Προγραμματισμός σε C - Συναρτήσεις

Ιωάννης Γ. Τσούλος

Τμήμα Πληροφορικής και τηλεπικοινωνιών
Πανεπιστήμιο Ιωαννίνων

2023

Περίληψη

Δηλώσεις

- Η δήλωση μιας συνάρτησης χωρίς ορίσματα ακολουθεί το σχήμα

```
type name()
{
   declarations;
   body;
   return value;
}
```

- type: ο τύπος επιστροφής
- name: το όνομα της συνάρτησης
- declarations: πιθανές τοπικές δηλώσεις μεταβλητών
- body: ο κυριώς κώδικας της συνάρτησης
- return: επιστροφή τιμής (μπορεί και να μην υπάρχει)

Συναρτήσεις void

- Αν μια συνάρτηση δεν χρειάζεται να επιστρέψει τιμή, τότε δηλώνεται σαν void.
- Σε αυτές τις συναρτήσεις δεν απαιτείται να υπάρχει δήλωση return.
- Αν εμφανίζεται κάπου μια δήλωση return, τότε διακόπτεται η συνάρτηση.

Παράδειγμα συνάρτησης void

- Συνάρτηση που εκτυπώνει τον αριθμό 0 5 φορές
- ```
  void printZero()
  {
     int i;
     for(i=0;i<5;i++)
        printf("0");
  }
  ```
- Η μεταβλητή i είναι τοπική μεταβλητή στην συνάρτηση.
  ```

Κλήση συνάρτησης void

- Για την κλήση συνάρτησης void απαιτείται απλά το όνομά της.
- Παράδειγμα στην main: printOne().
- Με την κλήση ο έλεγχος του κώδικα μεταφέρεται στην συνάρτηση και μετά τον τερματισμό της επιστρέφει στην καλούσα συνάρτηση.

Ολοκληρωμένο παράδειγμα κλήσης συνάρτησης void

```
# include <stdio.h>
void printZero()
{
        int i;
        for(i=0;i<5;i++)
                printf("0");
}
int main()
{
        printf("First call\n");
        printZero();
        printf("\nSecond call\n");
        printZero();
        return 0;
}
```

Εκτέλεση προγράμματος

- Πρώτα θα εκτυπωθεί η φράση First Call.
- Στην συνέχεια θα κληθεί για πρώτη φορά η συνάρτηση printZero() και θα εκτυπώσει 5 φορές το μηδέν.
- Στην συνέχεια θα εκτυπωθεί το μήνυμα Second Call
- Στο τέλος θα κληθεί για δεύτερη φορά η συνάρτηση printZero() και θα εκτυπώσει πάλι 5 φορές το μηδέν.

Επιστροφή τιμής

- Μια συνάρτηση μπορεί να επιστρέψει τιμή με την εντολή return.
- Μετά την εντολή return ακολουθεί η τιμή επιστροφής.
- Αν η συνάρτηση έχει δηλωθεί σαν void, τότε η return δεν επιστρέφει τιμή και δεν είναι υποχρεωτική η παρουσία της.

Συνάρτηση ανάγνωσης ακέραιων τιμών.

```
# include <stdio.h>
int readInt()
{
        int x;
        printf("Enter int\n");
        scanf("%d",&x);
        return x;
}
int main()
{
        int value = readInt();
        printf("Value = %d\n",value);
        return 0;
}
```

Η συνάρτηση readInt()

- Η μεταβλητή x θεωρείται τοπική μεταβλητή στην συνάρτηση readInt() και δεν είναι ορατή έξω από αυτήν.
- Η συνάρτηση με την επιστροφή της τιμής x τερματίζει την ζωή της.
- Η main() καλεί την readInt() γράφοντας το όνομά της.
- Η main() παραδίδει τον έλεγχο στην readInt() και όταν επιστραφεί η τιμή x, τότε η main() παίρνει και πάλι τον έλεγχο.

Η εντολή return σε λάθος σημείο

- ```
  int readInt()
  {
          int x;
          printf("Enter int\n");
          return x;
          scanf("%d",&x);
  }
  ```
- Η συνάρτηση είναι συντακτικά σωστή
- Η συνάρτηση θα επιστρέφει πάντα ότι έχει μέσα η μεταβλητή x (συνήθως 0), χωρίς ποτέ να διαβάσει κάποια τιμή.
- Αυτό ονομάζεται λογικό λάθος.
  ```

Δήλωση ορισμάτων

1. Με την χρήση ορισμάτων οι συναρτήσεις γίνονται πιο ισχυρές, καθώς ο προγραμματιστής μπορεί να παραμετροποιεί την συμπεριφορά τους.
2. Τα ορίσματα δίνουν πληροφορία από τον έξω κόσμο στην συνάρτηση.
3. Τα ορίσματα θεωρούνται τοπικές μεταβλητές σε μια συνάρτηση.

Υπολογισμός αθροίσματος τιμών

```
int sum(int low,int upper)
{
        int s=0.0;
        int i;
        for(i=low;i<=upper;i++)
        {
                s=s+i;
        }
        return s;
}
```

Κλήση συνάρτησης υπολογισμού αθροισμάτων

1. Η συνάρτηση υπολογίζει το άθροισμα των τιμών από low μέχρι και upper.
2. Αν low>upper, προφανώς θα υπολογίσει σαν άθροισμα το 0.
3. Για την κλήση μπορεί να δωθεί η έκφραση: int value=sum(2,5); Σε αυτήν την περίπτωση θα αποθηκευτεί στο value η τιμή 14.

Κλήση συνάρτησης από συνάρτηση

```
# include <stdio.h>
int readInt()
{
        int x;
        scanf("%d",&x);
        return x;
}
void printInt()
{
        printf("Value: %d\n",readInt());
}
int main(){
        printInt();
        return 0;
}
```

Κλήση συνάρτησης από συνάρτηση

- Η συνάρτηση main() θα καλέσει την printInt().
- Η συνάρτηση printInt() θα καλέσει την readInt().
- Μόλις η συνάρτηση readInt() διαβάσει έναν αριθμό ο έλεγχος επιστρέφει στην printInt().
- Η συνάρτηση printInt() θα εκτυπώσει την τιμή επιστροφής και ο έλεγχος θα επιστρέψει στην main().
- Σε αυτήν την περίπτωση θα λέγαμε πως οι κλήσεις των συναρτήσεων γίνονται σε μορφή στοίβας.

Λανθασμένη σειρά συναρτήσεων

```
# include <stdio.h>
void printInt()
{
        printf("Value: %d\n",readInt());
}
int readInt()
{
        int x;
        scanf("%d",&x);
        return x;
}
int main(){
        printInt();
        return 0;
}
```

Δήλωση συναρτησεων

- Οι συναρτήσεις θα πρέπει να υλοποιούνται πριν την χρήση τους.
- Αν αυτό δεν είναι εφικτό, τότε πρέπει να δηλώνονται πριν την χρήση τους.
- Παράδειγμα δήλωσης: void printInt(); μέσα στον κώδικα της readInt().
- Με αυτόν τον τρόπο η readInt() θα είναι ενήμερη για την ύπαρξη της συνάρτησης printInt().

Χρήση εξωτερικών μεταβλητών

```
# include <stdio.h>
int val=1;
void f1(){
        val++;
}
void f2(){
        val--;
}
int main(){
        int i;
        for(i=0;i<5;i++){
                f1();f2();f1();
        }
        printf("final val = %d \n",val);
        return 0;}
```

Χρήση καθολικών μεταβλητών

- Με τις καθολικές μεταβλητές διευκολύνεται η επικοινωνία μεταξύ συναρτήσεων.
- Ωστόσο, ο κώδικας είναι πιο δυσνόητος
- Πολλές φορές ο προγραμματιστής πρέπει να είναι προσεκτικός μήπως και μια καθολική μεταβλητή αλλάξει από λάθος.

Στατικές μεταβλητές

```
# include <stdio.h>
int f(int x)
{
        static int val=0;
        val++;
        printf("val = %d \n",val);
        return x+val;
}
int main()
{
        printf("F(2)=%d\n",f(2));
        printf("F(3)=%d\n",f(3));
        return 0;
}
```

Χρήση στατικών μεταβλητών

- Οι στατικές μεταβλητές αρχικοποιούνται μόνο μια φορά.
- Δεν εξαφανίζονται από την μνήμη όταν τερματιστεί μια συνάρτηση.

Ρώσικη διαίρεση με συνάρτηση

- Εύρεση πηλίκου με διαδοχικές αφαιρέσεις.
- ```
  int divide(int M, int N)
  {
          int count=0;
          while(M >= N)
          {
                  M=M-N;
                  count++;
          }
          return count;
  }
  ```
  ```

Ορισμός

- Οταν μια συνάρτηση καλεί τον εαυτό της, τότε είναι αναδρομική.
- Με την αναδρομή ο κώδικας είναι πιο κατανοητός.
- Δεν χρειάζεται στις περισσότερες φορές να γίνει επανάληψη.
- Πολλές φορές αν δεν γίνει προσεκτικά ο κώδικας δύναται να μην τερματίσει.

Παραγοντικό (μη αναδρομική συνάρτηση)

```
int fact(int x)
{
        int p=1;
        int i;
        for(i=1;i<=x;i++)
                p=p*i;
        return p;
}
```

Ορισμός παραγοντικού

- $N! = \begin{cases} N \times (N-1)! & N > 1 \\ 1 & \text{otherwise} \end{cases}$

Αναδρομικό παραγοντικό

```
int fact(int x)
{
        if(x>1) return x*fact(x-1);
        else return 1;
}
```

Σύνοψη

- Συναρτήσεις χωρίς ορίσματα.
- Συναρτήσεις με ορίσματα.
- Αναδρομή.