έπεται του προσδιορισμού const για να αποτρέπει την τροποποίηση των τιμών του πίνακα σε μία συνάρτηση.

ΠΑΡΑΔΕΙΓΜΑ

```
/* Klisi synartisis me  orisma  pinaka  */
#include <stdio.h>
#define SIZE 5
void modifyArray(const int []);
void main()
{
    int a[SIZE]={0,1,2,3,4};
    int i;

    printf("before  calling  the  function  modifyArray\n");
    for(i=0;i<SIZE;i++)
        printf("value=%d\n",a[i]);

    modifyArray(a);

    printf("after call the function modifyArray\n");
    for(i=0;i<SIZE;i++)
        printf("value=%d\n",a[i]);
}

void modifyArray(const int b[])
{
    b[0]=5;
    b[1]=14;
    b[3]=6;
}
```

Στην οθόνη του υπολογιστή κατά τη διάρκεια της μεταγλώττισης εμφανίζεται το παρακάτω αποτέλεσμα:

Line	Col	File	Message
		F:\M...	**In function 'modifyArray':**
22	6	F:\M...	[Error] assignment of read-only location '*b'
23	6	F:\M...	[Error] assignment of read-only location '*(b + 4u)'
24	6	F:\M...	[Error] assignment of read-only location '*(b + 12u)'

1.8 Μεταβίβαση Παραμέτρων σε μία Συνάρτηση

Δύο είναι οι διαφορετικοί τρόποι μεταβίβασης παραμέτρων σε μία συνάρτηση

- Κλήση μέσω τιμής (call by value)
- Κλήση μέσω αναφοράς (call by reference)

1.8.1 Κλήση Συνάρτησης μέσω τιμής (call by value)

Όταν γίνεται κλήση συνάρτησης μέσω τιμής, τότε στη συνάρτηση διοχετεύονται οι τιμές των παραμέτρων του προγράμματος που την καλεί. Οποιαδήποτε αλλαγή γίνει στις τιμές των παραμέτρων μέσα στο σώμα της συνάρτησης δεν επηρεάζει τις τιμές των παραμέτρων που διοχετεύθηκαν στη συνάρτηση, γιατί οι τυχόν αλλαγές γίνονται σε διαφορετικές διευθύνσεις μνήμης (όπως είδαμε στο προηγούμενο παράδειγμα).

1.8.2 Κλήση Συνάρτησης μέσω αναφοράς (call by reference)

Όταν επιθυμούμε μία συνάρτηση να μπορεί να αλλάξει τις τιμές των παραμέτρων, τότε πρέπει να γίνει κλήση της συνάρτησης μέσω αναφοράς. Σε αυτή την περίπτωση, στη συνάρτηση διοχετεύονται οι διευθύνσεις μνήμης των παραμέτρων του προγράμματος που την καλεί και όχι οι τιμές τους (γεγονός που πραγματοποιείται κατά την κλήση μέσω τιμής). Επομένως, αφού η συνάρτηση έχει πρόσβαση στις διευθύνσεις των παραμέτρων του προγράμματος που την κάλεσε, τότε μπορεί να μεταβάλλει τις τιμές αυτών.

Αν και η κλήση μέσω τιμής και η κλήση μέσω αναφοράς αναφέρονται σαν διαφορετικοί τρόποι μεταβίβασης παραμέτρων σε μία συνάρτηση, στην πραγματικότητα είναι ίδιοι. Δηλαδή, υπάρχει μόνο ένας τρόπος μεταβίβασης και αυτός είναι η κλήση μέσω τιμής. Θα μπορούσαμε δηλαδή να θεωρήσουμε ότι υπάρχει μόνο αυτός ο τρόπος και ότι στην προσωρινή μνήμη μπορούν να αποθηκευτούν είτε οι τιμές των μεταβλητών είτε οι διευθύνσεις των μεταβλητών. Σημειώστε επίσης ότι, όταν γίνεται κλήση μίας συνάρτησης, επιτρέπεται να γίνει συνδυασμός των δύο μεθόδων, δηλαδή κάποιες παράμετροι να διοχετευθούν μέσω τιμής και κάποιες άλλες μέσω αναφοράς.

ΠΑΡΑΔΕΙΓΜΑ

```
#include <stdio.h>

void fun(int*, int);

void main()
{
    int i = 100, j = 200;
    int* ptr;

    ptr = &i;
    fun(ptr, j);

    printf("%d %d\n", i, j);

}

void fun(int* ptr1, int a)
{
    *ptr1 = 300;
    a = 400;
}
```

```
Στην οθόνη του υπολογιστή εμφανίζεται το παρακάτω αποτέλεσμα:

300 200
```

1.9 Συναρτήσεις και Δομημένος Προγραμματισμός

Η χρήση των συναρτήσεων είναι πολύ σημαντική για το σχεδιασμό προγραμμάτων με βάση το δομημένο προγραμματισμό στον οποίο εκτελούνται μεμονωμένες εργασίες προγράμματος από αυτόνομα τμήματα κώδικα. Τα πλεονεκτήματα του δομημένου προγραμματισμού είναι τα ακόλουθα:

- Τα πολύπλοκα προβλήματα μπορούν να διασπαστούν σε ένα αριθμό μικρότερων και απλούστερων εργασιών.
- Είναι ευκολότερο να ανιχνευθούν λάθη σε ένα δομημένο πρόγραμμα. Εάν το πρόγραμμα έχει λάθος (κάτι που προκαλεί δυσλειτουργία)

1.10 Η Μαθηματική Βιβλιοθήκη

Η βιβλιοθήκη <math.h> περιλαμβάνει μία ακολουθία από μαθηματικές συναρτήσεις. Οι συναρτήσεις αυτές είναι οι ακόλουθες:

- double sin(double x): συνάρτηση υπολογισμού του ημιτόνου. Παίρνει σαν όρισμα την γωνία x και επιστρέφει το ημίτονο. **Προσοχή:** η γωνία δίνεται σε ακτίνια και όχι σε μοίρες. π.χ. οι 360^o = 2π ακτίνια.
- double cos(double x): συνάρτηση υπολογισμού του συνημίτονου. Παίρνει σαν όρισμα την γωνία x και επιστρέφει το συνημίτονο.
- double tan(double x): συνάρτηση υπολογισμού της εφαπτομένης. Παίρνει σαν όρισμα την γωνία x και επιστρέφει την εφαπτομένη.
- double log(double x): υπολογισμός του φυσικού λογάριθμου (lnx). Η βάση του φυσικού λογάριθμου είναι το e=2.718. Το x πρέπει να είναι μεγαλύτερο του μηδέν.
- double log10(x): υπολογισμός του λογάριθμου με βάση το 10. Το x πρέπει να είναι μεγαλύτερο του μηδέν.
- double pow(double x, double y): Υπολογίζει τη δύναμη x^y . Η συνάρτηση παράγει λάθος εάν το x=0 και y≤0 ή x<0 και το y δεν είναι ακέραιος.
- double sqrt(double x): Επιστρέφει την τετραγωνική ρίζα του x. Το x θα πρέπει να είναι μεγαλύτερο του μηδέν.
- double ceil(double x): Επιστρέφει το πάνω ακέραιο μέρος του x.
- double floor(double x): Επιστρέφει το κάτω ακέραιο μέρος του x.
- double fabs(double x): Επιστρέφει την απόλυτη τιμή του x.
- double ldexp(double x, int n): Επιστρέφει την μαθηματική έκφραση x * 2^n .

ΠΑΡΑΔΕΙΓΜΑ

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

#define PI 3.14159265

void main()
{
   double x, aktinia, result;

   x = 45.0;  // gwnia se moires
   aktinia = x * PI / 180;  // metatropi gwnias se aktinia
   result = sin(aktinia);
   printf("To hmitono twn %lf moirwn einai %lf", x, result);
}
```

Στην οθόνη του υπολογιστή εμφανίζεται το παρακάτω αποτέλεσμα:

```
To hmitono twn 45.000000 moirwn einai 0.707107
```