Εργαστήριο 1: Επανάληψη Βασικών Εννοιών στη Γλώσσα C

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:
Εισαγωγή στις έννοιες:

- Συναρτήσεις
- Εξωτερικές μεταβλητές
- Κανόνες εμβέλειας
- Διάρκεια μεταβλητών
- Αρχικοποίηση μεταβλητών

Οδηγίες για Προγραμματιστικές Ασκήσεις

- Η Εργασία θα παραδοθεί σε ηλεκτρονική μορφή μέσο email στον υπεύθυνο του εργαστηρίου (petrosp OR paul.antoniou @cs.ucy.ac.cy)
- Στο email σας πάντα να γράφεται για Θέμα/Subject τα ακόλουθα: EPL035 Homework <αριθμός άσκησης> ID:<αριθμός ταυτότητας> π.χ., EPL035 Homework 1 ID:123456
- Η εργασία να παραδίδεται και σε έντυπη μορφή η οποία θα περιέχει τον κώδικα σας και ένα αντιπροσωπευτικό παράδειγμά εκτέλεσης του μαζί με τα σχόλια που ζητούνται πιο πάνω.
- Το όνομα του αρχείου με τον πυγαίο κώδικα να είναι ο αριθμός ταυτότητας σας.
- Οι κώδικες σας θα αναλυθούν με το σύστημα moss για αντιγραφές.
- …και φυσικά θα πρέπει να γίνεται compile με τον compiler **gcc** της αίθουσας B103.

Συναρτήσεις (Επανάληψη)

```
#include <stdio.h>

int power(int m, int n);

main()
{   int i;
    for (i = 0;  i < 10; i++)
         printf("%d %d \n", i, power(2,i));
    return 0;
}
int power(int base, int n)
{   int i, p;
    p = 1;
    for (i = 1; i <= n; i++)
         p = p * base;
    return p;
}
```

Συναρτήσεις (Επανάληψη)

- Συναρτήσεις μπορούν να δηλωθούν σε οποιαδήποτε σειρά, σε ένα ή περισσότερα αρχεία.

- Βασικές Έννοιες
  - Πρωτότυπο συνάρτησης
  - Κλήση/Επιστροφή
  - Παραμέτροι (τυπικές - πραγματικές)

- **Παραμέτροι - πέρασμα δια τιμής (call by value):**

```
int power(int base, int n)
```

  - διοχέτευση πληροφοριών προς τη συνάρτηση
  - οι τιμές των πραγματικών παραμέτρων αντιγράφονται σε νέες (προσωρινές) μεταβλητές κατά την εκτέλεση της κληθείσας συνάρτησης.
  - η κληθείσα συνάρτηση δεν μπορεί να επηρεάσει άμεσα τις τιμές μεταβλητών στη συνάρτηση κλήσης.

Συναρτήσεις (Επανάληψη)

- **Πέρασμα δια αναφοράς (call by pointer or reference)**

  ```
  swap(int *a, int *b)
  ```

  - Η συνάρτηση κλήσης παρέχει στη συνάρτηση, *δείκτες* προς τις μεταβλητές (δηλαδή πάλι η διεύθυνση της μεταβλητής).
  - Η κληθείσα συνάρτηση μπορεί να αλλάξει τις τιμές των μεταβλητών.

- **Στην C++ υπάρχει και το καθαρό πέρασμα δια αναφοράς (call by reference)** `swap(int &a, int &b)`

  Σε αυτό το μάθημα θα χρησιμοποιήσουμε μόνο C (όχι C++)

- Πίνακες:

  ```
  sort(int array[]);
  ```

  - Όταν το όνομα ενός πίνακα δοθεί ως παράμετρος σε μια συνάρτηση, η τιμή που διοχετεύεται στη συνάρτηση είναι η διεύθυνση της αρχής του πίνακα (δεν αντιγράφονται τα στοιχεία του πίνακα).
  - Έτσι η συνάρτηση μπορεί να αλλάξει τις τιμές του πίνακα, και οι αλλαγές γίνονται ορατές στην κληθείσα συνάρτηση.
  - Με χρήση δεικτών μπορούμε να αναφερθούμε στα στοιχεία του πίνακα.

Εξωτερικές/Εσωτερικές Μεταβλητές

- Οι εξωτερικές μεταβλητές ορίζονται έξω από το σώμα οποιασδήποτε συνάρτησης και έτσι μπορούν να χρησιμοποιηθούν από πολλές συναρτήσεις. Αντίθετα οι εσωτερικές μεταβλητές ορίζονται στο σώμα κάθε συνάρτησης και είναι γνωστές μόνο στη συνάρτηση στην οποία ορίζονται.

```
#include <stdio.h>

int  sum;     /* external variable */

int main(void) {
   int i;       /* local or internal or automatic
  variable */
     ……
}
```

Εξωτερικές Μεταβλητές (συν.)

Χρήση Εξωτερικών μεταβλητών:

- Θεωρείτε ένας εναλλακτικός μηχανισμός για πέρασμα παραμέτρων σε συναρτήσεις και επιστροφή τιμών από αυτές.

Προσοχή:

- Λανθασμένη χρήση τους (υπερβολική χρήση) καταστρέφει τη δομή και την αναγνωσιμότητα του κώδικα
- Συναρτήσεις που έχουν έναν πολύ μεγάλο αριθμό παραμέτρων καλύτερα να κάνουν χρήση των εξωτερικών μεταβλητών
- Οι εξωτερικές μεταβλητές είναι επίσης χρήσιμες λόγω της μεγαλύτερης εμβέλειας και της μεγαλύτερης διάρκεια ζωής σε σχέση με τις εσωτερικές μεταβλητές.

Κανόνες Εμβέλειας

- Η **εμβέλεια** του ονόματος μιας μεταβλητής ή μιας συνάρτησης είναι το τμήμα του προγράμματος στο οποίο το όνομα αυτό μπορεί να χρησιμοποιηθεί.

- Οι **αυτόματες μεταβλητές** (automatic variables) που δηλώνονται στην αρχή κάθε συνάρτησης έχουν εμβέλεια μόνο τη συνάρτηση στην οποία δηλώνονται. Το ίδιο ισχύει και για τα ονόματα των παραμέτρων μιας συνάρτησης.

  void swap(int firstpart, int secondpart);

- Οι **εξωτερικές μεταβλητές** έχουν εμβέλεια από το σημείο στο οποίο δηλώθηκαν ως στο τέλος του αρχείου αυτού.

Κανόνες Εμβέλειας (συν.)

- Είναι πολύ σημαντικός ο διαχωρισμός μεταξύ **δήλωσης** μιας εξωτερικής μεταβλητής και **ορισμού** μιας εξωτερικής μεταβλητής.
- **Δήλωση Μεταβλητής (Declaration)**
  - Κάνουμε γνωστές τις ιδιότητες της μεταβλητής (κυρίως τον τύπο της)

```
extern int a;
```

- **Ορισμός Μεταβλητής (Definition)**
  - Δηλώνουμε μια μεταβλητή και επιπλέον δεσμεύουμε χώρο γι΄αυτή τη μεταβλητή.

```
int a = 1;
```

- Πρέπει να υπάρχει μόνο ένας **ορισμός** μιας εξωτερικής μεταβλητής σε **ένα και μόνο αρχείο** από τα πολλά τα οποία αποτελούν την οντότητα ενός προγράμματος. Στα υπόλοιπα αρχεία μπορούν να υπάρχουν δηλώσεις τύπου extern για την προσπέλαση της μεταβλητής αυτής.

Παράδειγμα....

```
int main(void) { ……}

int sp =  0;
double val[MAXVAL];

void push(double f) {……}
double pop(void)  {……}
```

- Οι μεταβλητές sp και val μπορούν να χρησιμοποιηθούν στις συναρτήσεις push() και pop() απλά μέσω του ονόματός τους χωρίς καμιά επιπλέον δήλωση. Αλλά οι μεταβλητές αυτές δεν είναι γνωστές στο main(), ούτε και τα ονόματα των συναρτήσεων push() και pop().

Κανόνες Εμβέλειας (συν.)

- Αν μία εξωτερική μεταβλητή πρέπει να χρησιμοποιηθεί πριν τον ορισμό της ή έχει οριστεί σε ένα διαφορετικό αρχείο τότε η δήλωση της μεταβλητής ως extern είναι απαραίτητη.
- Για παράδειγμα:

```
extern int sp;
extern double val[];

int main(void) {……}

int sp=0;
double val[MAXVAL];

void push(double f) {……}
```

Κανόνες Εμβέλειας (συν.)

Αξίζει να σημειωθεί ότι **όπως στις μεταβλητές** έτσι και στις συναρτήσεις, όταν πρέπει να κάνουμε κλήση μιας συνάρτησης η οποία έχει οριστεί παρακάτω ή σε κάποιο άλλο αρχείο τότε απαιτείται η δήλωση του πρωτοτύπου της συνάρτησης.

```
int square(int);     /* function prototype */

int main(void) {
…..
  x = square(5);
}

int square(int I) {                    /* function
  definition */
……
}
```

Διάρκεια Μεταβλητών

- Οι **εξωτερικές μεταβλητές** έχουν διάρκεια ως το τέλος του προγράμματος.
- Οι **τοπικές μεταβλητές** σε μία συνάρτηση έχουν διάρκεια ως το τέλος της εκτέλεσης της συνάρτησης.
- Σε περίπτωση που απαιτείται μια τοπική μεταβλητή να έχει διάρκεια ζωής όσο όλο το πρόγραμμα χρησιμοποιείται η λέξη κλειδί static κατά τον ορισμό της.

```
void func(int x) {
     static int temp;        Να αποφεύγεται
 …}                          αν είναι δυνατόν
```

- Ορίζοντας μια εξωτερική μεταβλητή ως static τότε η εμβέλεια της μεταβλητής περιορίζεται στο αρχείο όπου αυτή ορίζεται και διαφέρει από οποιεσδήποτε συνώνυμες μεταβλητές σε άλλα αρχεία που αποτελούν το ίδιο πρόγραμμα.

Δηλώσεις Μεταβλητών (συν.)

- Οι τοπικές (ή αλλοιώς αυτόματες) μεταβλητές που ορίζονται στο σώμα των συναρτήσεων καθώς και οι τυπικές παράμετροι των συναρτήσεων **κρύβουν** (**override**) οποιεσδήποτε εξωτερικές μεταβλητές ή συναρτήσεις με το ίδιο όνομα.

```
int x;
  int y;

  void f(double x) {
        double y;
        ……
  }
```

- Μέσα στη συνάρτηση f, το όνομα x αναφέρεται στην παράμετρο της συνάρτησης που είναι ένας double και όχι στην εξωτερική μεταβλητή που είναι int. Το ίδιο ισχύει και για τη μεταβλητή y.

- Προτείνεται να μη χρησιμοποιούνται μεταβλητές στις συναρτήσεις τα ονόματα των οποίων να συγκρούονται με μεταβλητές του εξωτερικού επιπέδου.

Αρχικοποίηση Μεταβλητών

Γενικοί κανόνες αρχικοποίησης εξωτερικών και στατικών μεταβλητών:

- **Απουσία ρητής αρχικοποίησης**
  - οι **εξωτερικές** και οι **στατικές** (static) μεταβλητές αρχικοποιούνται με την **τιμή μηδέν**.

- **Ρητή αρχικοποίηση**
  - Οι μεταβλητές μπορούν να λάβουν πρέπει να είναι μία **σταθερή έκφραση** (constant expression).

```
int x = 1;
char squote = '\'';
```

- Η αρχικοποίησή τους γίνεται μόνο μία φορά πριν ακριβώς την εκτέλεση του προγράμματος.

Αρχικοποίηση Μεταβλητών (συν.)

Γενικοί κανόνες αρχικοποίησης αυτόματων μεταβλητών

- Σε περίπτωση ρητής αρχικοποίησης, η τιμή που μπορούν να λάβουν μπορεί να είναι **οποιαδήποτε** έκφραση που περιέχει **μεταβλητές** ή ακόμα και **κλήσεις συναρτήσεων**.

```
int f(int x, int v[], int n) {
      int low = 0;
      int high = n-1;
      ……
  }
```

Αρχικοποίηση Μεταβλητών (συν.)

Κανόνες Αρχικοποίησης Πινάκων

- Οι αρχικές τιμές ενός πίνακα δίνονται κατά τον ορισμό του, περιέχονται μεταξύ δεξιάς και αριστερής αγκύλης και διαχωρίζονται με κόμμα:

```
int a[] = { 1, 5, 18, 23};
```

- Αν η διάσταση παραληφθεί τότε υπολογίζεται από τον μεταφραστή βάσει του πλήθους των αρχικών τιμών (στο παράδειγμα η διάσταση του πίνακα a θα είναι τέσσερα).
- Αν οι αρχικές τιμές που δίνονται είναι λιγότερες της διάστασης τότε οι υπόλοιπες θέσεις τίθονται μηδέν για εξωτερικές, στατικές αλλά και αυτόματες μεταβλητές τύπου πίνακα.
- Είναι λάθος να δίνονται περισσότερες αρχικές τιμές από τις θέσεις του πίνακα.

Αρχικοποίηση Μεταβλητών (συν.)

Κανόνες Αρχικοποίησης Πινάκων (συν.)

- Δεν υπάρχει τρόπος επανάληψης της ίδιας αρχικής τιμής για όλες τις θέσεις ενός πίνακα (εκτος του 0 δηλ int a[10] ={})

- Στους πίνακες χαρακτήρων υπάρχει ένας εναλλακτικός πιο εύκολος τρόπος αρχικοποίησης αντί της χρήσης των { , } και , .

```
char text[] = "Hello";
```

είναι συντομογραφία του

```
char text[] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

Ή διάσταση του πίνακα γίνεται και στις δύο περιπτώσεις έξι (πέντε χαρακτήρες συν ο τερματικός χαρακτήρας '\0').