ΣΕΜΙΝΑΡΙΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ

FUNCTIONS, STRINGS, STRUCTS

Επιμέλεια:
Μπεκτασιάδης Ευστράτιος

Functions (Συναρτήσεις)

- Χρησιμοποιούντε προκειμένου να μειώσουμε όσο το δυνατόν περισσότερο τον επαναλαμβανόμενο κώδικα.
- Βοηθούν στο να είναι πιο κατανοητός ο κώδικας ως προς εμάς αλλά και ως προς τρίτους.
- Η συντήρηση της οποιαδήποτε εφαρμογής γίνεται αρκετά πιο απλή.
- Η συναρτήσεις μπορούν να επιστρέψουν κάποια τιμή κατά την κλήση της.
- Μπορούμε να φτιάξουμε μία «βιβλιοθήκη» την οποία να χρησιμοποιούμε σε διάφορες εφαρμογές μας. (Συνήθως αυτές οι συναρτήσεις είναι πολύ γενικές, για αυτό και αποτελεί σπάνιο φαινόμενο κάτι τέτοιο.)

Δομή των συναρτήσεων

```
[Τύπος επισ. Μεταβλητής] [Όνομα Συνάρτησης] ([Είσοδος]) {
        //Κομμάτι κώδικα
        return [Έξοδος];
}
```

Κλήση συνάρτησης

- Έστω ότι έχουμε μία συνάρτηση που βάζουμε 2 αριθμούς σαν είσοδο και μας εκτυπώνει τον μεγαλύτερο. Η συνάρτηση αυτή έστω ότι λέγεται: **max**.

```
int a = 4, b = 5;
max(a, b);
```

- Αυτό θα μας εκτυπώσει τον αριθμό 5. (Υπάρχει **printf** μέσα στην συνάρτηση **max**)

Παράδειγμα: Συνάρτηση αθροίσματος 2 αριθμών

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

void add_two_nums(int a, int b){
  int sum = a + b;
  printf("The result is: %d \n", sum);
  return;
}

int main(int argc, char* argv[]){
  add_two_nums(6,5);
  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
The result is: 11
```

Επιστροφή τιμής από συνάρτηση

- Όπως είπαμε και στην αρχή μία συνάρτηση μπορεί να επιστρέφει κάποια τιμή (σαν αποτέλεσμα της κλίσης της).
- Ο τύπος αυτής της μεταβλητής καθορίζεται από το τί έχουμε ορίσει ως τύπο **κατά την δημιουργία** της συνάρτησης.
- Για να επιστρέψουμε κάποια τιμή στο σημείο από το οποίο κλήθηκε η συνάρτηση χρησιμοποιούμε το **return**. Με το **return** η συνάρτηση σταματάει την εκτέλεση της. (Αυτό σημαίνει πως δεν μπορούν να γίνουν 2 **return** ταυτόχρονα με μία κλήση συνάρτησης.)
- Μπορεί να χρησιμοποιηθεί και ως ένδειξη για το αν όλα πήγαν καλά κατά την διαδικασία εκτέλεσης της συνάρτησης ή όχι. (Με την χρήση 1 και 0)
- Η επιστροφή τιμής **δεν είναι υποχρεωτική**. Αν δεν θέλουμε η συνάρτηση μας να επιστρέφει κάτι σαν αποτέλεσμα την δηλώνουμε ως **void**.

Παράδειγμα: Συνάρτηση πολλ/σμου 2 αριθμών

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

int mul_two_nums (int a, int b) {
  printf("I am on function \n");
  int mul = a * b;
  return mul;
}

int main(int argc, char* argv[]){
  int result;
  printf("I am on main \n");
  result = mul_two_nums(5, 5);
  printf("The result is %d \n", result);
  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
I am on main
I am on function
The result is 25
```

Αναδρομικές συναρτήσεις

- Υπάρχει περίπτωση να θέλουμε μέσα σε μία συνάρτηση να καλέσουμε την ίδια συνάρτηση (πιθανώς με άλλη είσοδο).
- Αυτό δημιουργεί μία «ουρά» συναρτήσεων που πρέπει να εκτελεστούν. Συνήθως σε τέτοιες περιπτώσεις χρησιμοποιούμε μία συνθήκη με την οποία και σταματάμε την αναδρομική αυτή διαδικασία.

Παράδειγμα: Εκτύπωση αριθμών από 1 εως 10

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

void printNumbers(int num) {
  printf("%d \n", num);
  if(num+1 < 10)
    printNumbers(num+1);
  return;
}

int main(int argc, char* argv[]){
  printNumbers(0);
  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
0
1
2
3
4
5
6
7
8
9
```

Κάτι Περίεργο...

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

void printNumbers(int num) {
  if(num+1 < 10)
    printNumbers(num+1);
  printf("%d \n", num);
  return;
}

int main(int argc, char* argv[]){
  printNumbers(0);
  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
9
8
7
6
5
4
3
2
1
0
```

Strings (Συμβολοακολουθίες)

- Ένα **String** στην ουσία αποτελεί έναν «πίνακα» χαρακτήρων όπου στη κάθε θέση του περίεχει και ένα γράμμα.
- Στο τέλος αυτού του πίνακα υπάρχει το **\0** . Αυτό δηλώνει το τέλος του **String** και χρησιμοποιείτε από τον υπολογιστή προκειμένου να καταλάβει πότε πρέπει να σταματήσει την ανάγνωση.
- Υπάρχει η βιβλιοθήκη **string.h** η οποία περίεχει όλες τις πιθανές συναρτήσεις που πρόκειται να χρησιμοποιήσετε.
- **Print format** ενός **String**: **%s**

Μερικές χρήσιμες συναρτήσεις που περιέχει η βιβλιοθήκη

- strcpy(String1, String2): Αντιγράφει το **String2** στην μεταβλήτη **String1**.
- strcat(String1, String2): «Κολλάει» την λέξη του **String2** δεξιά του **String1**. Το επιστρέφει σαν ένα νέο **String**.
- strlen(String1): Επιστρέφει το μέγεθος του **String** (πόσα γράμματα περιέχει)
- strcmp(String1, String2): Ελέγχει αν τα 2 **String** είναι ίσα. Τότε επιστρέφει 0. Σε άλλη περίπτωση επιστρέφει κάτι μεγαλύτερο ή μικρότερο του 0.*
- strtok(String1, Parameter): Σπάει σε υπό-**strings** το **String1** όποτε βρίσκει μέσα στο τελευταίο το **Parameter**.

I/O & Strings

- Γενικά τα **Strings** έχουν έναν παραπάνω κίνδυνο από τις υπόλοιπες μεταβλητές και αυτό ωφείλεται στο μεταβλητό τους μέγεθος.
- Προκείμενου να εξαλείψουμε όσο το δυνατόν γίνεται περισσότερο τον κίνδυνο αυτό πρέπει να χρησιμοποιούμε συναρτήσεις με ασφαλής (περιορισμένων χαρακτήρων) I/O.

Μερικές «ασφαλής» συναρτήσεις

- fgets(char[] word, sizeof(word), stdin): Δέχεται κείμενο από το πληκτρολόγιο (**stdin**) και διαβάζει τους **sizeof(word) – 1** χαρακτήρες.* Αυτούς τους περνάει μέσα στον πίνακα **char[] word.**
- puts(char[] word): Εκτυπώνει το **string** "**word**" και στο τέλος προσθέτει ένα **\n.**

Παράδειγμα: Ανάγνωση και Εκτύπωση String

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

int main(int argc, char* argv[]){
  char name[25];

  fgets(name, sizeof(name), stdin);
  printf("Name: %s ", name);

  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
Stratos
Name: Stratos
```

Παράδειγμα: Αντιγραφή String

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

int main(int argc, char* argv[]){
  char name[25], name2[25];

  fgets(name, sizeof(name), stdin);
  printf("Name: %s", name);

  strcpy(name2, name);
  printf("Name 2: %s", name2);

  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
Stratos
Name: Stratos
Name 2: Stratos
```

Παράδειγμα με εύρεση μεγέθους String

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

int main(int argc, char* argv[]){
  int first_name_length;
  char first_name[25];

  fgets(first_name, sizeof(first_name), stdin);
  first_name_length = strlen(first_name);

  printf("First Name: %s (Total Characters: %d)",
   first_name, first_name_length);

  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
Stratos
First Name: Stratos
 (Total Characters: 8)
```

Παράγινε το κακό... Ώρα να φύγει το **\n** !

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

int main(int argc, char* argv[]){
  int first_name_length;
  char first_name[25];

  fgets(first_name, sizeof(first_name), stdin);

  first_name_length = strlen(first_name);
  first_name[first_name_length-1] = '\0';

  first_name_length = strlen(first_name);

  printf("First Name: %s (Total Characters: %d)",
   first_name, first_name_length);

  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
Stratos
First Name: Stratos (Total Characters: 7)
```

Πάραδειγμα: Σπάσιμο ενός string σε υποstring

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

int main(int argc, char* argv[]){
  char full_name[50] = "Stratos Bektasiadis";
  char full_name_backup[50];
  char target[2] = " ";
  char *token;

  strcpy(full_name_backup, full_name);
  token = strtok(full_name, target);
  while(token != NULL){
    printf("Word: %s\n", token);
    token = strtok(NULL, target);
  }

  return 0;
}
```

```
Stratos@DESKTOP-0TVLJ7H MINGW64 ~/Desktop
$ ./ex1
Word: Stratos
Word: Bektasiadis
```

Structs (Δομές)

- Ένα **struct** στην ουσία αποτελεί μία «νέου τύπου μεταβλητή» όπως π.χ **int**, **char** κ.λ.π, και μπορεί να συμπεριφερθεί με τον ίδιο ακριβώς τρόπο.
- Μέσα σε ένα **struct** εμπεριέχονται συνήθως παραπάνω από μία μεταβλητές.
- Μέσα σε ένα **struct** μπορούμε να βάλουμε ένα άλλο **struct** ίδιου ή διαφορετικού τύπου.*
- Το μέγεθος που χρειάζεται στην μνήμη το κάθε **struct** προκύπτει από το άθροισμα των επιμέρους μεταβλητών του.
- Είναι απίστευτα χρήσιμα όταν θέλουμε να περιγράψουμε κάτι πιο σύνθετο από μία μεταβλητή (π.χ έναν Άνθρωπο ή ένα βιβλίο) . Βοηθάει επίσης πολύ στην οργάνωση.

Δομή τών Structs

```
typedef struct {
        //Διάφορες μεταβλητές
} [όνομα struct];
```

Χρήση ενός Struct

- Έστω ότι έχουμε ένα **struct** το οποίο ονομάζεται coords. Μέσα σε αυτό βρίσκοντε κάποιες μεταβλητές: double latitude, double longitude. Αν θέλουμε να φτιάξουμε μία μεταβλητή τύπου coords στη **main** απλά γράφουμε: coords Sintetagmenes;
- Προκειμένου να προσπελάσουμε τις 2 μεταβλητές που εμπεριέχονται σε αυτό χρησιμοποιούμε «.» και στη συνέχεια το όνομα της μεταβλητής:

```
Sintetagmenes.latitude = 38.899899;
Sintetagmenes.longitude = 22.433680;
```

Παράδειγμα: 2 εφαρμογές με Structs

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

typedef struct{
  char first_name[50];
  char last_name[50];
  int age, afm;
} person;

int main(int argc, char* argv[]){
  person p1;
  strcpy(p1.first_name, "Stratos");
  strcpy(p1.last_name, "Bektasiadis");
  p1.age = 21;
  p1.afm = 498004812;
  return 0;
}
```

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

typedef struct{
  char title[100];
  char ISBN[50];
  int pages, version;
} book;

int main(int argc, char* argv[]){
  book b1;
  strcpy(b1.first_name, "Harry Potter and the Philosopher's Stone");
  strcpy(b1.ISBN, "0-7475-3269-9");
  b1.pages = 332;
  b1.version = 2014;
}
```

Πίνακας με Structs

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

typedef struct{
  char title[100];
  char ISBN[50];
  int pages, version;
} book;

int main(int argc, char* argv[]){
  book bookshelf[50];

  strcpy(bookshelf[0].first_name, "Harry Potter...");
  strcpy(bookshelf[0].ISBN, "0-7475-3269-9");
  bookshelf[0].pages = 332;
  bookshelf[0].version = 2014;

  strcpy(bookshelf[1].first_name, "The Giver");
  strcpy(bookshelf[1].ISBN, "0-553-57133-8");
  bookshelf[1].pages = 208;
  bookshelf[1].version = 2006;
}
```

Όπως κάθε τύπου μεταβλητή (π.χ **int**, **char** κ.λ.π) μπορούν να γίνουν πίνακας έτσι και το **struct book** που φτιάξαμε μπορεί το ίδιο! Όπως είπαμε και πριν, ο τρόπος που συμπεριφερόμαστε σε οποιαδήποτε τύπου μεταβλητή είναι ίδια με αυτή των **struct**.

Προσωπική εμπειρία με Structs...

```c
typedef struct serverStatistics{
    int totalAccounts;
    int totalSingleCharacters;
    int totalMultiCharacters;
}server_statistics_t;

typedef struct connectionProperties{
    int socketId;
    int hostPositionInStruct;

    //char charName[20];
    char username[20];
    char **currentMaze;

    GC_combat_character_t *GCCombatCharacter;
    GC_load_stats_character_t GCLoadStatsCharacter;

    multi_char_properties_t hostSlots[3];
}connection_properties_t;

typedef struct playerVariables{
    int length;
    char **hosts;
    connection_properties_t *connectionProperties;
    server_statistics_t *serverStatistics;
}player_variables_t;
```

```c
typedef struct GCLoadStatsCharacter{
    int charSlot;

    int charMaxHP;
    int charCurHP;

    int charMaxEXP;
    int charCurEXP;

    int charCurDMG;

    int charLevel;
    int charHealth;
    int charArmor;
    int charAttack;
    int charAccuracy;
    int charCurPoints;

    int charWins;
    int charLoses;

    int charMazeLevel;
    int charX;
    int charY;

    int currentCharMazeLevel;
    int currentCharX;
    int currentCharY;

    char charName[20];
}GC_load_stats_character_t;

typedef struct GCImproveStatsCharacter{
    int confirmation;
}GC_improve_stats_character_t;
```

```c
typedef struct packetData{
    int curPos;
    void* packetReceived;
    void* packetToSend;
    player_variables_t *playerVariables;
}packet_data_t;

typedef struct packetProperties{
    int   CGPacketSize;
    int   GCPacketSize;
    void (*curFunc)(packet_data_t packetData);
}packet_properties_t;
```

```c
typedef struct GCCombatCharacter{
    int damageReceivedCharacter;
    int damageReceivedMob;
    int winner;

    GC_load_stats_character_t GCLoadStatsCharacter;
    GC_load_stats_mob_t GCLoadStatsMob;
}GC_combat_character_t;
```

Τέλος Διαφανειών

Ώρα να δούμε τα πάντα στην πράξη