Α' Εξάμηνο
ΕΙΣΑΓΩΓΗ ΣΤΟ ΔΟΜΗΜΕΝΟ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟ

Ασκήσεις Επανάληψης

Άσκηση 1

Το παρακάτω πρόγραμμα περιέχει μια δομή επανάληψης τύπου for. Να ξαναγραφεί ώστε να έχει ακριβώς την ίδια λειτουργία, χρησιμοποιώντας δομή while

```
#include <stdio.h>

int main(int argc, char **argv)
{
    int i, n=10, a=0, g=1;
    for(i=1; i<n+1; i++) {
        a = a + i;
        g = g * i;
        printf("Όρος: %d. Αριθμητική πρόοδος: %d. Γεωμετρική πρόοδος: %d\
n", i, a, g);
    }
    return 0;
}
```

Άσκηση 2

Το παρακάτω πρόγραμμα περιέχει ένα διπλό for. Να μετατραπεί ώστε να κάνει το ίδιο ακριβώς, με δομές τύπου while

```
#include <stdio.h>
#define N 3
int main(int argc, char **argv)
{
    int i, j, n, A[N][N], B[N][N]={1,2,3,2,1,3,3,2,1};

    for(i=0;i<N;i++)
        for(j=0;j<N;j++)
            A[i][j]=-B[i][j];
    return 0;
}
```

Άσκηση 3

Η παρακάτω δομή περιέχει μία δομή επανάληψης for με τη δυνατότητα διαφυγής από αυτή με χρήση της εντολής goto. Μετατρέψτε το πρόγραμμα ώστε να κάνει το ίδιο με μια δομή επανάληψης τύπου while αλλά η διαφυγή να υλοποιείται με άλλο τρόπο αντί της goto.

```
#include <stdio.h>
#include <math.h>
#define N 10
int main(int argc, char **argv)
{
    int i,x;
    for(i=0; i<N; i++) {
        printf("Δώσε έναν ακέραιο: ");
        scanf("%d",&x);
```

```
        if(x<0)
            goto finish;
        printf("Η ρίζα του %d είναι: %lf\n", x, sqrt((double) x));
    }
finish: printf("Τελειώσαμε!\n");
    return 0;
}
```

Άσκηση 4

Η παρακάτω δομή περιέχει μία δομή επανάληψης for με τη δυνατότητα διαφυγής από αυτή με χρήση της εντολής goto. Μετατρέψτε το πρόγραμμα ώστε να κάνει το ίδιο με μια δομή επανάληψης τύπου while αλλά η διαφυγή να υλοποιείται με άλλο τρόπο αντί της goto.

```
#include <stdio.h>
#include <math.h>
int main(int argc, char **argv)
{
    int x;
    for(; ; ) {
        printf("Δώσε έναν ακέραιο (αρνητικό για τέλος): ");
        scanf("%d",&x);
        if(x<0)
            goto finish;
        printf("Η ρίζα του %d είναι: %lf\n", x, sqrt((double) x));
    }
finish: printf("Τελειώσαμε!\n");
    return 0;
}
```

Άσκηση 5

Το παρακάτω πρόγραμμα περιέχει μία δομή switch. Να μετατραπεί έτσι ώστε να εκτελεί την ίδια λειτουργία με μία δομή if.
Υπόδειξη: θα χρειαστείτε μία μεταβλητή τύπου char για να αποθηκεύσετε την απάντηση του χρήστη.

```
#include <stdio.h>
int main(int argc, char **argv)
{
    FILE *fp=fopen("votes", "a");
    printf("Σας αρέσει το μάθημα του Δομημένου Προγραμματισμού;\n");
    printf("Ναι = N, Όχι = O, Άλλο = *\n");
    switch(getchar( )) {
        case 'N':
        case 'n':
              fprintf(fp, "Το πάω με χίλια!\n");
              break;
         case 'O':
         case 'o':
              fprintf(fp, "Μου τη δίνει!\n");
              break;
          default:
              fprintf(fp, "Δεν ξέρω/Δεν απαντώ\n");
    }
    fclose(fp);
    return 0;
}
```

Άσκηση 6:

Το παρακάτω πρόγραμμα περιέχει μία πολλαπλή δομή if. Μετατρέψτε το πρόγραμμα ώστε να επιτελεί την ίδια λειτουργία με μία δομή switch.

Υπόδειξη: η fprintf που αντιστοιχεί στην περίπτωση βαθμών από 1 έως 5 μπορεί να συνενωθεί με καθεμία από τις fprintf των επιμέρους περιπτώσεων. Οι μη έγκυροι βαθμοί αντιστοιχούν στην περίπτωση default.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char **argv)
{
    int n;
    char answer[1];
    FILE *fp=fopen("votes", "a");
    printf("Τι γνώμη έχετε για το μάθημα του Δομημένου
Προγραμματισμού;\n");
    printf("Βαθμολογήστε από 1 ως 5. Δώστε άλλο αριθμό αν δεν έχετε
γνώμη.\n");
    scanf("%s", answer);
    n = atoi(answer);
    if(n<1 || n > 5)
          fprintf(fp, "Δεν ξέρω/Δεν απαντώ\n");
    else {
        fprintf(fp, "Βαθμός: %d. ", n);
        if(n==1)
            fprintf(fp, "@$^#%$^#$!\n");
        else if(n==2)
            fprintf(fp, "Δε λέει...\n");
        else if(n==3)
            fprintf(fp, "Χμ...\n");
        else if(n==4)
            fprintf(fp, "Λέει!\n");
        else if(n==5)
            fprintf(fp, "Άπαιχτο!\n");
    }
    fclose(fp);
    return 0;
}
```

Άσκηση 7

Το παρακάτω πρόγραμμα περιέχει δύο ένθετες δομές επανάληψης τύπου while και τη δυνατότητα διαφυγής και από τις δύο με χρήση της εντολής goto. Να μετατραπεί το πρόγραμμα ώστε να έχει την ίδια λειτουργία, με χρήση ένθετων δομών τύπου for και διαφυγή και από τις δύο με χρήση της εντολής break.

Υπόδειξη: επειδή η εντολή break μας βγάζει μόνο από ένα for, θα χρειαστούν δύο break, ένα σε κάθε for, με το break στο εξωτερικό for να εκτελείται αν ισχύει κάποια κατάλληλη συνθήκη, π.χ. σε σχέση με την τιμή του εσωτερικού μετρητή j.

```
#include <stdio.h>
#define N 3
int main(int argc, char **argv)
{
    int i, j, n, A[N][N], B[N][N];
    FILE *fp=fopen("data","r");
    i = 0;
```

```
    while ( i<N ) {
        j = 0;
        while ( j<N ) {
            fscanf(fp,"%f", &n);
            if( (B[i][j] = n) == 0 )
                goto finish;
             A[i][j] = 1 / n;
             j++;
        }
         i++;
    }
finish: printf((n?"Τελειώσαμε!\n":"Διαίρεση με το μηδέν!\n"));
    fclose(fp);
    return 0;
}
```

Άσκηση 8

Το παρακάτω πρόγραμμα υλοποιεί μια δομή επιλογής με τη βοήθεια του τριαδικού τελεστή ?:. Ζητείται να αναδιατυπωθεί ώστε να εκτελεί την ίδια ακριβώς λειτουργία με τη βοήθεια της δομής επιλογής τύπου if.

```
#include <stdio.h>

int main (int argc, char **argv)
{
    int a, b, c, d;
    printf("Δώσε τρεις ακέραιους: \n");
    scanf("%d", &a);
    scanf("%d", &b);
    scanf("%d", &c);
    a = a>b?a:b;
    a = a>c?a:c;
    printf("Ο μέγιστος είναι ο %d\n", a);
    return 0;
}
```

Άσκηση 9

Το παρακάτω πρόγραμμα έχει 5 λάθη. Μπορείτε να τα βρείτε;
Υπόδειξη: αν το ίδιο λάθος εμφανίζεται πάνω από μία φορά, μετράει μόνο σαν ένα.

```
int main(int argc, char **argv)
{
    int x, y
    printf("Δώσε δύο ακέραιους αριθμούς: \n");
    scanf("%f", x);
    scanf("%f", y);
    if(y != 0)
        printf("Το πηλίκο τους είναι %d\n", ((float) x) / ((float) y));
    return 0;
}
```

Άσκηση 10

Στο παρακάτω πρόγραμμα, οι παράμετροι εισόδου της συνάρτησης func περνάνε κατά διεύθυνση. Επειδή ο προγραμματιστής δε θέλει να αλλάξει την τιμή τους, ορίζει τοπικές μεταβλητές στις οποίες αντιγράφει τις τιμές τους και κάνει τις απαραίτητες ενέργειες με αυτές. Αυτός που έγραψε το πρόγραμμα δεν έχει πείρα από τη γλώσσα C και δεν κατάλαβε ότι δε χρειαζόταν να τα κάνει όλα

αυτά. Μπορείτε να του δείξετε πώς αλλιώς μπορεί να γράψει και να καλέσει τη func;
Υπόδειξη 1: εκτός από πέρασμα κατά διεύθυνση, ποιον άλλο τρόπο έχουμε για να περνάμε τα ορίσματα των συναρτήσεων;
Υπόδειξη 2: για να διατηρήσετε τις τιμές εισόδου που θα τυπώνονται από το μήνυμα της func στην οθόνη χωρίς να ορίσετε τοπικές μεταβλητές, σπάστε την printf σε δύο κλήσεις: η μία στην αρχή της συνάρτησης θα τυπώνει το πρώτο μισό του μηνύματος χωρίς \n και η άλλη, στο τέλος, θα τυπώνει το άλλο μισό, δηλαδή τη διαφορά των απόλυτων τιμών.

```
#include <stdio.h>

void func (int *, int *);

int main(int argc, char **argv)
{
    int x, y;
    printf("Δώσε δύο ακέραιους αριθμούς: \n");
    scanf("%d", &x);
    scanf("%d", &y);
    func(&x, &y);
    printf("%d-%d=%d\n",x, y, x-y);
    return 0;
}

void func(int *x, int *y)
{
    int a, b;
    if(*x < 0)
        a = -*x;
    else
        a = *x;
    if(*y < 0)
        b = -*y;
    else
        b = *y;
    printf("|%d|-|%d|=%d\n",*x, *y, a-b);
    return;
}
```