Εργαστήριο 6

Επαναληπτικές εντολές:
Εντολές **while, do-while** και **for**
Χρήση της **for** σε προβλήματα πινάκων.

Ανακύκλωση **while**.

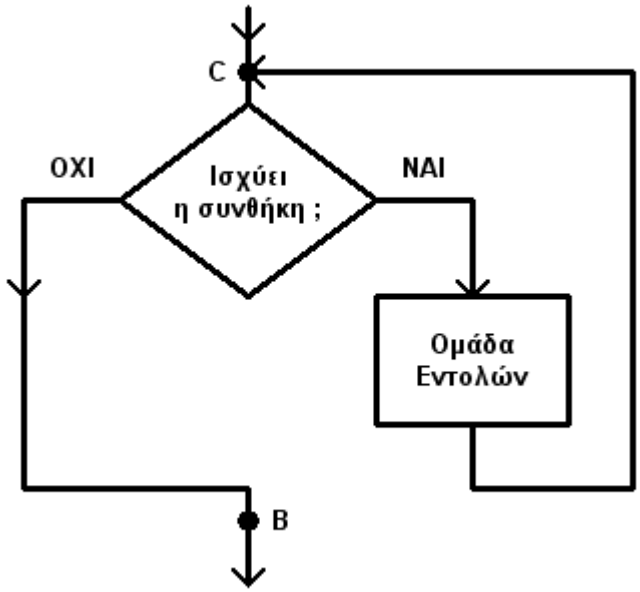
Στο πρόγραμμα γράφουμε :

```
while (Συνθήκη) Ομάδα Εντολών
```

πχ.

```
i = 3;
while (i>0) { printf("%d\n", i); i--; } // τυπώνει 3,2,1
```

Το λογικό διάγραμμα της **while**.



Όταν το πρόγραμμα φτάνει στην εντολή **while**,
υπολογίζει **πρώτα** την λογική τιμή της συνθήκης.

Αν η συνθήκη ισχύει, τότε εκτελείται η ομάδα εντολών.
Μετά η τιμή της συνθήκης υπολογίζεται ξανά.
Αν ισχύει, τότε εκτελείται ξανά η ομάδα εντολών κ.ο.κ.

Αν η συνθήκη ΔΕΝ ισχύει, τότε η ανακύκλωση τερματίζεται
και το πρόγραμμα προχωράει προς την επόμενη εντολή του προγράμματος.

Στην ομάδα εντολών της **while** μπορούμε να χρησιμοποιήσουμε εντολές **break** και **continue**.
Η **break** (όπως και στη **switch**) προκαλεί άμεση έξοδο από την ανακύκλωση (goto **B**).
Η **continue** αγνοεί τις επόμενες εντολές της ομάδας και μας φέρνει στο σημείο **C** (goto **C**).

Ανακύκλωση **do while**.

Στη **while** η συνθήκη υπολογίζεται **στην αρχή**.
Η **do while** πρώτα εκτελεί την ομάδα εντολών και **μετά**
υπολογίζει την τιμή της συνθήκης.

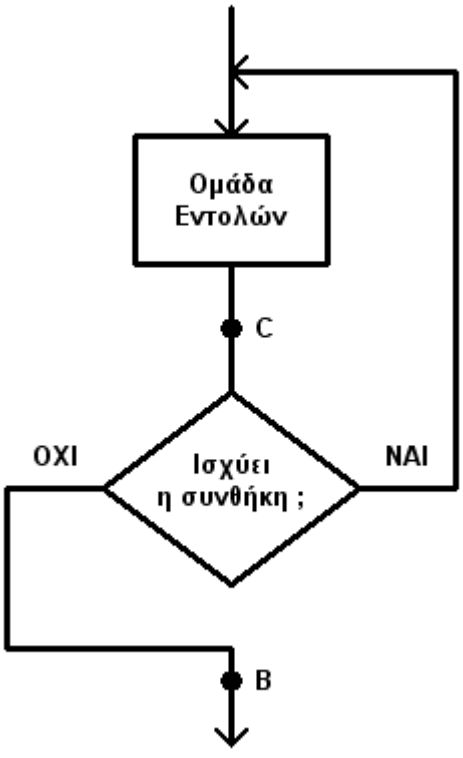
Στο πρόγραμμα γράφουμε :

```
do Ομάδα Εντολών while (Συνθήκη);
```

πχ.

```
i = 3;
do { printf("%d\n", i); i--; } while (i>0);
// τυπώνει 3,2,1
```

Το λογικό διάγραμμα της **do while.**



Στη **while** η συνθήκη μπορεί να είναι ψευδής αμέσως στην αρχή,
με συνέπεια να μην εκτελεστεί η ομάδα εντολών της **while** καθόλου!
Επειδή στη **do while** η συνθήκη υπολογίζεται μετά,
η ομάδα εντολών της **do while** εκτελείται τουλάχιστον μία φορά.

Τα σημεία **B** και **C** δείχνουν πού θα μετακινηθούμε
με την χρήση των εντολών **break** και **continue**.

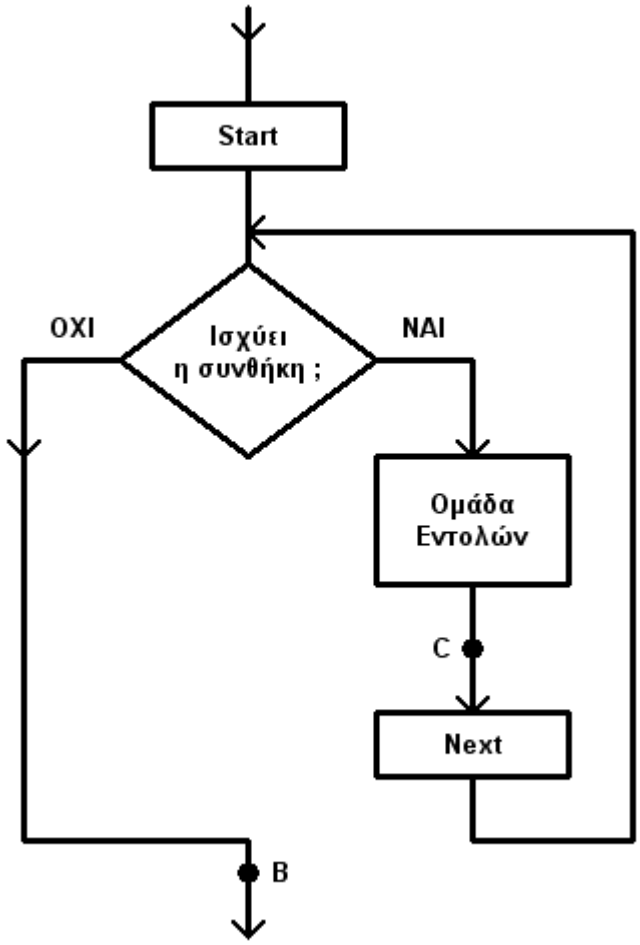
Ανακύκλωση **for**.

Η **for** έχει την εξής μορφή.

```
for ( Start; Συνθήκη; Next ) Ομάδα Εντολών
```

πχ.
```
for ( i=1; i<=3; i++ ) printf("%d\n",i); // τυπώνει 1,2,3
```

Το λογικό διάγραμμα της **for**.



Αν και η **for** φαίνεται δύσκολη στην αρχή, στην πράξη είναι πολύ χρήσιμη.
Η **Start** ορίζει την αρχική τιμή της μεταβλητής (ή των μεταβλητών),
η **Συνθήκη** ελέγχει τον τερματισμό της ανακύκλωσης,
και η **Next** περιγράφει πως προχωράμε στο επόμενο βήμα της ανακύκλωσης.

Το παραπάνω διάγραμμα μας θυμίζει την ανακύκλωση **while**.
Πράγματι, μπορούμε να εκφράσουμε την **for** με μια ειδική ανακύκλωση **while** ως εξής :

```
Start;
while (Συνθήκη) {
     Ομάδα Εντολών
     Next;
}
```

Προσέξτε όμως την διαφορά συμπεριφοράς της **continue**.

Παράδειγμα. Ανακυκλώσεις **while**, **do while** και **for**

```
#include <stdio.h>
#include <stdlib.h>
main(){ int i;

     i = 5;
     while (i<5) { i++; } // ΔΕΝ εκτελείται

     i = 5;
     do { i++; } while(i<5); // μια φορά μόνο

     for (i=1;i<=10;i++){ // 1,2,3,4,5 μόνο
          printf("%d\n", i);
          if (i==5) break;
     }

     for (i=-5;i<=5;i++){ // όχι διαίρεση δια το μηδέν
          if (i==0) continue;
          printf("%f\n", 1.0/i);
     }

     for (i=1;i<10;i+=2)
          printf("%d\n", i); // τι τυπώνεται;

     for (i=10;i>0;i-=3)
          printf("%d\n", i); // τι τυπώνεται;

     system("pause");
}
```

Παράδειγμα. Γενικευμένα αθροίσματα και γινόμενα.

```
#include <stdio.h>
#include <stdlib.h>
main(){ int n,i; float s,p;

     printf("give n -> "); scanf("%d", &n);

     s = 0; // αρχική τιμή για το άθροισμα
     for(i=1;i<=n;i++) s = s + i; // s = 1 + 2 + ... + n

     p = 1; // αρχική τιμή για το γινόμενο
     for(i=1;i<=n;i++) p = p * i; // p = 1 * 2 * ... * n

     printf("s = %f p = %f\n", s, p);

     system("pause");
}
```

Παράδειγμα. Άθροισμα τιμών του πίνακα.

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

#define N 10

main(){ int i; float a, pin[N];

     ...

     a = 0; // αρχική τιμή για το άθροισμα

     for( i=0; i<N; i++ ) a += pin[i];

     printf(" a = %f \n", a );

     system("pause");
}
```