Α' Εξάμηνο
ΕΙΣΑΓΩΓΗ ΣΤΟ ΔΟΜΗΜΕΝΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Εργαστήριο – 2η σειρά ασκήσεων.
Κοζάνη, 19 Οκτωβρίου 2007.

Δεν ξεχνάμε τα εξής:

- Case Sensitive!!! Άλλα τα κεφαλαία και άλλα τα μικρά! Με ο,τι τρόπο γράψουμε ένα όνομα στην αρχή με αυτόν πρέπει να το γράφουμε μέχρι το τέλος.
- Οι εντολές τελειώνουν με ;
- Οι διάφορες συναρτήσεις αλλά και τα τμήματα κώδικα (βασικές δομές) στο εσωτερικό τους αρχίζουν με { και τελειώνουν με }. Συνεπώς, προσέχουμε να κλείνουν τα άγκιστρα (για κάθε { να υπάρχει και ένα })
- Τα αλφαριθμητικά, π.χ. τα μηνύματα που τυπώνουμε με την printf, αρχίζουν με “ και τελειώνουν πάλι με “. Συνεπώς, προσέχουμε να κλείνουμε και τα εισαγωγικά.

Πρόγραμμα p2-1
Ένας φοιτητής προσπάθησε να γράψει και να εκτελέσει το παρακάτω απλό πρόγραμμα από την προηγούμενη σειρά ασκήσεων. Όμως έκανε κάποια λάθη οπότε το πρόγραμμα δε μπορεί να μεταγλωττιστεί. Βοηθήστε τον ανακαλύπτοντας τα λάθη και βεβαιωθείτε γράφοντας σωστά, μεταγλωττίζοντας και εκτελώντας το πρόγραμμα.

```
#include <stdio,h>

int intSum(void)
{
    int x, Number, Count, Sum

    Count = 0;
    Sum = 0;
    printf("Dose mou ena plithos akereon arithmon: ");
    scanf("%d", Number)
    while(Count < Number);
    {
        printf("Dose mou ton ip' arithmon d oro: ", Count+1);
        scanf("%d", x);
        Count++
        Sum += x;
    }
    return Sum;


void main(void)
(
    int Sum;

    Sum = intSum();
    printf("To athrisma olon ine %d\", Sum);
)
```

Πρόγραμμα p2-2
Εδώ εισάγουμε δύο παραλλαγές του τύπου int και βλέπουμε στην πράξη τα όρια των αριθμών που μπορούν ή δεν μπορούν να αναπαραστήσουν. Επίσης, εισάγεται μια νέα οδηγία προς τον προεπεξεργαστή η οποία μας βοηθάει να εκφράζουμε συμβολικά κάποιες σταθερές που εμφανίζονται συχνά μέσα στο πρόγραμμά μας. Τέλος, η συνάρτηση getchar( ) που διαβάζει χαρακτήρες εισάγεται για να παίξει βοηθητικό ρόλο στην εκτέλεση του προγράμματος.
Προσοχή στις προδιαγραφές μορφοποίησης (%) στην printf!

```
#include <stdio.h>

#define BIG 10000 /* δέκα χιλιάδες */

int main()
{
    unsigned short int i = 0;
    unsigned int ii = 0;
    unsigned long int iii = 0;
```

```
    printf("\nPress Enter to continue, 0 to stop\n");
    while(getchar() != '0')
    {
        i += BIG;
        ii += BIG;
        iii += BIG;
        printf("i = %d, ii = %d, iii = %ld \n", i, ii, iii);
    }
}
```

Πειραματιστείτε βγάζοντας τη λέξη unsigned από τους τύπους και επανεκτελώντας το πρόγραμμα.

Πρόγραμμα p2-3
Παραλλαγή του προηγούμενου, για να εισάγουμε νέους τύπους αναπαράστασης των πραγματικών αριθμών.
Και πάλι, προσοχή στις προδιαγραφές μορφοποίησης (%) στην printf!

```
#include <stdio.h>

#define BIG 1000 /* χίλια */

int main()
{
    float x = 1.0;
    double xx = 1.0;
    long double xxx = 1.0;

    printf("\nPress Enter to continue, 0 to stop\n");
    while(getchar() != '0')
    {
        x *= BIG;
        xx *= BIG;
        xxx *= BIG;
        printf("x = %f, xx = %lf, xxx = %Lf \n", x, xx, xxx);
    }
}
```

Πρόγραμμα p2-4
Εισάγεται η έννοια του αρχείου και η συνάρτηση fprintf που μας επιτρέπει να γράψουμε δεδομένα σε ένα αρχείο. Χρησιμοποιείται ο τελεστής αύξησης για να εκφράσει μία συνθήκη που θα επιτρέψει την εκτέλεση μιας δομής επανάληψης για ένα καθορισμένο αριθμό φορών (10).

```
#include <stdio.h>

int main()
{
    int i, j, k, count = 0;
    FILE *myFile;

    printf("\nPress Enter to write a file");
```

```
    getchar();
    myFile = fopen("test.txt", "w");
    while(count++ < 10)
    {
        printf("\nGive me three numbers: \n");
        scanf("%d%d%d", &i, &j, &k);
        printf("Record %d: i=%d, j=%d, k=%d \n", count, i, j, k);
        fprintf(myFile, "%d    %d    %d \n", i, j, k);
    }
    fclose(myFile);
}
```

Πρόγραμμα p2-5

Σχετικό με το προηγούμενο πρόγραμμα, που χρησιμοποιεί τη συνάρτηση fscanf για να διαβάσει και να τυπώσει στην οθόνη τα δεδομένα από το αρχείο που δημιουργήθηκε κατά την προηγούμενη άσκηση. Εδώ χρησιμοποιείται ο τελεστής μείωσης αντί του τελεστή αύξησης, σε μια παραλλαγή της προηγούμενης συνθήκης για τον έλεγχο της επανάληψης, επιτελώντας ουσιαστικά την ίδια λειτουργία.

```
#include <stdio.h>

int main()
{
    int i, j, k, count = 0;
    FILE *myFile;

    printf("\nPress Enter to read each record of the file");
    getchar();

    count = 10;
    myFile = fopen("test.txt", "r");
    while(--count >= 0)
    {
        printf("\nRecord %d: ", 10 - count);
        fscanf(myFile, "%d%d%d", &i, &j, &k);
        printf("%d  %d  %d", i, j, k);
        getchar();
    }
    fclose(myFile);
}
```

Μερικά χρωστούμενα από την προηγούμενη σειρά...

Πρόγραμμα p1-6

Παράδειγμα επανάληψης συνδυασμένο και με επιλογή/λήψη απόφασης. Πρόκειται για παραλλαγή του αλγόριθμου εύρεσης μέγιστου και ελάχιστου που αναφέρθηκε στο πρώτο μάθημα.

```
#include <stdio.h>

void main(void)
{
    int x, number, count, minimum=0, maximum=0;
```

```
    printf("Give me a number of positive integers: ");
    scanf("%d", &number);

    printf("Give me the term number %d: ", 1);
    scanf("%d", &x);
    minimum = x;
    if(x > maximum)
        maximum = x;
    count = 1;

    while(count < number)
    {
        printf("Give me the term number %d:: ", count+1);
        scanf("%d", &x);
        if(x < minimum)
            minimum = x;
        if(x > maximum)
            maximum = x;
        count++;
    }

     printf("The maximum is %d and the minimum is %d \n", maximum,
minimum);
}
```

Το Πρόγραμμα p1-8 ελαφρώς αλλαγμένο

Πρόγραμμα που εξηγεί παραστατικά την έννοια της εμβέλειας μεταβλητών ή των τοπικών μεταβλητών, καθώς και και τις πρακτικές συνέπειες του περάσματος ορισμάτων με τιμή σε αντίθεση με το πέρασμα μέσω αναφοράς. Γραμμένο σε δύο παραλλαγές. Παρατηρείστε τα διαφορετικά αποτελέσματα που προκύπτουν σε κάθε περίπτωση.

<u>Παραλλαγή πρώτη:</u>

```
#include <stdio.h>

void testFunc(int, int);

void main(void)
{
    int x, y;
    printf("\nGive me an integer x :");
    scanf("%d", &x);
    printf("\nGive me another one y :");
    scanf("%d", &y);
    testFunc(x, y);
     printf("\nThe values of x and y after calling testFunc are %d
and %d, respectively\n", x, y);
}

void testFunc(int x, int y)
```

```
{
    printf("\nx and y equal %d and %d, respectively", x, y);
    x += 2;
    y -= -x;
    printf("\nNow,x and y equal %d and %d, respectively", x, y);
}
```

Παραλλαγή δεύτερη:

```
#include <stdio.h>

void testFunc(int *, int *);

int main(void)
{
    printf("\nGive me an integer x :");
    scanf("%d", &x);
    printf("\nGive me another one y :");
    scanf("%d", &y);
    testFunc(&x, &y);
     printf("\nThe values of x and y after calling testFunc are %d
and %d, respectively\n", x, y);
}

void testFunc(int *a, int *b)
{
    printf("\nx and y equal %d and %d, respectively", *a, *b);
    *a += 2;
    *b -= *a;
    printf("\nNow,x and y equal %d and %d, respectively", *a, *b);
}
```