Προγραμματισμός Υπολογιστών

Συναρτήσεις και αναδρομή

Νικόλαος Ζ. Ζάχαρης
Καθηγητής

Πανεπιστήμιο Δυτικής Αττικής
Τμήμα Μηχανικών Πληροφορικής και Υπολογιστών

Γιατί δημιουργούμε συναρτήσεις?

Οι βασικοί λόγοι είναι τρεις :

- Μικρότερος Κώδικας

-Καλύτερος έλεγχος του προγράμματος.

-Επαναχρησιμοποίηση του κώδικα και σε άλλα προγράμματα

Ας δούμε αυτά τα πλεονεκτήματα στην πράξη μέσω ενός παραδείγματος.

<u>Παράδειγμα</u>

Να διαβάσετε δύο αριθμούς και να ελέγξετε αν είναι και οι δύο μέσα στο διάστημα από 0..4. Να εκτυπώνονται αντίστοιχα μηνύματα.

```
Αρχή Αλγόριθμου
Διάβασε Α, Β
Αν Α εντός ορίων Τότε
        Αν Β εντός ορίων Τότε
                Εκτύπωσε "Ο Α και ο Β εντός ορίων"
        Διαφορετικά
                Εκτύπωσε "Ο Α εντός ορίων ο Β εκτός ορίων"
        Τέλος Αν
Διαφορετικα
        Αν Β εντός ορίων Τότε
                Εκτύπωσε "Ο Α είναι εκτός και ο Β εντός των ορίων"
        Διαφορετικά
                Εκτύπωσε "Ο Α και ο Β είναι εκτός των ορίων"
        Τέλος Αν
Τέλος Αν
Τέλος Αλγόριθμου
```

```
int main(int argc, char *argv[]) {
    int A, B;
    printf("Type two integers : ");
    scanf("%d%d",&A,&B);
    if ((A > 0) && (A < 4))  {
        if((B > 0) && (B < 4))   {
           printf("both numbers belong to range (0..4)\n");
        }
        else      {
           printf("A belons to range (0..4) but not B\n");
        }
    }
    else    {
        if((B > 0) && (B < 4))   {
                printf("B belons to range (0..4) but not A\n");
        }
        else {
           printf("Neither number belong to range (0..4)\n");
        }
    }    return 0;
}
```

Αν για οποιοδήποτε λόγο θέλουμε σε κάποιο σημείο αργότερα να επαναλάβουμε τον έλεγχο για τις ίδιες ή για δύο άλλες μεταβλητές θα πρέπει να ξαναγράψουμε τον ίδιο κώδικα.

Αν σε περίπτωση με την ολοκλήρωση του προγράμματος ανακαλύψουμε ότι υπάρχει ένα σφάλμα στο κώδικα τότε θα πρέπει να ανατρέξουμε σε κάθε σημείο του κώδικα που κάνουμε έλεγχο των ορίων και να διορθώσουμε το σφάλμα.

<u>Τι θα πρέπει να κάνουμε;</u>

Θα πρέπει να δημιουργήσουμε μια συνάρτηση η οποία θα κάνει την εργασία που πρέπει να επαναλάβω αρκετές φορές μέσα στο πρόγραμμα. Κατά αυτό το τρόπο γράφουμε λιγότερες εντολές αφού σε κάθε περίπτωση χρησιμοποιούμε το όνομα της συνάρτησης.

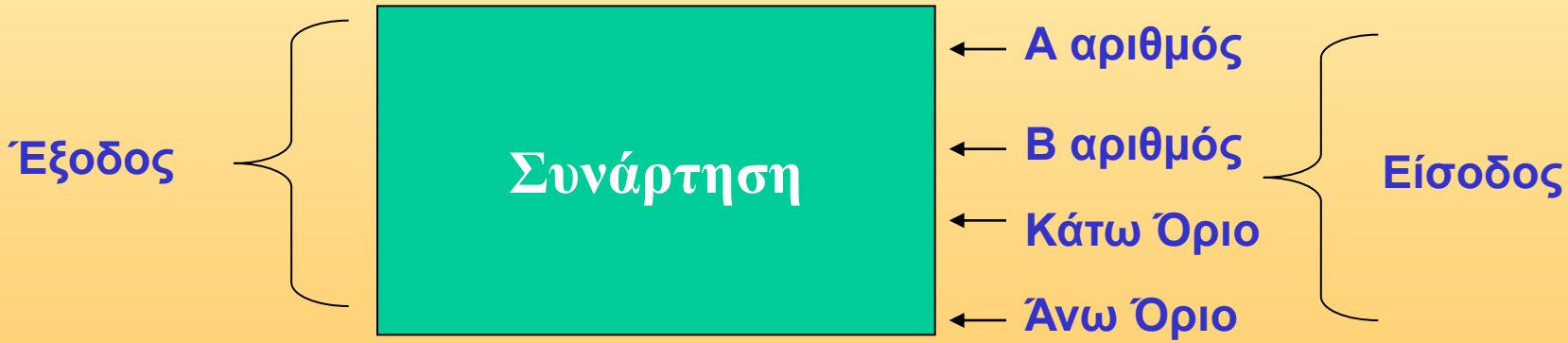
Σε περίπτωση σφάλματος θα αλλάξουμε το κώδικα μόνο μέσα στο σώμα της συνάρτησης.

Όταν κατασκευάζουμε μια συνάρτηση θα πρέπει να την θεωρούμε σε ένα κουτί που δέχεται μια ή περισσότερες εισόδους και παράγει μια έξοδο. Οι είσοδοι είναι οι αναγκαίες μεταβλητές ή ορίσματα που θα πρέπει να γνωρίζει η συνάρτηση ώστε να παράγει αποτέλεσμα.

Στην περίπτωση που δημιουργούμε μια συνάρτηση για το ανωτέρω πρόβλημα σαν εισόδους θα πρέπει να έχουμε

Τους δύο αριθμούς που θα ελέγξουμε αν ανήκουν μέσα στα όρια

Τους δύο αριθμούς που καθορίζουν το διάστημα ελέγχου.



Αντίθετα από την είσοδο όπου μπορούμε να έχουμε απεριόριστο πλήθος στην έξοδο έχουμε μόνο μια απάντηση, η οποία είναι το αποτέλεσμα του υπολογισμού της συνάρτησης.

Η συνάρτηση που κατασκευάζουμε θα πρέπει να μας ενημερώνει για τις σχέσεις των δύο αριθμών σε σχέση με το διάστημα. Οπότε υπάρχουν οι εξής περιπτώσεις :

Και οι δύο αριθμοί εκτός ορίων	Περίπτωση 0
Ο πρώτος εντός ο δεύτερος εκτός	Περίπτωση 1
Ο πρώτος εκτός ο δεύτερος εντός	Περίπτωση 2
Και οι δύο αριθμοί εντός ορίων	Περίπτωση 3

Οπότε μπορούμε να αντιστοιχίσουμε ένα ακέραιο αριθμό όπου ανάλογα την περίπτωση θα επιστρέφει την αντίστοιχη τιμή.

```
int checkLimits(int A, int B, int downLimit, int upLimit)
{
   int apot1 = 0, apot2 = 0;

   apot1 =   (A > downLimit) && (A < upLimit);

   apot2 =   (B > downLimit) && (B < upLimit);

   switch (apot1 + apot2)
   {
      case  0 : return 0; break;
      case  2 : return 3; break;
      case  1 : if(apot1)
                    return 1;
                else
                    return 2;
                break;
    }
}
```

Η δημιουργία μια συνάρτησης είναι θέμα εφαρμογής. Η τιμή που θα επιστρέψει μια συνάρτηση, το πλήθος και το είδος των ορισμάτων εξαρτάται από το πρόβλημα που επιλύουμε την κάθε στιγμή.

Εύρεση του παραγοντικού ενός αριθμού

Το παραγοντικό ενός αριθμού ορίζεται σαν το γινόμενο όλων των αριθμών από το 1 μέχρι και το αριθμό. Παράδειγμα το παραγοντικό του 5 είναι 1 * 2 * 3 * 4 * 5

Πριν προχωρήσουμε στην δημιουργία μιας συνάρτησης θα πρέπει να σκεφτούμε τι θα δέχεται σαν είσοδο και τι θα επιστρέφει σαν αποτέλεσμα.

Στο συγκεκριμένο πρόβλημα σαν είσοδο χρειαζόμαστε τον αριθμό που θα υπολογίσουμε το παραγοντικό του και σαν έξοδο θα έχουμε το αποτέλεσμα του υπολογισμού.

```
int paragontiko(int number)
{
   int i, apot;

   apot = 1;

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

Η υλοποίηση που θα κάνουμε θα πρέπει να λαμβάνει υπόψη της, τις ιδιαιτερότητες που έχει το πρόβλημα ώστε να επιτυγχάνουμε μεγαλύτερες ταχύτητες εκτέλεσης.

Για παράδειγμα στην συγκεκριμένη περίπτωση θα μπορούσαμε να ξεκινούμε την επανάληψη από το 2 δηλαδή

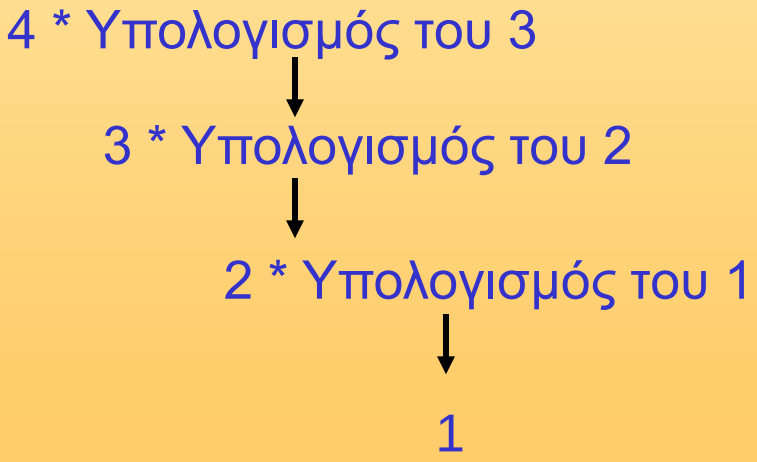
```
for(i = 2; i <= number; i++)
```

Μέσα στο σώμα μιας συνάρτησης μπορούμε να καλέσουμε άλλες συναρτήσεις ή και την ίδια την συνάρτηση. Σε αυτή την περίπτωση η συνάρτηση ονομάζεται **αναδρομική**.

```
int RecursiveParagontiko(int number)
{
   if(number == 1)
     return 1;
   else
     return number * RecursiveParagontiko(number - 1);
}
```

Αν σε κάποιο σημείο του προγράμματος καλέσουμε τη συνάρτηση με όρισμό το 4, δηλαδή printf("%d", RecursiveParagontiko(4)); Τότε οι πράξεις που θα γίνουν είναι οι παρακάτω :

Υπολογισμός του 4



Να δημιουργήσετε μια συνάρτηση με το όνομα CalcSum, η οποία θα δέχεται σαν ορίσματα 1 ακέραιο αριθμό, έστω a, και θα υπολογίζει το άθροισμα όλων των αριθμών από το 1 μέχρι και το a.

```
int CalcSum(int number)
{
  int I, sum = 0;
  for(i = 1; i <= number; i++)
      sum = sum + i;
  return(sum);
}
```

```
int CalcSum(int number)
{
   if(number <= 1)
     return number;
   else
     return number + CalcSum(number - 1);
}
```