Δομές Ανακυκλώσεων

- Σε όλες τις γλώσσες προγραμματισμού, οι εντολές ανακυκλώσεων επιτρέπουν να επαναλαμβάνουμε ένα σύνολο εντολών, περισσότερες από μια φορές και μέχρι να επιτευχθεί μια ορισμένη συνθήκη
- Αυτή η συνθήκη μπορεί να έχει ορισθεί εκ των προτέρων ή να μην έχει οριστεί δηλαδή, να έχει ανοικτό τέλος
- Οι εντολές ανακυκλώσεων είναι:

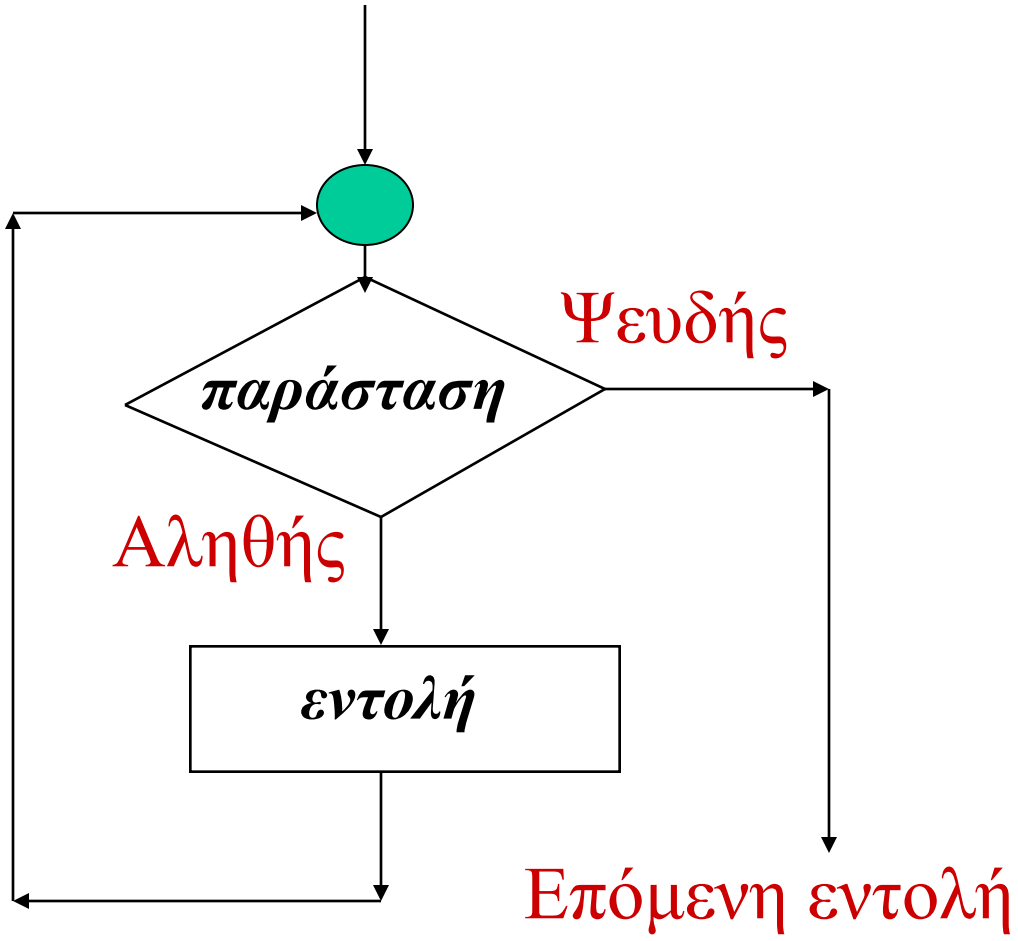
1. Η ανακύκλωση for
2. Η ανακύκλωση while
3. Η ανακύκλωση do-while
4. Η εντολή goto

Η ανακύκλωση while

- Ο βρόχος while λειτουργεί με τον εξής τρόπο:
- Ελέγχεται κατ' αρχήν η παράσταση μέσα στις παρενθέσεις.
- Αν είναι αληθής (μη-μηδενική) τότε εκτελείται το σώμα του βρόχου.
- Κατόπιν η παράσταση ελέγχεται ξανά και εφόσον είναι αληθής εκτελείται ξανά ο βρόχος.
- Όταν ο έλεγχος διαπιστώσει ότι η παράσταση είναι ψευδής ο βρόχος τερματίζεται και η εκτέλεση συνεχίζεται με την εντολή που ακολουθεί το βρόχο. Εκτός και αν δεν υπάρχουν στο πρόγραμμα άλλες εντολές οπότε τερματίζεται το πρόγραμμα.
- Οι περισσότερες εργασίες γίνεται μέσα στο σώμα του βρόχου

Η ανακύκλωση while

while (*παράσταση*)
εντολή;



Η ανακύκλωση while

- Το σώμα ενός βρόχου μπορεί να αποτελείται από μία ή περισσότερες εντολές κλεισμένες σε άγκιστρα ή από μια μοναδική εντολή χωρίς άγκιστρα:

```
while ( i < j )
    i= 2 *  i ;
```

- Και στις 2 περιπτώσεις θα γράφουμε πάντα τις εντολές που ελέγχονται από την while μια θέση στηλοθέτη πιο μέσα ή αλλιώς 4 κενά πιο μέσα ώστε να βλέπουμε με μια ματιά ποιες εντολές βρίσκονται μέσα στο βρόχο.
- Συνιστούμε να γράφετε μόνο μία εντολή σε κάθε γραμμή και να χρησιμοποιείτε κενά πριν και μετά από τους τελεστές για να αποσαφηνίσετε την ομαδοποίηση.
- Η θέση των αγκίστρων είναι λιγότερο σημαντική πρέπει να επιλέξετε το στυλ που σας αρέσει και να το χρησιμοποιείτε συστηματικά.

Παράδειγμα

```
#include <stdio.h>
/* εμφάνιση πίνακα Φαρενάιτ-Κελσίου
για θερμοκρασίες Φαρενάιτ = 0, 20, ..., 300 */
main()
{
   int fahr, celsius;
   int lower, upper, step;

   lower = 0;       /* κάτω όριο του πίνακα
                       θερμοκρασιών */
   upper = 300;     /* άνω όριο */
   step=20;         /* μέγεθος βήματος */

   fahr = lower;
   while (fahr <= upper) {
   celsius = 5 * (fahr-32) / 9;
   printf("%d\t%d\n", fahr, celsius);
   fahr = fahr + step;
   }
}
```

```
0   -17
20  -6
40  4
60  15
80  26
100         37
120         48
140         60
160         71
180         82
200         93
220         104
240         115
260         126
280         137
300         148
```

Η ανακύκλωση for

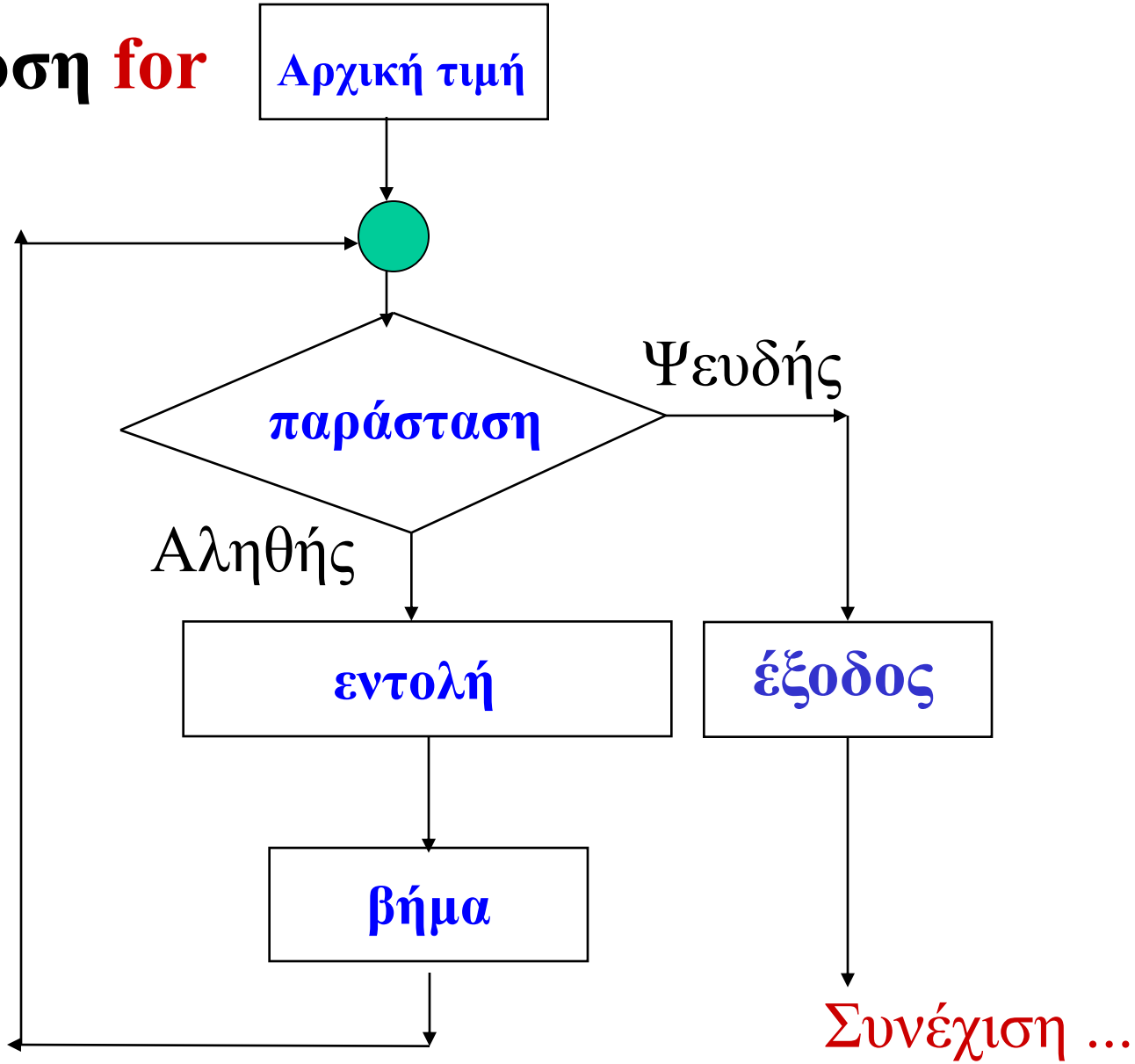
- Ο γενικός τύπος της ανακύκλωσης for για να επαναλάβει μία μόνον εντολή είναι:

```
for (αρχική τιμή; συνθήκη; βήμα)
        εντολή;
```

Π.χ.

```
for ( k = 1 ; k <=100 ; k++ )
    printf ("%d ", k) ;
```

Η ανακύκλωση for



Η ανακύκλωση for

- Για να επαναλάβει μία σειρά ομαδοποιημένων εντολών ο γενικός τύπος είναι:

```
for (αρχική τιμή; συνθήκη; βήμα) {
  εντολή 1;
  εντολή 2;
     .
  εντολή x;
}
```

Ισοδυναμία for με while

Η εντολή for:

for ($παρ_1$; $παρ_2$; $παρ_3$)
    εντολή

είναι ισοδύναμη με την:

$παρ_1$;
while ($παρ_2$) {
    εντολή
     $παρ_3$;
}

Με μόνη διαφορά τη συμπεριφορά της εντολής continue

Η ανακύκλωση while

```
/* Εμφάνιση των αριθμών από το 1 μέχρι και το 9 */
   int i;
   i = 1;
   while ( i < 10 ) {
       printf("%d\n", i);
       i++;
   }
```

```
/* Αναμονή μέχρι να πατηθεί ο χαρακτήρας A */
   wait_ for_ char( ) {
   char ch;
   ch=0;
   while ( ch ! = 'A')
       ch = getchar( );
    }
```

for αναλυτικά

- Γραμματικά τα 3 στοιχεία ενός βρόχου for είναι παραστάσεις.
- Συνήθως η $παρ_1$ και η $παρ_3$ είναι παραστάσεις απόδοσης τιμής ή κλήσεις συναρτήσεων ενώ η $παρ_2$ είναι παράσταση συσχέτισης. Οποιοδήποτε από τα 3 τμήματα μπορεί να παραλείπεται αν και τα ελληνικά ερωτηματικά πρέπει να υπάρχουν.
- Αν παραλειφθεί η $παρ_1$ ή η $παρ_3$ απλώς απορρίπτεται και στην ανάπτυξη της εντολής.
- Αν δεν υπάρχει ο έλεγχος δηλαδή η $παρ_2$ τότε θεωρείται μόνιμα αληθής και έτσι η κατασκευή:

```
for( ; ; ) {
. . .
}
```

είναι ένας ατέρμων βρόχος που διακόπτεται με άλλα μέσα όπως μια εντολή break ή μια εντολή return.

Προτίμηση για while?

- Η χρήση βρόχου **while** είναι κυρίως ζήτημα προσωπικής προτίμησης για παράδειγμα στο τμήμα κώδικα:

```
while ( ( c = getchar() ) = = ´ ´ || c = =  ´\n´ || c = = ´\t ‘) ;
/* παράβλεψη χαρακτήρων λευκού διαστήματος */
```

- Δεν υπάρχει απόδοση ή αποκατάσταση αρχικών τιμών σε μεταβλητές και έτσι ο βρόχος while αποτελεί πολύ φυσική λύση.

Προτίμηση για for

- Ο βρόχος **for** είναι προτιμότερος όταν έχουμε απλή απόδοση αρχικής τιμής και αύξηση επειδή διατηρεί συγκεντρωμένες τις εντολές ελέγχου του βρόχου στο πάνω μέρος του όπου φαίνονται εύκολα αυτό γίνεται προφανές στο τμήμα κώδικα

```
for ( i = 0 ; i < n ; i ++)
```

- πού είναι το ιδίωμα της C για την επεξεργασία των πρώτων στοιχείων ενός πίνακα

2ος τρόπος

```
#include <stdio.h>

 /* Εμφάνιση πίνακα Φαρενάιτ-Κελσίου */
 main()
 {
    int fahr;

    for (fahr = 0; fahr <= 300; fahr = fahr + 20)
    printf("%3d %6.1f\n", fahr, (5.0/9.0)*(fahr-32));
  }
```

```
  0  -17
 20  -6
 40   4
 60  15
 80  26
100  37
120  48
140  60
160  71
180  82
200  93
220 104
240 115
260 126
280 137
300 148
```

```
riza()
{
    int x,z;
    for (x = 100 ; x != 65 ;  x = x – 5 )
    {
        z = x * x ;
        printf ( "Το τετράγωνο του
                %d  =  %d ", x, z ) ;
     }
}
```

```
sign_on ( )
{
    int x ;
    for ( x = 0 ; x < 3 ; ++x )
    {
        printf ( ” Δώστε το password ") ;
        …….
     }
     if ( x < 3 ) hang_up ( );
     else printf ( ” Λάθος password ” ) ;
 }
```

```
x = 10 ;
for ( y = 10 ; y != x ; ++x )
{
    printf ( "%d", y ) ;
}
    printf ( "%d", y ) ;
```

```
char y;
int x;
for ( x = 0, y = 0 ; x + y < 10 ; ++x )
{
    y = getchar ( ) ;
    y = y + 'A’ ;
    printf ( "%c”, y ) ;
  }
```

do-while

- Οι βρόχοι for και while ελέγχουν τη συνθήκη τερματισμού στην κορυφή του βρόχου. Αντίθετα ο τρίτος βρόχος της C do-while, ελέγχει τη συνθήκη στο τέλος του, αφού ολοκληρώσει το κάθε πέρασμα από το σώμα του βρόχου. Το σώμα εκτελείται πάντοτε τουλάχιστον μία φορά.
- Η σύνταξη του βρόχου do είναι:

```
do
   εντολή
while (παράσταση) ;
```

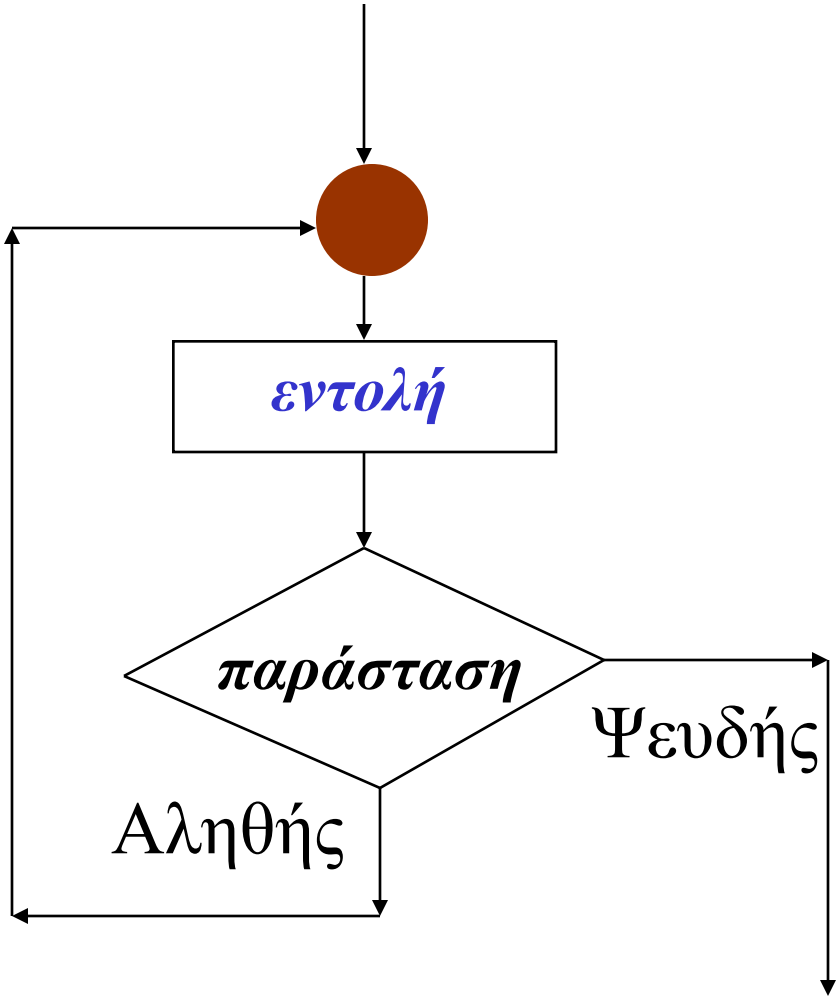
- Εκτελείται η εντολή και μετά υπολογίζεται η παράσταση. Αν είναι αληθής τότε η εντολή εκτελείται ξανά κ.ο.κ . Όταν η παράσταση γίνει ψευδής ο βρόχος τερματίζεται. Ο βρόχος do-while είναι ισοδύναμος με την εντολή repeat-until της pascal

Η ανακύκλωση do-while

do

εντολή

while (*παράσταση*) ;



Επόμενη εντολή

Η ανακύκλωση do-while

```
/* Αθροισμα των ζυγών αριθμών από το 2 μέχρι και το 10 */

x = 2;
sum = 0;
do
{
    sum += x;
    x += 2;
}
while ( x <= 10 );
printf("Άθροισμα= %d ", sum);
```

```
/* Ανάγνωση αριθμών μέχρι να δοθεί μια τιμή μικρότερη του 100 */

char s[80];
do
{
    num= getnum2(s,80 );
 }
while (num>99);
```

Break

- Μερικές φορές εξυπηρετεί να μπορούμε να βγούμε από ένα βρόχο χωρίς να εκτελέσουμε τον έλεγχο της αρχής ή του τέλους.
- Η εντολή break προσφέρει τη δυνατότητα πρόωρης εξόδου από τους βρόχους for, while, do που ακριβώς όπως και στην εντολή switch.
- Ή εντολή break προκαλεί άμεση έξοδο από τον πιο εσωτερικό βρόχο ή την εντολή switch.

Παράδειγμα

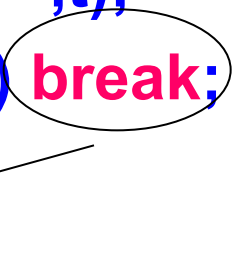
- Η strlen επιστρέφει το μήκος ενός αριθμητικού.
- Ο βρόχος for ξεκινά από το τέλος και σαρώνει το αλφαριθμητικό προς τα πίσω αναζητώντας τον πρώτο χαρακτήρα που δεν είναι κενό, στηλοθέτης ή χαρακτήρα αλλαγής γραμμής.
- Ο βρόχος διακόπτεται όταν βρεθεί ένας τέτοιος χαρακτήρας ή όταν το n γίνει αρνητικό (δηλαδή όταν θα έχει σαρωθεί ολόκληρο το αλφαριθμητικό.)

```
/*trim: αφαιρεί κενά, στηλοθέτες, και χαρακτήρες
          αλλαγής γραμμής από το τέλος αλφαριθμητικών */
   int trim(char s[])
   {
       int n;

       for (n = strlen(s)-1; n>=0; n--)
            if (s[n] != ' ' && s[n] != '\t' && s[n] != '\n')
                break;
       s[n+1]='\0';
       return n;
}
```

Έξοδος των ανακυκλώσεων (εντολή break)

```
main ( )
    {
        int t;
      for(t=0;t<100;t++)
      {
       printf("%d ",t);
       if( t = =10 ) break;
      }
  . . . . .
  }
```



Continue

- Η εντολή **continue** σχετίζεται με τη break αλλά χρησιμοποιείται σπανιότερα.
- Προκαλεί την έναρξη της επόμενης επανάληψης του βρόχου for, while, do που την περιέχει.
- Στους βρόχους while και do αυτό σημαίνει ότι εκτελείται αμέσως το τμήμα ελέγχου ενώ στο βρόχο for ο έλεγχος περνάει στο βήμα αύξησης.
- Η εντολή continue εμφανίζεται μόνο σε βρόχους όχι όμως και στην εντολή switch.
- Μια εντολή continue σε μια switch που βρίσκεται μέσα σε βρόχο προκαλεί την επόμενη επανάληψη του βρόχου.
- Ως παράδειγμα το επόμενο απόσπασμα επεξεργάζεται μόνο τα μη αρνητικά στοιχεία του πίνακα α ενώ οι αρνητικές τιμές παραβλέπονται.

```
for ( i = 0 ; i < n; i++ ) {
    if ( a [i] < 0 ) /* παράβλεψη αρνητικών στοιχείων */
            continue;
   . . .  /* επεξεργασία θετικών στοιχείων */
}
```

Συνέχιση των ανακυκλώσεων (εντολή continue)

```
do  {
  x=getnum(s,80);
  if(x<0) continue;
  printf("%d ", x);
      } while(x!=100);
```

Τερματισμός του προγράμματος (συνάρτηση exit ( ))

- Η συνάρτηση exit( ) προκαλεί άμεσο τερματισμό του προγράμματος και επιστροφή στο λειτουργικό σύστημα

π.χ.

```
main( )
   {
      if ( !color_card( ) ) exit(1);
        . . . . .
    }
```

Goto

- Η C διαθέτει την απεριόριστα κακοποιημένη εντολή **goto** καθώς και ετικέτες για τις διακλαδώσεις της.
- Τυπικά η goto δεν χρειάζεται ποτέ και όντως είναι σχεδόν πάντα εύκολο να γράψουμε τον κώδικα χωρίς αυτή πάντως υπάρχουν λίγες περιπτώσεις που η goto μπορεί να έχει θέση.
- Η πιο συνηθισμένη περίπτωση είναι η έξοδος από την επεξεργασία σε κάποια πολύ βαθιά ένθετη δομή, όπως η ταυτόχρονη έξοδος από 2 ή περισσότερους βρόχους. Η εντολή break δε μπορεί να χρησιμοποιηθεί άμεσα επειδή προκαλεί την έξοδο μόνο από τον πιο εσωτερικό βρόχο.

Παράδειγμα

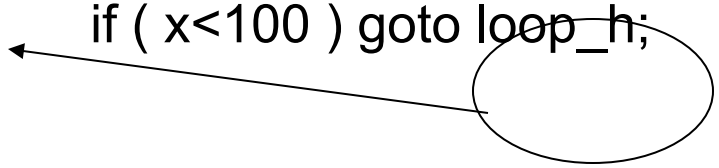
Έτσι:

```
for (  …  )
   for (  …  ) {
        ….
        if ( disaster ) \* αν συμβεί καταστροφή *\
              goto error;
   }
. . .
error:
   τακτοποιεί την κατάσταση
```

Ετικέτες και goto

Μία ανακύκλωση από το 1 έως το 100 θα μπορούσε να γραφτεί με τη χρησιμοποίηση του goto ως εξής:

```
            x=1;
loop_h  :   x++;
            if ( x<100 ) goto loop_h;
```



Η goto δεν πρέπει να χρησιμοποιείται συχνά, αν όχι καθόλου

Αν είναι δύσκολο να αλλάξουμε τη δομή ενός υπάρχοντος προγράμματος ή

αν έχει σημασία η ταχύτητα εκτέλεσης,

μόνο τότε, μπορούμε να χρησιμοποιήσουμε την εντολή goto

Ασκήσεις

Δείξτε τρεις τρόπους γραφής της συνάρτησης count( ) οι οποίοι απλώς εμφανίζουν τους αριθμούς από το 1 μέχρι το 100 στην οθόνη

(Μη χρησιμοποιείτε την εντολή goto).

Λύσεις

```
count1( )
 {
    int  t;
    for ( t = 1; t < 101 ;
++t )
    printf( "%d”,t ) ;
  }
```

```
count2( )
{
 int t ;
 t =1 ;
 while ( t < 101 )
 printf( "%d", t++);
  }
```

```
count3( )
{
    int t;
    t=1;
  do
 {
   printf("%d",t);
        t++;
  } while( t<101) ;
 }
```

Να αναπτυχθεί ένα πρόγραμμα το οποίο θα πρέπει να διαβάζει από το πληκτρολόγιο έναν ακέραιο αριθμό έστω K, ο οποίος θα πρέπει να ελέγχεται ώστε να ισχύει η σχέση

$$0 \leq K \leq 1000$$

Στη συνέχεια να υπολογίζει το άθροισμα

$$1+2+...+K.$$

```
(k<0 || k>1000)
```

```
while (k<0 || k>1000)
```

```
#include <stdio.h>
 main()
{ int i, k, sumk;  // Δήλωση των μεταβλητών
  do  {
printf("Δώστε ένα θετικό αριθμό K μικρότερο του 1000 \n");
printf("για να υπολογιστεί ");
printf("το άθροισμα 1+2+ . .+K \n K = ");
     scanf("%d", &k);  // Ανάγνωση της τιμής
 } while (k<0 || k>1000);  // Έλεγχος της πληκτρολόγησης
 sumk=0;   // Αρχική τιμή του αθροίσματος
 for (i=1; i<=k; i++)
   sumk=sumk+i; // Υπολογισμός του διαδοχικού αθροίσμ.
 printf("Άθροισμα των αριθμών = %d\n", sumk);
 }
```

Άσκηση (TEST)

Να γράψετε ένα πρόγραμμα το οποίο όταν ξεκινά την εκτέλεσή του, θα ζητά υποχρεωτικά ένα θετικό ακέραιο αριθμό μικρότερο από το 10 και μεγαλύτερο από το 4, από το πληκτρολόγιο και θα εμφανίζει στην οθόνη το μήνυμα **ΕΜΦΑΝΙΣΗ ΤΙΜΩΝ** τόσες φορές όσες εκφράζει η τιμή του ακέραιου αριθμού που έχει πληκτρολογηθεί.

Το πρόγραμμα θα πρέπει να επαναλαμβάνεται αυτόματα και να ζητά εκ νέου ένα νέο αριθμό μικρότερο από το 10 και μεγαλύτερο από το 4 και να σταματά τις επαναλήψεις (τέλος του προγράμματος) όταν ο εισαγόμενος αριθμός είναι η τιμή **-999**, με την εμφάνιση και ενός μηνύματος που θα δείχνει πόσες φορές επαναλήφθηκε η όλη διαδικασία εισαγωγής τιμής από το πληκτρολόγιο.