ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
Εθνικόν και Καποδιστριακόν Πανεπιστήμιον Αθηνών

Σχολή Θετικών Επιστημών
ΤΜΗΜΑ ΑΕΡΟΔΙΑΣΤΗΜΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ

104
Αλγόριθμοι, Προγραμματισμός και Δομές Δεδομένων

Εργαστήριο – Κεφάλαιο 6β & 7α
Δομές επανάληψεις (while) & Μονοδιάστατοι πίνακες

Διδάσκουσα:

Σοφία Αλμπάνη

Άσκηση 6.1

6.1) Να γραφεί πρόγραμμα σε γλώσσα **C** το οποίο να ζητά από τον χρήστη έναν αριθμό, να υπολογίζει το άθροισμα **τετραγώνων διαδοχικών θετικών ακεραίων** και να βρίσκει ποιος είναι ο **τελευταίος αριθμός** που μπορεί να προστεθεί πριν το άθροισμα ξεπεράσει μία δοσμένη τιμή.

Παράδειγμα Εκτέλεσης
Enter number: 50
The last number is = 4
(Γιατί $1^2 + 2^2 + 3^2 + 4^2 = 30 < 50$, ενώ $+ 5^2 = 55 \geq 50 \rightarrow$ σταματά πριν το 5)

Στόχος:
- Χρήση άπειρης επανάληψης while(1) με έξοδο μέσω break.
- Εξάσκηση στη συσσώρευση αθροισμάτων.
- Εφαρμογή αριθμητικών υπολογισμών μέσα σε επανάληψη.

Ενδεικτική Λύση 6.1

```
#include <stdio.h>
int main()
{
        int i, num, sum;

        printf("Enter number: ");
        scanf("%d", &num);
        if(num <= 0)
        {
                printf("Error: The number should be positive\n");
                return 0;
        }
        sum = 0;
        i = 1;
        while(1)
        {
                sum += i*i;
                if(sum >= num)
                        break;
                i++;
        }
        printf("The last number is = %d\n", i-1);     }
```

Άσκηση 6.2

6.2) Να γραφεί πρόγραμμα σε γλώσσα **C** το οποίο να ζητά από τον χρήστη επαναλαμβανόμενα να εισάγει έναν ακέραιο αριθμό (num).

- Αν ο αριθμός είναι **αρνητικός**, η επανάληψη **σταματά** (τερματισμός με break).
- Αν ο αριθμός είναι **μη αρνητικός**, τότε:
  - Εκτελείται ένας **εσωτερικός βρόχος while**, ο οποίος τυπώνει τη λέξη: «Hello» **όσες φορές** δείχνει η τιμή του αριθμού.
  - Ο αριθμός αυτός προστίθεται στο μετρητή times, ο οποίος κρατά το **σύνολο** των επαναλήψεων.
- Μετά τον τερματισμό, εμφανίζεται ο συνολικός αριθμός εμφανίσεων της λέξης "Hello".

Στόχος:
- Χρήση άπειρης επανάληψης: while(1)
- Διακοπή επανάληψης με break
- Χρήση εσωτερικού while για καθορισμένο αριθμό επαναλήψεων
- Χρήση μεταβλητής συσσώρευσης (times)

Ενδεικτική Λύση 6.2

```
#include <stdio.h>
int main()
{
	int i, num, times;

	times = 0;
	while(1)
	{
		printf("Enter number: ");
		scanf("%d", &num);
		if(num < 0)
			break;
		i = 0;
		while(i < num)
		{
			printf("Hello\n");
			i++;
		}
		times += num;
	}
	printf("Total number is = %d\n", times);
	}
```

Άσκηση 6.3

6.3) Να γραφεί πρόγραμμα σε C που να διαβάζει επαναλαμβανόμενα τη βαθμολογία **ασκήσεων** και **εξέτασης** μαθητών.

- Η εισαγωγή τερματίζει όταν δοθεί **-1** σε έναν από τους δύο βαθμούς.
- Αν κάποιος βαθμός δεν είναι στο διάστημα **[0–10]**, να εμφανίζεται μήνυμα λάθους και να ζητείται ξανά.
- Αν **και οι δύο βαθμοί ≥ 5**, ο τελικός βαθμός να υπολογίζεται ως:

$$0.3 \times \textit{βαθμος ασκησεων} + 0.7 \times \textit{βαθμος εξετασης}$$

- Διαφορετικά, ο τελικός βαθμός να είναι ο **μικρότερος** από τους δύο βαθμούς.
- Στο τέλος, να εμφανίζεται ο **μέσος όρος των τελικών βαθμών** όλων των μαθητών.

Στόχος:

- Χρήση άπειρης επανάληψης while(1), έλεγχος δεδομένων εισόδου και χρήση της εντολής continue, και όροι με λογικούς τελεστές,

Ενδεικτική Λύση 6.3

```
#include <stdio.h>
int main()
{
        int studs;
        float sum_grd, stud_grd, grd_exc, grd_exam;

        studs = 0;
        sum_grd = 0;
        while(1)
        {
                printf("--------------------------------\n");

                printf("Enter exercise grade [0-10]: ");
                scanf("%f", &grd_exc);

                printf("Enter exam grade [0-10]: ");
                scanf("%f", &grd_exam);

                if(grd_exc == -1 || grd_exam == -1)
                        break;

                if((grd_exc < 0) || (grd_exc > 10) ||
(grd_exam < 0) || (grd_exam > 10))
                {
                        printf("Error: Grades should be in [0-10]\n");
                        continue;
                }
```

Ενδεικτική Λύση 6.3

```
studs++;
                if(grd_exc >= 5 && grd_exam >= 5)
                        stud_grd = 0.3*grd_exc + 0.7*grd_exam;
                else
                {
                        if(grd_exc <= grd_exam)
                                stud_grd = grd_exc;
                        else
                                stud_grd = grd_exam;
                }
                printf("Student grade = %.2f\n", stud_grd);
                sum_grd += stud_grd;
        }
        if(studs)                       /* Ισοδύναμο με if(studs != 0). */
                printf("\nAverage grade = %.2f\n", sum_grd/studs);
}
```

Άσκηση 6.4

6.4) Να γραφεί πρόγραμμα σε C που να πραγματοποιεί **εξάσκηση στον πολλαπλασιασμό** με τυχαίους αριθμούς. Το πρόγραμμα:

- Παράγει τυχαία δύο αριθμούς μεταξύ **1 και 10**.
- Εμφανίζει την πράξη πολλαπλασιασμού και ζητά από τον χρήστη την απάντηση.
- Αν η απάντηση είναι σωστή, αυξάνει τον μετρητή **σωστών** (wins), αλλιώς αυξάνει τον μετρητή **λανθασμένων** (fails) και εμφανίζει τη σωστή λύση.
- Η διαδικασία συνεχίζεται **μέχρι ο χρήστης να εισάγει -1**.
- Στο τέλος, εμφανίζει το πλήθος των σωστών και λανθασμένων απαντήσεων.

Στόχος

Χρήση της δομής do–while για επαναλαμβανόμενη εισαγωγή δεδομένων

Χρήση της βιβλιοθήκης stdlib.h για παραγωγή τυχαίων αριθμών (rand()), αρχικοποίηση του γεννήτορα τυχαιότητας με srand(time(NULL)).

Ενδεικτική Λύση 6.4

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main()
{
	int i, j, num, fails, wins;
	fails = wins = 0;
	srand(time(NULL));

	do
	{	i = rand()%10 + 1;
		j = rand()%10 + 1;
		printf("\n%dx%d=", i, j);

		scanf("%d", &num);
```

Ενδεικτική Λύση 6.4

```
        if(num != -1)
        {
                if(num == i*j)
                {
                        printf("Correct\n");
                        wins++;
                }
                else
                {
                        printf("Wrong(answer=%d)\n", i*j);
                        fails++;
                }
        }
} while(num != -1);

printf("Fails = %d, Wins = %d\n", fails, wins);
}
```

Άσκηση 6.5

6.5) Να γραφεί πρόγραμμα σε C που να υπολογίζει μία **προσεγγιστική τιμή του π** χρησιμοποιώντας το ανάπτυγμα της σειράς του Leibniz:

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \cdots$$

Το πρόγραμμα πρέπει να:

- χρησιμοποιεί **δομή επανάληψης do–while**,
- εναλλάσσει το πρόσημο του όρου σε κάθε βήμα,
- σταματά την επανάληψη όταν ο όρος $\frac{1}{i}$γίνει μικρότερος από **0.0001**,
- και να εμφανίζει την τελική προσεγγιστική τιμή του π με **10 δεκαδικά ψηφία**.

Στόχος:
Χρήση της δομής do–while

Ενδεικτική Λύση 6.5

```
#include <stdio.h>
int main()
{
	int i;
	double a, pi;

	pi = 0;
	i = 1;
	a = 4;

	do
	{
		pi += a/i;
		a = -a;
		i = i+2;
	} while(1.0/i >= 0.0001);

	printf("Pi = %.10f\n", pi);
	}
```

Άσκηση 7.1

7.1) Να γραφεί πρόγραμμα σε C το οποίο να διαβάζει **10 ακέραιους αριθμούς** και να τους αποθηκεύει σε έναν πίνακα. Στη συνέχεια, το πρόγραμμα πρέπει να ελέγχει αν ο πίνακας είναι **συμμετρικός**, δηλαδή αν ισχύει:

$$a[0] = a[9],\ a[1] = a[8],\ a[2] = a[7],\ \ldots$$

- Αν βρεθεί ζεύγος στοιχείων που **δεν είναι ίσο**, να εμφανίζεται το μήνυμα: «Non symmetric array» και το πρόγραμμα να τερματίζει.
- Αν όλα τα αντίστοιχα στοιχεία ταιριάζουν, να εμφανίζεται: «Symmetric array»

Στόχος:
Χρήση μονοδιάστατου πίνακα και έλεγχο συμμετρίας τους με χρήση επαναληπτικής δομής.

Ενδεικτική Λύση 7.1

```
#include <stdio.h>
#define SIZE 10
int main()
{
	int i, arr[SIZE];

	for(i = 0; i < SIZE ; i++)
	{
		printf("Enter element a[%d]: ", i);
		scanf("%d", &arr[i]);
	}
	for(i = 0; i < SIZE/2; i++)
		if(arr[i] != arr[SIZE-1-i])
		{
			printf("Non symmetric array\n");
			return 0; /* Αφού διαπιστώσαμε ότι ο πίνακας δεν είναι συμμετρικός,
το πρόγραμμα τερματίζεται άμεσα. */
		}
	printf("Symmetric array\n");
	return 0;
}
```

Άσκηση 7.2

7.2) Να γραφεί πρόγραμμα σε C το οποίο να διαβάζει **10 ακέραιες τιμές** για δύο πίνακες ίδιου μεγέθους (πίνακας 1 και πίνακας 2). Στη συνέχεια, το πρόγραμμα πρέπει να ελέγχει και να εντοπίζει **τα κοινά στοιχεία** των δύο πινάκων.

- Για κάθε κοινό στοιχείο που βρεθεί, να εμφανίζεται:

Common = <τιμή> (Pos_1 = <θέση στον πίνακα 1> , Pos_2 = <θέση στον πίνακα 2>)

- Αν δεν βρεθεί κανένα κοινό στοιχείο, να εμφανίζεται: No common elements were found

Στόχος:
Κατανόηση συσχέτισης δύο πινάκων με χρήση εμφωλευμένων επαναλήψεων (for μέσα σε for).

Ενδεικτική Λύση 7.2

```
#include <stdio.h>
#define SIZE 10
int main()
{
        int i, j, cnt, arr1[SIZE], arr2[SIZE];

        for(i = 0; i < SIZE; i++)
        {
                printf("Enter number_%d for the 1st array: ", i+1);
                scanf("%d", &arr1[i]);
                printf("Enter number_%d for the 2nd array: ", i+1);
                scanf("%d", &arr2[i]);
        }
        cnt = 0;                        /* Μεταβλητή που μετράει τα κοινά στοιχεία. */
        for(i = 0; i < SIZE; i++)
        {
                for(j = 0; j < SIZE; j++)
                {
                        if(arr1[i] == arr2[j])
                        {       cnt++;
                                printf("Common = %d (Pos_1 = %d Pos_2 = %d)\n", arr1[i], i, j);
                        }
                }
        }
        if(cnt == 0)
                printf("No common elements were found\n");}
```

Άσκηση 7.3

7.3) Να γραφεί πρόγραμμα σε C το οποίο:
Δημιουργεί έναν πίνακα **10** θέσεων και τον γεμίζει με **τυχαίους ακέραιους αριθμούς** στο διάστημα **[5, 20]**.
Στη συνέχεια, να **διαγράφει τα διπλότυπα στοιχεία** του πίνακα:

- Κάθε φορά που εντοπίζεται δεύτερη εμφάνιση ενός αριθμού,
- το στοιχείο αυτό να αφαιρείται,
- και τα επόμενα στοιχεία να **μετακινούνται μία θέση αριστερά**.

Μετά τη διαδικασία αφαίρεσης, οι **κενές θέσεις** που απομένουν στο τέλος του πίνακα να συμπληρώνονται με την τιμή **-1**. Τέλος, να εμφανίζεται ο πίνακας **μετά την αφαίρεση των διπλοτύπων**.

Στόχος:
Εξάσκηση σε: χρήση τυχαίων αριθμών (rand() και srand()), εργασία με πίνακες και διαγραφή στοιχείων, εμφωλευμένους βρόχους (for + while), και διατήρηση μεγέθους πίνακα με τη μεταβλητή size.

Ενδεικτική Λύση 7.3

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

#define SIZE 10

int main()
{
	int i, j, k, size, arr[SIZE];

	srand(time(NULL));
	for(i = 0; i < SIZE; i++)
		arr[i] = rand()%16 + 5;

	size = SIZE;
```

Ενδεικτική Λύση 7.3

```
for(i = 0; i < size; i++)
        {
                j = i+1;
                while(j < size)
                {
                        if(arr[i] == arr[j])
                        {
                                for(k = j; k < size-1; k++)
                                        arr[k] = arr[k+1]; /* Μετακίνηση μία θέση αριστερά. */
                                size--; /* Αφού διαγράφηκε το στοιχείο, μειώνουμε τον αριθμό */
                        }
                        else
                                j++;
                }
        }
        for(i = size; i < SIZE; i++)
                arr[i] = -1;
        for(i = 0; i < SIZE; i++)
                printf("%d ", arr[i]);
        return 0;
}
```

Συνοπτικά

Κατηγορία	Συντακτικό / Μορφή	Περιγραφή – Χρήση
Επανάληψη `while`	`while(συνθήκη) { ... }`	Εκτελείται **όσο** η συνθήκη είναι αληθής. Η συνθήκη ελέγχεται **πριν** από κάθε επανάληψη.
Επανάληψη `do–while`	`do { ... } while(συνθήκη);`	Εκτελείται **τουλάχιστον μία φορά**, γιατί η συνθήκη ελέγχεται **μετά** το σώμα της επανάληψης.
Εμφωλευμένες επαναλήψεις	`for(...) { for(...) { ... } }`	Μία επανάληψη μέσα σε άλλη. Χρησιμοποιείται για πίνακες και σχήματα.
Έλεγχος τερματισμού με `break`	`if(συνθήκη) break;`	**Διακόπτει** την επανάληψη και συνεχίζει το πρόγραμμα μετά τον βρόχο.
Παράλειψη βήματος με `continue`	`if(συνθήκη) continue;`	Παραλείπει τις υπόλοιπες εντολές του τρέχοντος βρόχου και περνά στο επόμενο βήμα.
Δήλωση πίνακα	`int arr[SIZE];`	Δέσμευση συνεχόμενης μνήμης για αποθήκευση πολλών τιμών του ίδιου τύπου.
Προσπέλαση στοιχείου	`arr[i]`	Πρόσβαση σε συγκεκριμένη θέση του πίνακα (i-οστό στοιχείο).
Διάσχιση πίνακα	`for(i=0;i<SIZE;i++)` `printf("%d", arr[i]);`	Περνάμε όλα τα στοιχεία με βρόχο.