Συναρτήσεις

Εισαγωγή

Η χρήση συναρτήσεων στα προγράμματα της γλώσσας C είναι πολύ σημαντική καθώς μας επιτρέπει τη διάσπαση ενός προβλήματος σε μικρότερα υποπροβλήματα τα οποία μπορούμε να επιλύσουμε πιο εύκολα. Παράλληλα, η διάσπαση αυτή, μας επιτρέπει την επαναχρησιμοποίηση κάθε συνάρτησης, κάθε φορά που εμείς θέλουμε να εκτελεστεί μια συγκεκριμένη εργασία.

Γενικά, κάθε συνάρτηση αναλαμβάνει να διεκπεραιώσει μια εργασία. Ακολούθως, ο συντονισμός και συνεργασία των συναρτήσεων μεταξύ τους αναλαμβάνει την επίλυση ενός προβλήματος. Θεωρήστε το ακόλουθο πρόβλημα:

Ζητείται ο χρήστης να καταχωρεί πέντε ακέραιους αριθμούς και να υπολογίζεται και τυπώνεται το άθροισμα τους, ο μέσος όρος τους, ο μεγαλύτερος και ο μικρότερος.

Στο πρόβλημα αυτό διακρίνουμε πέντε διαφορετικές συναρτήσεις οι οποίες κάνουν ανεξάρτητες εργασίες που θα χρειαστούν όμως για την λύση του προβλήματος. Η πρώτη συνάρτηση θα δέχεται ακέραιο αριθμό από το χρήστη, η δεύτερη θα υπολογίζει το άθροισμα, η τρίτη το μέσο όρο κλπ. Με αυτό το τρόπο επικεντρωνόμαστε στην επίλυση ενός πιο συγκεκριμένου προβλήματος, όπως το άθροισμα ακεραίων αριθμών.

Είναι σημαντικό, κάθε συνάρτηση να είναι όσο το δυνατό ανεξάρτητη από τις υπόλοιπες, όσον αφορά την εργασία που εκτελούν. Αυτό μας εξασφαλίζει ότι πιο εύκολα θα μπορέσουμε να την επαναχρησιμοποιήσουμε. Στο πιο πάνω παράδειγμα, θα μπορούσαμε να δημιουργήσουμε μια συνάρτηση που να υπολογίζει και τυπώνει όλες τις εργασίες (άθροισμα, μέσος όρος, μεγαλύτερος, μικρότερος). Αν όμως για κάποιο λόγο χρειαστεί ο υπολογισμός μόνο του αθροίσματος, δε θα είναι δυνατή η επαναχρησιμοποίηση της συνάρτησης αυτής γιατί θα υπολογίσει και τυπώσει αχρείαστες πληροφορίες.

Ο βαθμός διάσπασης ενός προβλήματος σε υποπροβλήματα εξαρτάται από τη φύση του προβλήματος, αλλά και από τον ίδιο τον προγραμματιστή. Μεγάλη διάσπαση μπορεί να κάνει το πρόβλημα πιο σύνθετο από ότι στην πραγματικότητα είναι, μικρή διάσπαση δημιουργεί προβλήματα επαναχρησιμοποίησης. Στο πιο πάνω παράδειγμα, αν το πρόβλημα κάνει συχνή χρήση ατομικών υπολογισμών (άθροισμα, μέσος όρος κλπ) τότε η δημιουργία ξεχωριστών συναρτήσεων προτιμάται. Αντίθετα, αν το πρόβλημα συχνά χρειάζεται να κάνει όλους τους υπολογισμούς κάθε φορά, τότε μια συνάρτηση με όλους τους υπολογισμούς είναι ικανοποιητική.

Ορισμός

Μια συνάρτηση έχει την εξής μορφή:

```
data_type function_name(arguments)
{
        Ορισμός μεταβλητών

    Εντολή 1η
    Εντολή 2η
    .
    .
    .
}
```

όπου

data_type	Function_name	Arguments
void	Οποιαδήποτε συμβολοσειρά που να αρχίζει με γράμμα ή underscore (_)	Μεταβλητές εισόδου ή εξόδου.
int		
float		
double		
Κλπ		

Το data_type συμβολίζει το τύπο δεδομένου που η συνάρτηση επιστρέφει. Αν δηλαδή μια συνάρτηση έχει σαν τύπο δεδομένου το int, σημαίνει ότι η συνάρτηση, μετά την εκτέλεση των εντολών της πρέπει να επιστρέψει ένα ακέραιο αριθμό. Αν ο τύπος δεδομένων είναι void τότε η συνάρτηση δεν επιστρέφει τίποτα.

Arguments είναι μεταβλητές που παριστάνουν δεδομένα εισόδου στη συνάρτηση ή και δεδομένα εξόδου. Η χρήση τους θα φανεί πιο κάτω.

Πρωτότυπα συναρτήσεων

Όπως στις μεταβλητές, έτσι και στις συναρτήσεις, για να είναι δυνατή η χρήση τους, πρέπει πρώτα να ορισθούν. Αυτό επιτυγχάνεται με τα πρωτότυπα συναρτήσεων τα οποία τοποθετούνται πάνω από την συνάρτηση main. Ένα πρωτότυπο δε διαφέρει καθόλου από τον ορισμό μιας συνάρτησης. Η διαφορά είναι ότι τελειώνει με ';' και δεν ακολουθείται με '{ }', αφού θέλουμε μόνο να την ορίσουμε και όχι να την υλοποιήσουμε.

```
data_type function_name(arguments);
```

Είναι ευνόητο ότι ο τύπος επιστροφής της συνάρτησης, το όνομα της και οι παράμετροι της να είναι τα ίδια και στην υλοποίηση της.

Άσκηση

Να υλοποιηθεί το πρόβλημα που περιγράφηκε πιο πάνω. Υλοποιήστε πέντε διαφορετικές συναρτήσεις που να υπολογίζουν το άθροισμα, μέσο όρο, το μικρότερο και μεγαλύτερο από πέντε ακέραιους μη αρνητικούς αριθμούς που θα δίνει ο χρήστης. Κάθε συνάρτηση θα έχει σαν παραμέτρους εισόδου, τους πέντε αριθμούς που έδωσε ο χρήστης και θα επιστρέφει η καθεμιά το αποτέλεσμα που υπολογίζει.

Λύση (Για τις συναρτήσεις αθροίσματος και μέσου όρου)

```
#include <stdio.h>



/*Prototypes*/

int User_Number(void);

float Number_Sum(int a, int b, int c, int d, int e);

float Number_Average(int num1, int num2, int num3, int num4, int num5);




void main(void)
{
```

```
        int num1, num2, num3, num4, num5;
        float sum;
        float average;

    /*Call function to get numbers from the user*/
    num1 = User_Number();
    num2 = User_Number();
    num3 = User_Number();
    num4 = User_Number();
    num5 = User_Number();

    /*Call function to compute the sum*/
    sum = Number_Sum(num1, num2, num3, num4, num5);
    printf("\nThe sum is %.2f.", sum);

  /*Call function to compute the average*/
    average = Number_Average(num1, num2, num3, num4, num5);
    printf("\nThe average is %.2f.", average);

}
```

```
int User_Number()
{
		int number;


	printf("Please, enter a number:");
	scanf("%d", &number);


	while (number < 0)
	{
		printf("You have entered a negative number. Please, enter a positive number:");
		scanf("%d", &number);
	}


   return number;


}




float Number_Sum(int a, int b, int c, int d, int e)
{
		float result;


	result = a + b + c + d + e;


   return result;


}



float Number_Average(int num1, int num2, int num3, int num4, int num5)
{
		float avg;
```

```
        /*Call function to compute the sum*/
        avg = Number_Sum(num1, num2, num3, num4, num5) / 5;

    return avg;


}
```

Να υλοποιηθούν και οι συναρτήσεις για εύρεση του μικρότερου και μεγαλύτερου αριθμού.

Γενικές σημειώσεις

- Η επιστροφή τιμών από μια συνάρτηση, επιτυγχάνεται με τη χρήση της εντολής return.
- Η κλήση μιας συνάρτησης, επιτυγχάνεται γράφοντας το όνομα της καθώς και τις παραμέτρους της. Αν δεν έχει παραμέτρους, γράφονται δύο κενές παρενθέσεις '()'.
- Αν μια συνάρτηση επιστρέφει κάποια τιμή, πρέπει η τιμή αυτή να ανατεθεί για παράδειγμα σε μια άλλη μεταβλητή.