Εισαγωγή στον Προγραμματισμό

Πανεπιστήμιο Θεσσαλίας

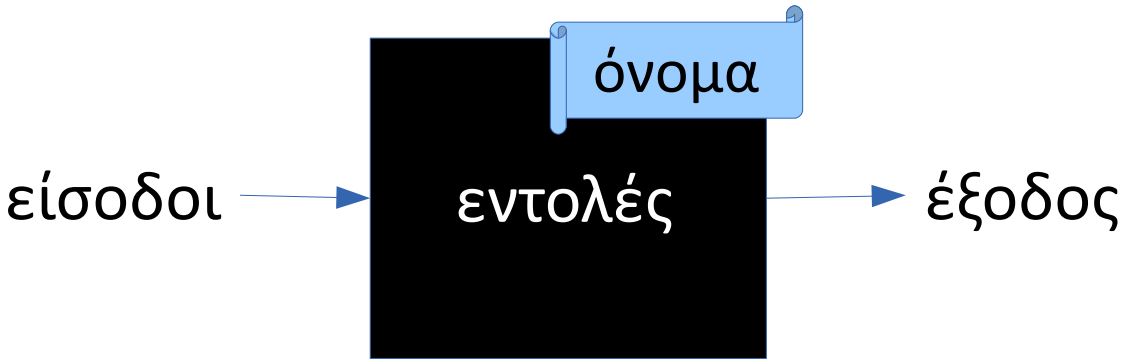
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Η/Υ

Συναρτήσεις

- Προβλήματα:
  - Οσο μεγαλώνουν τα προγράμματα, γίνονται πιο πολύπλοκα.
  - Πολλές λειτουργίες σε ένα πρόγραμμα επαναλαμβάνονται σε διάφορα σημεία.
- Λύση:
  - Διάσπαση προγράμματος σε μικρότερα, καλά ορισμένα κομμάτια.

Συναρτήσεις

- Συνάρτηση = ένα "μαύρο κουτί" που
  - έχει όνομα
  - δέχεται ένα αριθμό από εισόδους
  - παράγει μια έξοδο
  - εκτελεί μια καλά ορισμένη λειτουργία
    - μέσω μιας σειράς εντολών που παράγουν την έξοδο από τις εισόδους



Συναρτήσεις

- Δε χρειάζεται να γνωρίζουμε τι περιέχει το "μαύρο κουτί" για να χρησιμοποιήσουμε μια συνάρτηση.
  - π.χ. χρησιμοποιούμε την printf χωρίς να γνωρίζουμε πώς ακριβώς εκτυπώνει τα δεδομένα στην οθόνη.
- Αρκεί να γνωρίζουμε το όνομά της, τι εισόδους χρειάζεται και τι είδους έξοδο θα παράγει.
  - Αυτές οι πληροφορίες αναφέρονται στη δήλωση μιας συνάρτησης.
  - Το "μαύρο κουτί" αποτελεί τον ορισμό της συνάρτησης.

Συναρτήσεις

- Οι δηλώσεις των συναρτήσεων βιβλιοθήκης της C βρίσκονται σε ειδικά αρχεία με κατάληξη .h.
  - πχ. stdio.h για συναρτήσεις εισόδου/εξόδου,
  - math.h για μαθηματικές συναρτήσεις
  - string.h για συναρτήσεις που δρουν σε συμβολοσειρές
  - κτλ.
- Χρησιμοποιούμε #include<αρχείο.h> για να συμπεριλάβουμε το αρχείο στο πρόγραμμά μας.
- Τα < > σημαίνουν ότι το εν λόγω αρχείο βρίσκεται στο φάκελο των .h αρχείων της βιβλιοθήκης.

Συναρτήσεις

- Μπορούμε κι εμείς να φτιάξουμε .h αρχείο στο οποίο θα βάλουμε δηλώσεις δικών μας συναρτήσεων.
- Σε αυτή την περίπτωση χρησιμοποιούμε #include"αρχείο.h" για να συμπεριλάβουμε το αρχείο στο πρόγραμμά μας.
- Τα " " σημαίνουν ότι το εν λόγω αρχείο βρίσκεται στον ίδιο φάκελο όπου είναι και το πρόγραμμά μας (το .c αρχείο).
- Εναλλακτικά, βάζουμε τις δηλώσεις των δικών μας συναρτήσεων μέσα στο .c αρχείο.

Πώς γράφουμε μια δική μας συνάρτηση

- Δηλώνουμε τη συνάρτηση, προσδιορίζοντας
  - όνομα
  - τον τύπο της τιμής που επιστρέφει
  - μια λίστα των ποσοτήτων που χρειάζεται ως εισόδους και τους τύπους τους (λίστα παραμέτρων)
  - Οι παραπάνω πληροφορίες αποτελούν το prototype της συνάρτησης.
- Ορίζουμε τη συνάρτηση:
  - Γράφουμε το block εντολών που αποτελούν το σώμα της συνάρτησης και που εκτελούνται όταν χρησιμοποιούμε (καλούμε) τη συνάρτηση.

Prototype συνάρτησης

- Η σύνταξη του prototype μιας συνάρτησης είναι:

  τύπος_επιστροφής όνομα (λίστα παραμέτρων) ;

- Αν η συνάρτηση δεν επιστρέφει κάτι, χρησιμοποιούμε void ως τύπο επιστροφής.
- Οι κανόνες για το όνομα είναι ίδιοι με αυτούς για την ονομασία μεταβλητών. Συνήθως επιλέγουμε ρήματα ή ρηματικές φράσεις, διότι οι συναρτήσεις περιγράφουν πράξεις/ενέργειες.
- Η λίστα παραμέτρων μοιάζει με μια σειρά δηλώσεων μεταβλητών, με κόμματα ανάμεσα.
- Τα prototypes τοποθετούνται πριν τη main.

Ορισμός συνάρτησης

- Η σύνταξη του ορισμού μιας συνάρτησης είναι:

```
τύπος_επιστροφής όνομα (λίστα παραμέτρων)   {
        εντολές ;
}
```

- Η πρώτη γραμμή είναι ίδια με το prototype, χωρίς το ;
- Στα άγκιστρα περικλείεται το σώμα της συνάρτησης
- Αν ο τύπος επιστροφής δεν είναι void, η συνάρτηση πρέπει να περιέχει τουλάχιστον μία εντολή return η οποία επιστρέφει μία τιμή τύπου τύπος_επιστροφής
- Ο ορισμός τοποθετείται μετά τη main.

Παράδειγμα

- Γράψτε ένα πρόγραμμα που διαβάζει 2 ακεραίους και βρίσκει κι εκτυπώνει τον μεγαλύτερο.
  - χωρίς συνάρτηση
  - με συνάρτηση η οποία δεδομένων 2 ακεραίων, επιστρέφει τον μεγαλύτερο.
- Παρατήρηση:
  - Μια συνάρτηση πρέπει να υλοποιεί μία, καλά ορισμένη λειτουργία.
    - Δε θέλουμε να κάνει _και_ εκτύπωση αλλά μόνο να υπολογίζει και να μας δίνει το αποτέλεσμα.

Παράδειγμα - χωρίς συνάρτηση

```
#include <stdio.h>

int main (int argc, char *argv[]) {
   int num1, num2, max;

   printf("Enter 2 integers: ");
   scanf("%d %d", &num1, &num2);
   if (num1 > num2) {
      max = num1;
   }
   else {
      max = num2;
   }
   printf("max = %d\n", max);
   return 0;
}
```

Παράδειγμα - με συνάρτηση

- Το prototype:

```
int maxOf2 (int x, int y);
```

- Ο ορισμός:

```
int maxOf2 (int x, int y) {

   int max;
   if (x > y) {
      max = x;
   }
   else {
      max = y;
   }
   return max;
}
```

Παράδειγμα με συνάρτηση

- Η συνάρτηση που κατασκευάσαμε παίρνει δύο ακέραιες παραμέτρους και επιστρέφει μια ακέραια τιμή.
- Για να την χρησιμοποιήσουμε (καλέσουμε) γράφουμε το όνομά της και μέσα σε παρενθέσεις μια τιμή για κάθε μία παράμετρο, με κόμμα ανάμεσα.
- Οταν καλούμε μια συνάρτηση που επιστρέφει κάτι, τότε η κλήση είναι μια έκφραση (δηλαδή έχει τιμή) και μπορούμε να τη χρησιμοποιήσουμε στο δεξί μέρος μιας ανάθεσης ώστε να αποθηκεύσουμε το αποτέλεσμα σε μια μεταβλητή.

Παράδειγμα - με συνάρτηση

```
#include <stdio.h>

int maxOf2 (int x, int y);

int main (int argc, char *argv[]) {
   int num1, num2, max;

   printf("Enter 2 integers: ");
   scanf("%d %d", &num1, &num2);
   max = maxOf2(num1, num2);
   printf("max = %d\n", max);
   return 0;
}
```

Το πλήρες πρόγραμμα

```
int maxOf2 (int x, int y) {

   int maximum;
   if (x > y) {
      maximum = x;
   }
   else {
      maximum = y;
   }
   return maximum;
}
```

Παράδειγμα - παραλλαγή

```
#include <stdio.h>

int maxOf2 (int x, int y);

int main (int argc, char *argv[]) {
   int num1, num2, max;

   printf("Enter 2 integers: ");
   scanf("%d %d", &num1, &num2);
   max = maxOf2(num1, num2);
   printf("max = %d\n", max);
   return 0;
}
```

Το πλήρες πρόγραμμα

```
int maxOf2 (int x, int y) {

   if (x > y) {
      return x;
   }
   else {
      return y;
   }
}
```

Παράδειγμα - πώς δουλεύει

```
#include <stdio.h>

int maxOf2 (int x, int y);

int main (int argc, char *argv[]) {
    int num1, num2, max;

    printf("Enter 2 integers: ");
    scanf("%d %d", &num1, &num2);
    max = maxOf2(num1, num2);
    printf("max = %d\n", max);
    return 0;
}
```

```
int maxOf2 (int x, int y) {

   int maximum;
   if (x > y) {
      maximum = x;
   }
   else {
      maximum = y;
   }
   return maximum;
}
```

Πριν:

Η παράμετρος x παίρνει ως τιμή ένα αντίγραφο της τιμής της μεταβλητής num1.

Η παράμετρος y παίρνει ως τιμή ένα αντίγραφο της τιμής της μεταβλητής num1.

Μετά:

Η έκφραση maxOf2(num1,num2) παίρνει ως τιμή ένα αντίγραφο της τιμής της μεταβλητής maximum.

Λίγη ορολογία

- Οι παράμετροι που εμφανίζονται στο prototype μιας συνάρτησης λέγονται τυπικές παράμετροι (formal parameters)
- Οταν καλούμε τη συνάρτηση, πρέπει να προσδιορίσουμε συγκεκριμένες τιμές για τις παραμέτρους. Αυτές οι τιμές λέγονται πραγματικές παράμετροι (actual parameters).
- Κατά την εκτέλεση της συνάρτησης, οι πραγματικές παράμετροι αντιγράφονται στις τυπικές παραμέτρους. Επειδή αντιγράφεται η τιμή των παραμέτρων, λέμε ότι γίνεται κλήση κατ'αξία (call by value)

Παράδειγμα - παραλλαγή

```
#include <stdio.h>

int maxOf2 (int x, int y);

int main (int argc, char *argv[]) {
   int num1, num2;

   printf("Enter 2 integers: ");
   scanf("%d %d", &num1, &num2);
   printf("max = %d\n", maxOf2(num1, num2));
   return 0;
}
```

```
int maxOf2 (int x, int y) {

  if (x > y) {
    return x;
  }
  else {
    return y;
  }
}
```

Τοπικές μεταβλητές

- Μεταβλητές που δηλώνονται μέσα σε μια συνάρτηση λέγονται τοπικές (local) και είναι ορατές μόνο μέσα σε αυτή τη συνάρτηση.
  - Λέμε ότι η εμβέλειά τους (scope) είναι η συνάρτηση.
- Διαφορετικές συναρτήσεις μπορούν να χρησιμοποιούν ίδια ονόματα για τις τοπικές μεταβλητές τους χωρίς να υπάρχει πρόβλημα.
- Οι παράμετροι μιας συνάρτησης είναι επίσης ορατές μόνο μέσα στη συνάρτηση.

Τοπικές μεταβλητές

```c
#include <stdio.h>

int maxOf2 (int x, int y);

int main (int argc, char *argv[]) {
   int x, y, max;

   printf("Enter 2 integers: ");
   scanf("%d %d", &x, &y);
   max = maxOf2(x, y);
   printf("max = %d\n", max);
   return 0;
}
```

```c
int maxOf2 (int x, int y) {
   int max;
   if (x > y) {
      max = x;
   }
   else {
      max = y;
   }
   return max;
}
```

Καμία σχέση ανάμεσα στις κόκκινες και μπλε μεταβλητές/παραμέτρους

Καθολικές μεταβλητές

- Μπορούμε να δηλώσουμε μεταβλητές έξω από κάποια συνάρτηση. Αυτές λέγονται καθολικές (global) και είναι ορατές σε όλο το πρόγραμμα.
- Επικίνδυνες!
  - Πιθανότητα για επικαλυπτόμενες εμβέλειες
- Κακές!
  - Χαλάει την "ανεξαρτησία" ανάμεσα σε διαφορετικές συναρτήσεις.
- Συνήθως δε χρειάζονται!

Εμβέλεια τοπικών μεταβλητών

```
#include<stdio.h>

double square(double x);

int main () {
     double num, num_squared;
     num = -4.3;
     num_squared = square(num);
     printf("%lf squared is %lf\n",num, num_squared);
     return 0;
}

double  square(double x)  {
     double num_squared;
     num_squared = x*x;
     return (num_squared);
}
```

Εμβέλεια num, num_squared

ΔΙΑΦΟΡΕΤΙΚΕΣ ΜΕΤΑΒΛΗΤΕΣ!

Εμβέλεια num_squared

Εμβέλεια καθολικών μεταβλητών

```
#include<stdio.h>

double square();
double num;

int main () {
        double num_squared;
        num = -4.3;
        num_squared = square();
        printf("%lf squared is %lf\n",num,num_squared);
        return 0;
}

double  square() {
        double result;
        result = num*num;
        return (result);
}
```

Εμβέλεια num

Επικαλύψεις εμβέλειας

```
#include<stdio.h>
int x;
void func();

int main() {
   x = 5;        /* ανάθεση καθολικής x */
   printf("Before func, x=%d\n", x);
   func();
   printf("After func, x=%d\n", x);
   return 0;
}
void func() {
   int x;        /* δήλωση τοπικής  x */
   x=10;         /* τοπική x  */
   printf("Within func, x=%d\n", x);
}
```

Εμβέλεια καθολικής x

εδώ το x είναι 5

Εμβέλεια τοπικής x

εδώ το x είναι 10

Η τοπική δήλωση του x κρύβει την καθολική δήλωση

Παράδειγμα

- Γράψτε ένα πρόγραμμα το οποίο διαβάζει 3 ακεραίους από το πληκτρολόγιο και υπολογίζει κι εκτυπώνει τον μεγαλύτερο.
  - Κατασκευάστε και χρησιμοποιήστε μια συνάρτηση maxOf3 η οποία παίρνει ως παραμέτρους 3 ακεραίους κι επιστρέφει τον μέγιστο. Η συνάρτηση πρέπει να χρησιμοποιεί την maxOf2 που γράψαμε νωρίτερα.

Παράδειγμα

- Τι θα εκτυπώσει το παρακάτω πρόγραμμα?

```
#include<stdio.h>

void swap (int x, int y);

int main (int argc, char *argv[]) {
   int num1, num2;

   printf("Enter 2 integers: ");
   scanf("%d %d", &num1, &num2);
   swap(num1, num2)
   printf("num1=%d, num2=%d\n",num1, num2);
   return 0;
}
```

```
void swap (int x, int y) {
   int temp;
   temp = x;
   x = y;
   y = temp;
}
```