Προγραμματισμός Υπολογιστών

Δημιουργία συναρτήσεων

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Συναρτήσεις (Functions)

Για την αποτελεσματική επίλυση ενός προβλήματος με την βοήθεια ενός ΗΥ είναι σημαντικό να το διασπάσουμε σε μικρότερα τμήματα με σκοπό την εύκολη επίλυση και διαχείριση του προγράμματος. Έτσι λοιπόν τις εντολές με τις οποίες επιλύουμε ένα τμήμα του προβλήματος μας τις ομαδοποιούμε κάτω από ένα αναγνωριστικό ώστε να αποτελέσουν μια ανεξάρτητη συνάρτηση. Η εκτέλεση του συγκεκριμένου τμήματος γίνεται με την κλήση του ονόματος της συνάρτησης. Η γενική δήλωση μιας συνάρτησης είναι :

Τύπος_Επιστρεφόμενης_Τιμής Ονομα_Συνάρτησης(ορίσματα) {

Εντολές

}

Τα ορίσματα δεν είναι υποχρεωτικά και η δήλωση ότι η συνάρτηση δεν χρειάζεται ορίσματα γίνεται με την δεσμευμένη λέξη void. Την ίδια λέξη χρησιμοποιούμε για να δηλώσουμε ότι η συνάρτηση δεν επιστρέφει τιμή άρα δεν έχει ένα συγκεκριμένο τύπο δεδομένων. Σε αυτή την περίπτωση σε άλλες γλώσσες προγραμματισμού, η συνάρτηση ονομάζεται λειτουργία (procedure). Ο καλύτερος τρόπος για να καταλάβουμε τα παραπάνω είναι με ένα παράδειγμα.

Δημιουργία μιας συνάρτησης για τον υπολογισμό της μικρότερης τιμής μεταξύ δύο ακεραίων αριθμών

```
int main(int argc, char *argv[]){
int min, a, b;
…..
scanf("%d",&a);
scanf("%d",&b);
if(a < b){
    min = a;
}
else {
min = b;
}
……
}
```

Έστω ένα πρόγραμμα το οποίο μεταξύ των άλλων εργασιών κάνει και τον υπολογισμό της μικρότερης τιμής μεταξύ δύο ακεραίων αριθμών, των a και b.

Αν πριν ή μετά από το συγκεκριμένο τμήμα χρειαζόμασταν την εύρεση της μικρότερης τιμής μεταξύ δύο άλλων αριθμών έστω x και y. Τότε θα έπρεπε να επαναλάβουμε τις ίδιες εντολές όχι για τα a b αλλά για τα x και y.

Αν υποθέσουμε ότι επαναλαμβάνουμε την λειτουργία για άλλα 5 ζευγάρια αριθμών. Αν σε περίπτωση έχουμε κάποιο λάθος θα πρέπει να το διορθώσουμε σε 5 διαφορετικά σημεία του προγράμματος. Έτσι λοιπόν για να αποφύγουμε την συχνή επανάληψη μιας σειράς εντολών και για την καλύτερη διαχείριση δημιουργούμε συναρτήσεις.

Στο συγκεκριμένο πρόβλημα αν έχουμε δύο οποιουσδήποτε ακεραίους αριθμούς έστω k και m η κύρια εργασία που πρέπει να κάνουμε για την εύρεση της μικρότερης τιμής είναι

```
if(k < m){
   minTimi = k;
}
else {
  minTimi = m;
}
```

Το συγκεκριμένο σύνολο εντολών το απομονώνουμε και δημιουργούμε μια συνάρτηση. Τα ορίσματα της συνάρτησης, δηλαδή τα δεδομένα που χρειάζεται για να λειτουργήσει είναι τα k και m γιατί αυτά απαιτούνται για την επίλυση του προβλήματος.

Ο τύπος δεδομένων της minTimi είναι ακέραιος αφού σε αυτήν θα αποθηκεύσουμε το αποτέλεσμα της σύγκρισης. Στο τέλος του συνόλου των εντολών η μεταβλητή minTimi περιέχει την απάντηση του προβλήματος μας. Σε αυτό το σημείο θα πρέπει να επιστρέψουμε αυτή την τιμή στο πρόγραμμα που "κάλεσε" την συνάρτηση. Η επιστροφή της τιμής γίνεται με την εντολή return Τιμή_Επιστροφής; Το αναγνωριστικό που επιλέγουμε για όνομα της συνάρτησης είναι MyMin. Έτσι συνολικά έχουμε :

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

```
int MyMin(int k, int m) {
int minTimi;
    if(k < m){
        minTimi = k;
    }
    else {
        minTimi = m;
    }
    return(minTimi);
}
```

```
int main(int argc, char *argv[]){
int min, a, b;
scanf("%d",&a);
scanf("%d",&b);
min = MyMin(a,b);
min = MyMin(10, 20);
min = MyMin(10, MyMin(a,b));
printf("min = %d", min);
 return 0;
}
```

Σε ένα πρόγραμμα μπορούμε να καλέσουμε οποιαδήποτε συνάρτηση όσες φορές θέλουμε χρησιμοποιώντας σαν ορίσματα είτε μεταβλητές είτε τιμές είτε μια συνάρτηση, ακόμη και την ίδια τη συνάρτηση.

Κλήση της συνάρτησης με τιμή

Στην συνάρτηση main δηλώνουμε τρεις ακέραιες μεταβλητές min, a και b με συνέπεια να δεσμεύσουμε στον ΗΥ τρεις διαφορετικές θέσεις μνήμης. Στην συνέχεια αποδίδουμε στην μεταβλητή a την τιμή 3 και στην b την τιμή 5. Στην επόμενη εντολή για να αποδώσουμε τιμή στην μεταβλητή min θα πρέπει πρώτα να εκτελέσουμε την συνάρτηση **MyMin**.

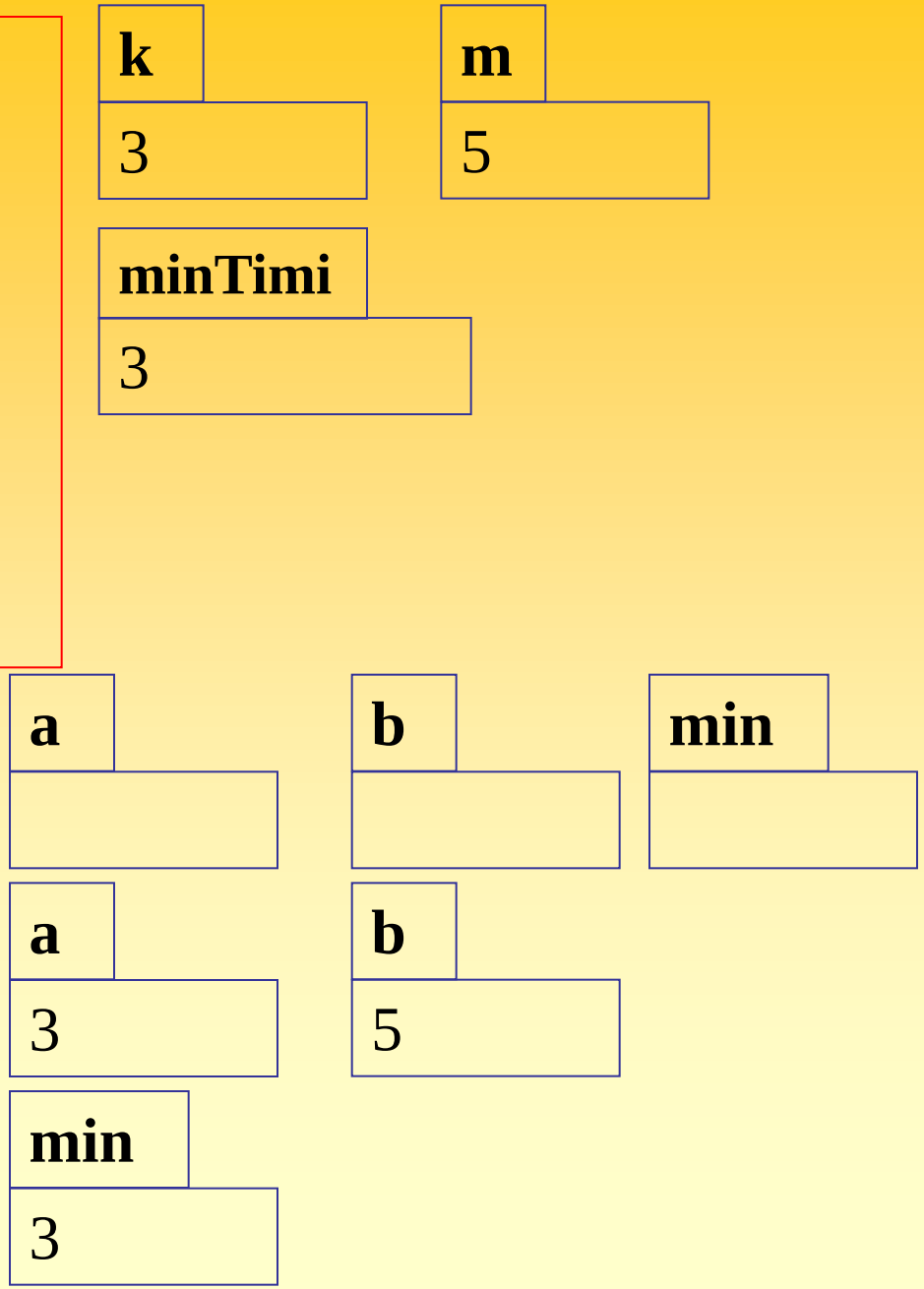
Η **MyMin** δέχεται δύο ορίσματα, τους ακεραίους k και m. Πριν αρχίσει η εκτέλεση της συνάρτησης δεσμεύονται 2 νέες θέσεις μνήμης στον ΗΥ και αντιγράφονται οι τιμές που έχουν οι μεταβλητές a και b, αντίστοιχα. Τα σημεία που πρέπει να παρατηρήσουμε, είναι :

1. Όταν καλούμε μια συνάρτηση, οι θέσεις μνήμης που δεσμεύουμε για τα ορίσματα καθώς και για τις τοπικές μεταβλητές (αυτές που δηλώνουμε μέσα στο σώμα της συνάρτησης) παύουν να υπάρχουν για το πρόγραμμα, με την λήξη της συνάρτησης.
2. Οι θέσεις μνήμης των ορισμάτων είναι διαφορετικές από τις θέσεις μνήμης των μεταβλητών που χρησιμοποιούμε για να καλέσουμε την συνάρτηση. Έτσι ακόμα και αν χρησιμοποιήσουμε στην συνάρτηση σαν ονόματα των ορισμάτων τα a και b, τότε αυτές οι μεταβλητές είναι διαφορετικές και ανεξάρτητες από τα a και b της main. Εκτός από την αρχική τιμή που είναι ίδια, στην συνέχεια δεν έχουν καμία σχέση.

Κλήση της συνάρτησης με τιμή

```
int MyMin(int k, int m) {
int minTimi;
if(k < m){
    minTimi = k;
}
else {
   minTimi = m;
}
    return(minTimi);
}
```

```
int main(int argc, char *argv[]){
int min, a, b;
a = 3;  b = 5;
min = MyMin(a,b);
}
```



Κλήση της συνάρτησης με αναφορά

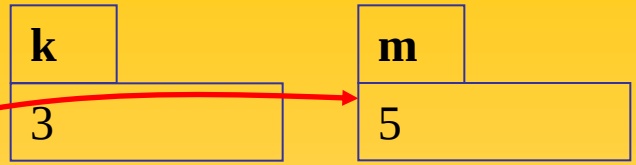
Στην συνάρτηση main δηλώνουμε τρεις ακέραιες μεταβλητές min, a και b με συνέπεια να δεσμεύσουμε στον ΗΥ τρεις διαφορετικές θέσεις μνήμης. Στην συνέχεια αποδίδουμε στην μεταβλητή a την τιμή 3 και στην b την τιμή 5. Στην επόμενη εντολή για να αποδώσουμε τιμή στην μεταβλητή min θα πρέπει να εκτελέσουμε την συνάρτηση **MyMin**.

Η **MyMin** δέχεται τρία ορίσματα, τους ακεραίους k και m και τον δείκτη minTimi. Πριν αρχίσει η εκτέλεση της συνάρτησης δεσμεύονται 3 νέες θέσεις μνήμης στον ΗΥ και τους αποδίδουμε τις τιμές που έχουν οι μεταβλητές a και b, καθώς και την θέση μνήμης της μεταβλητής min αντίστοιχα.

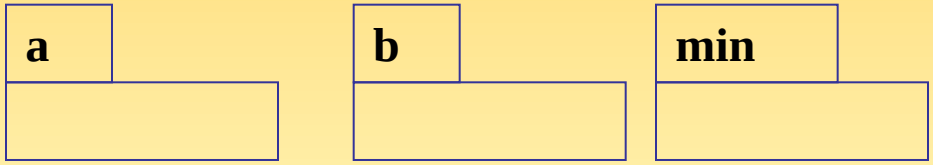
Μέσα στο σώμα της συνάρτησης συγκρίνουμε την τιμές των k και m και αποδίδουμε την μικρότερη τιμή στη θέση μνήμης που δείχνει η minTimi, δηλαδή στην πραγματικότητα στην min. Οπότε μέσα στο σώμα της συνάρτησης, οποιαδήποτε μεταβολή στην τιμή του δείκτη έχει αντίκρισμα στην τιμή της μεταβλητής που περάσαμε σαν όρισμα στην θέση του δείκτη.

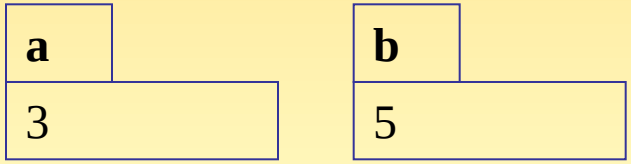
Κλήση της συνάρτησης με αναφορά

```
void refMyMin(int k, int m, int *minTimi) {
 if(k < m){
    *minTimi = k;
}
else {
   *minTimi = m;
}
}
```




```
int main(int argc, char *argv[]){
int min, a, b;
a = 3;  b = 5;
refMyMin(a,b, &min);
printf("Min = %d", min);
}
```

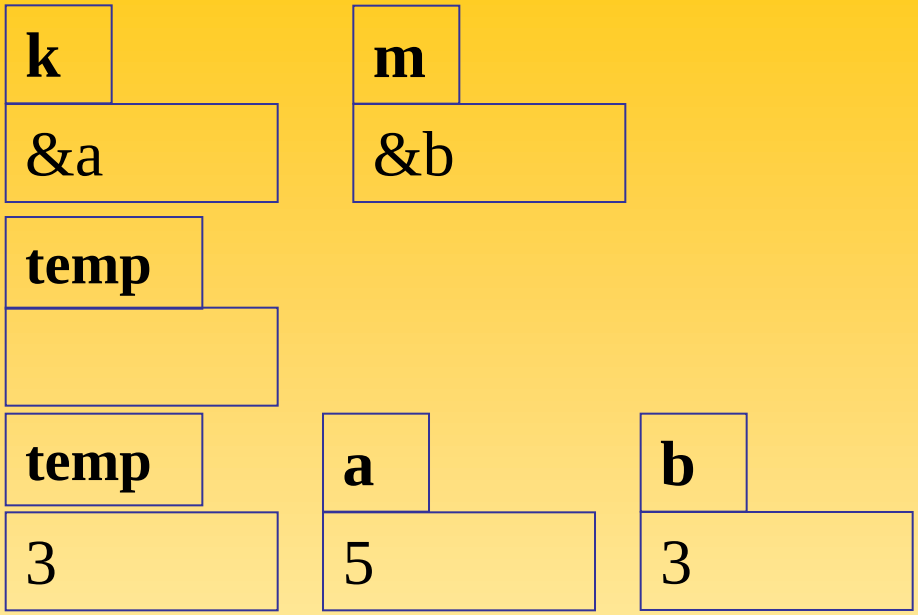




Κάθε συνάρτηση μετατρέπεται σε λειτουργία (procedure) με την προσθήκη μιας ακόμα μεταβλητής, η οποία θα κληθεί όχι με τιμή αλλά με αναφορά με σκοπό να επιστρέψει σε αυτή το αποτέλεσμα της συνάρτησης. Κάθε λειτουργία έχει μπροστά από το όνομα της τη λέξη void για να δηλώσει ότι δεν επιστρέφει τιμή με την εντολή return.

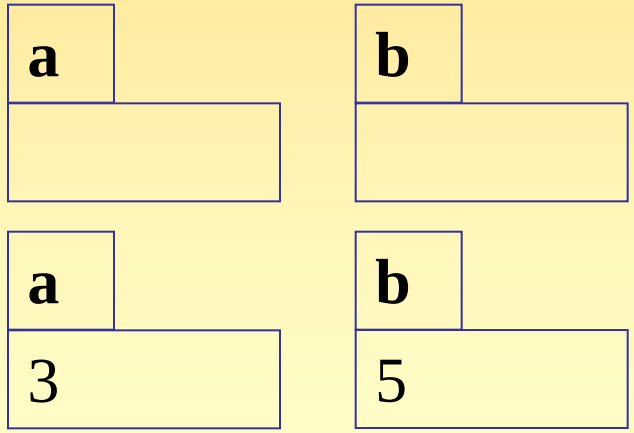
Κλήση της συνάρτησης με αναφορά

```
void swap(int *k, int *m) {
int temp;
temp = *k;
*k = *m;
*m = temp;
}
```



Η τιμή της διεύθυνσης που δείχνει ο δείκτης k είναι η διεύθυνση της μεταβλητής a που δηλώσαμε στην main. Οπότε η εντολή *k αναφέρετε στη τιμή του a. Το *m αναφέρετε στη τιμή του b.

```
int main(int argc, char *argv[]){
int a, b;
a = 3;  b = 5;
swap(&a,&b);
}
```



Δεν είναι ανάγκη να υπάρχει επιστρεφόμενη τιμή. Στην ανωτέρω περίπτωση γίνεται η ανταλλαγή των τιμών μεταξύ δύο μεταβλητών.

1. Να δημιουργήσετε μια συνάρτηση με το όνομα power, η οποία θα δέχεται σαν ορίσματα 1 πραγματικό αριθμό, έστω a και ένα ακέραιο έστω b, και θα επιστρέφει την δύναμη τους, δηλαδή το a υψωμένο στην δύναμη του b.
2. Να δημιουργήσετε μια συνάρτηση με το όνομα abs, η οποία θα δέχεται σαν ορίσμα 1 ακέραιο αριθμό, έστω a, και θα επιστρέφει την απόλυτη τιμή του.
3. Να δημιουργήσετε μια συνάρτηση με το όνομα poly, η οποία θα υπολογίζει την τιμή του πολυωνύμου ax^2+bx+c

```
float power(float a, int b){
   int i;
   float apot = 1.0;

   for(i = 1; i <= b; i++) {
      apot = apot * a;
   }
   return apot;
}
```

```
int abs(int a){
    if(a < 0)
        return(a * -1);
    return a;
}
```

```
float poly(float a, float b, float c, float x){
    return (a * power(x, 2)) + (b * x) + c ;
}
```