Προτάσεις επανάληψης – βρόχοι

Προτάσεις επανάληψης - γενικά

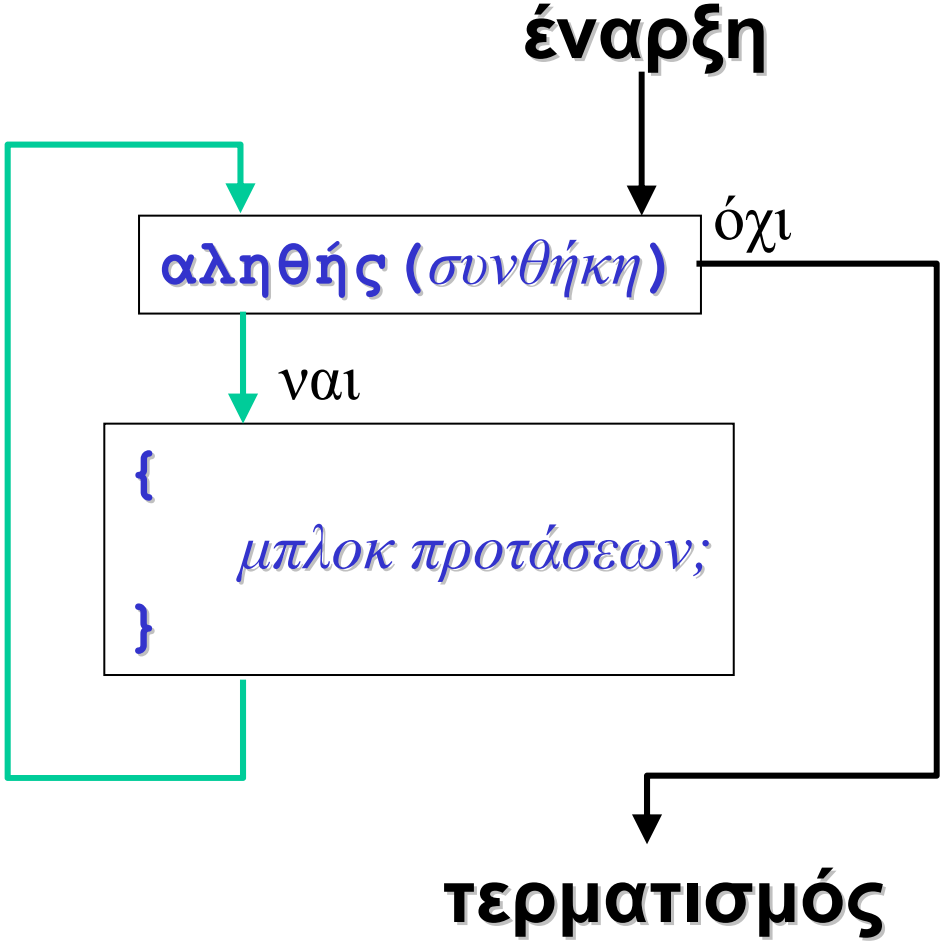
- **Οι προτάσεις επανάληψης επαναλαμβάνουν ένα μπλοκ προτάσεων είτε για όσες φορές το επιθυμούμε είτε έως ότου πληρωθεί μία συνθήκη τερματισμού.**

- **Η πλήρωση του κριτηρίου τερματισμού (terminating condition) οδηγεί στην περάτωση του βρόχου (loop).**

- **Εάν δεν υπάρχει συγκεκριμένος αριθμός επαναλήψεων ή συνθήκη τερματισμού, ο βρόχος θα εκτελείται αενάως, οδηγώντας σε σφάλμα.**

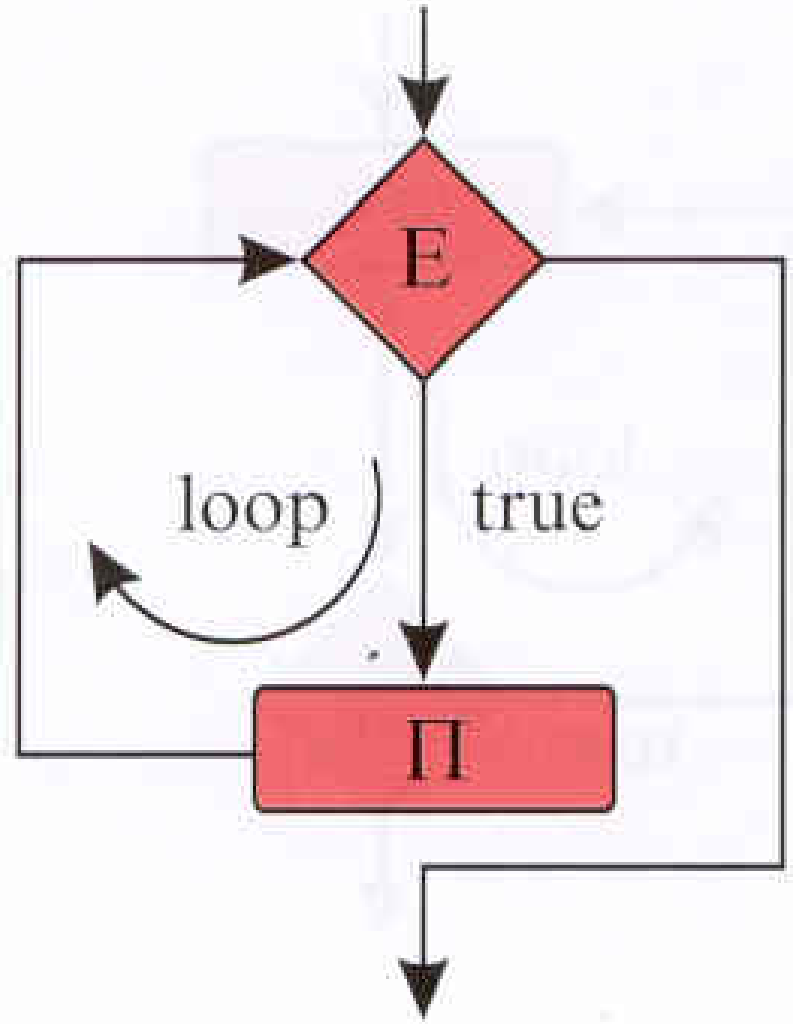
while - do

1) **Βρόχος με συνθήκη εισόδου (pre-test loop):**
   - **α) οδηγούμενος από γεγονός**
   - **β) οδηγούμενος από μετρητή**



while - do

while E do Π

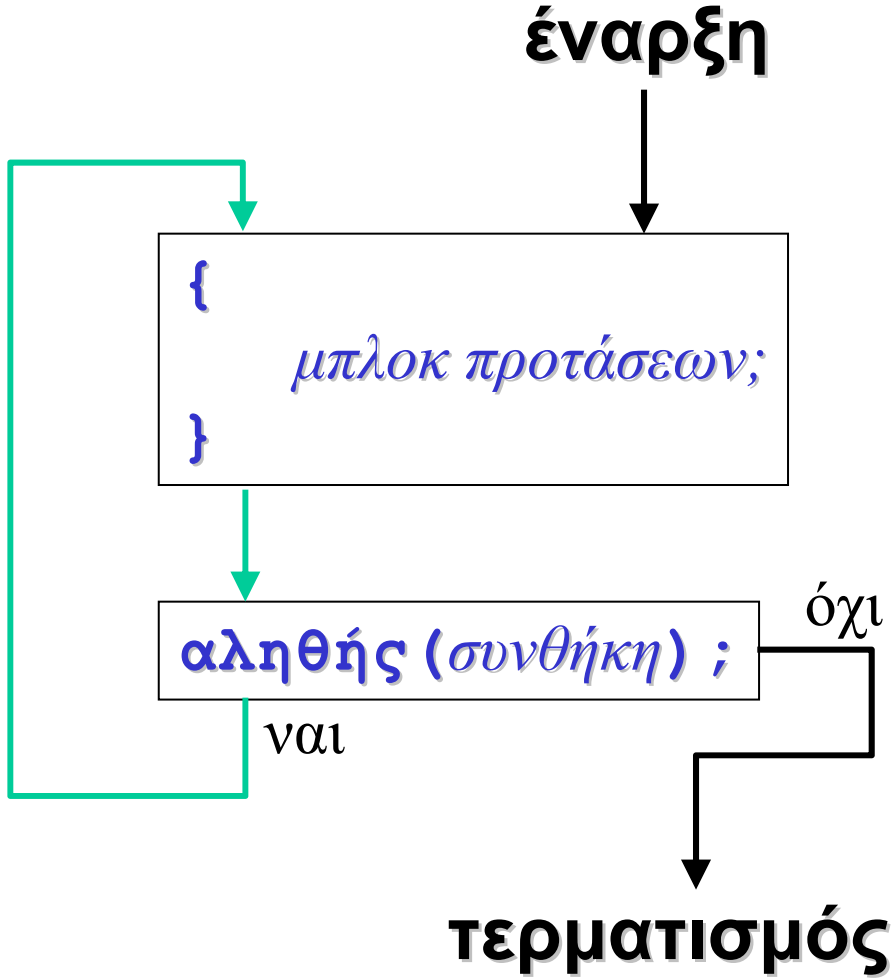


do - while

2) **Βρόχος με συνθήκη εξόδου** (post-test loop):

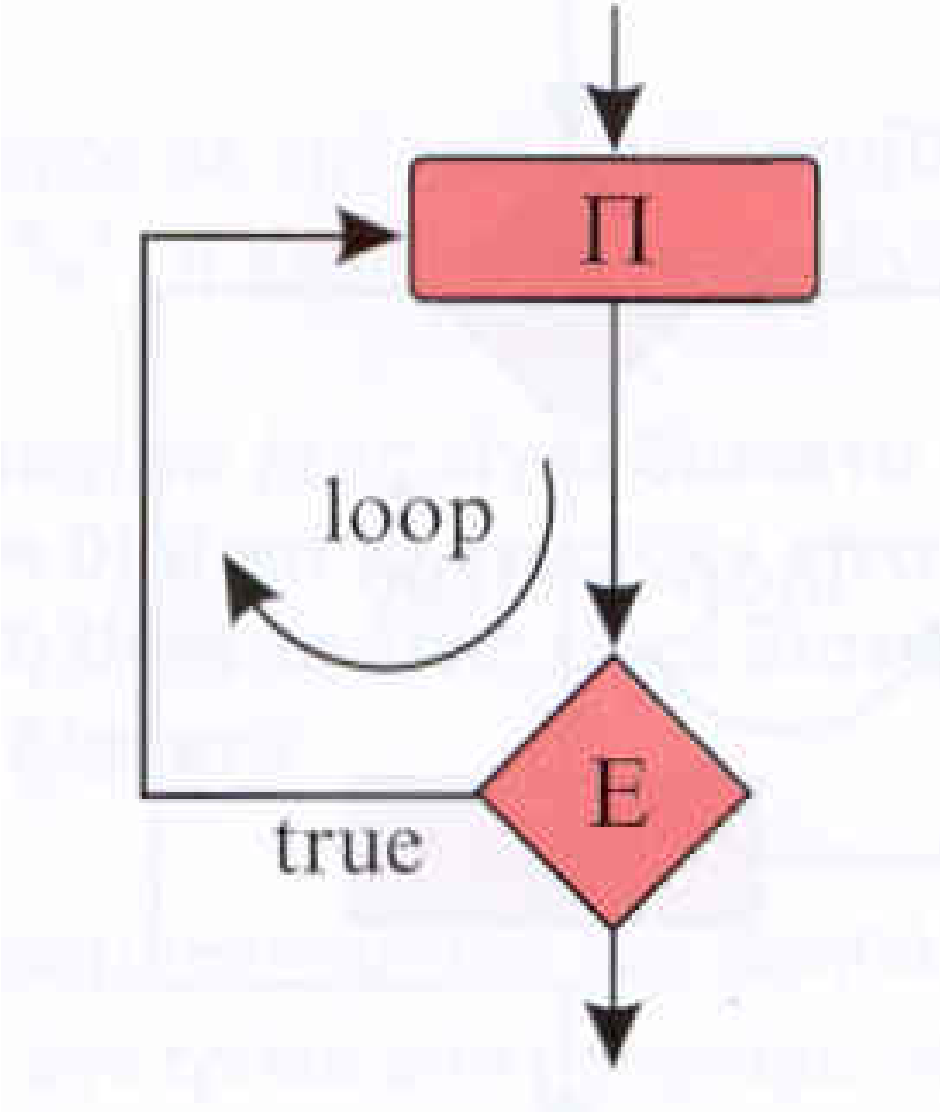
α) οδηγούμενος από γεγονός

β) οδηγούμενος από μετρητή



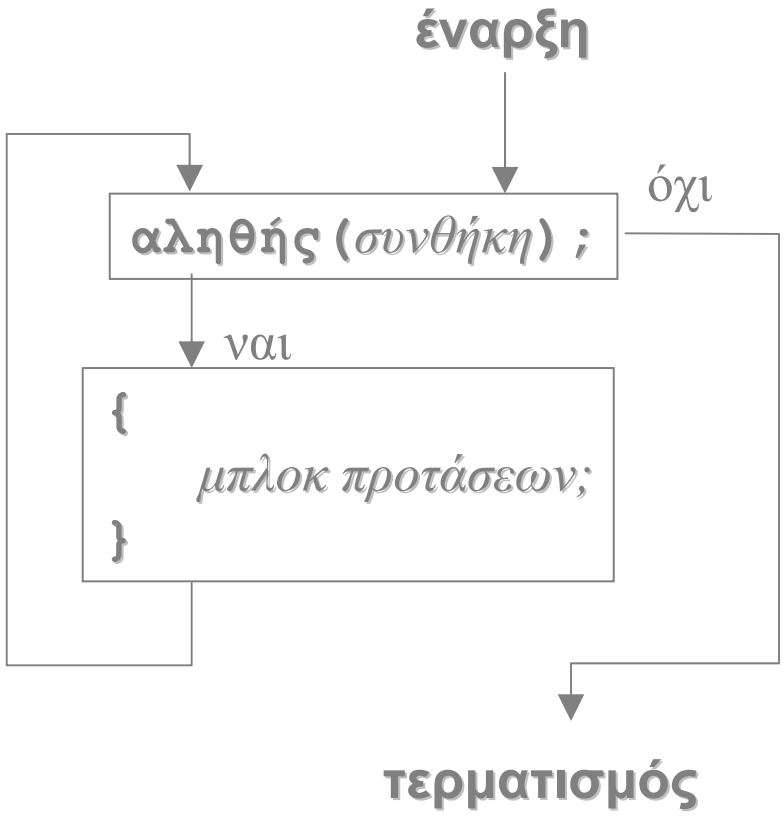
do - while

do Π while E



Βρόχος με συνθήκη εισόδου στη C, οδηγούμενος από το γεγονός: while

```
while (συνθήκη)
{
   προτάσεις στις οποίες
   αλλάζει η συνθήκη;
}
```



Βρόχος με συνθήκη εισόδου στη C, οδηγούμενος από το γεγονός: while

Η λειτουργία της πρότασης επανάληψης *while* μπορεί να μορφοποιηθεί σε δομημένα Ελληνικά ως εξής:

Έλεγξε τη συνθήκη.

Εάν είναι αληθής

Προχώρησε στις προτάσεις

Ξεκίνησε από την αρχή

Αλλιώς σταμάτησε

Βρόχος με συνθήκη εισόδου στη C, οδηγούμενος από το γεγονός: while

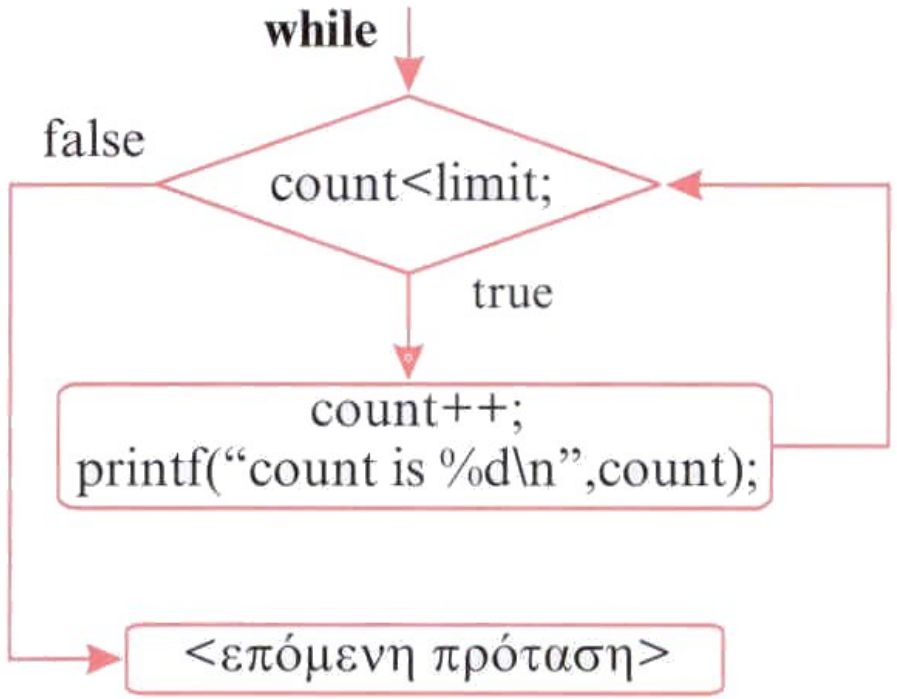
- **Ο βρόχος *while* είναι κατάλληλος στις περιπτώσεις που δεν είναι γνωστός εκ των προτέρων ο αριθμός των επαναλήψεων. Εκτελείται καθόσον η συνθήκη παραμένει αληθής. Όταν η συνθήκη καταστεί ψευδής, ο έλεγχος του προγράμματος παρακάμπτει το περιεχόμενο του βρόχου και προχωρά στην επόμενη εντολή.**

- **Θα πρέπει να σημειωθεί ότι εάν το σώμα του βρόχου αποτελείται από μία πρόταση, δεν απαιτούνται {}. Ωστόσο προτείνεται η χρήση των αγκίστρων σε κάθε περίπτωση, ανεξάρτητα από τον αριθμό των προτάσεων που απαρτίζουν το σώμα του βρόχου.**

Παράδειγμα 1:

```
int count=30;
int limit=40;
while (count<limit)
{
  count++;
  printf("count is %d\n",count);
}
<επόμενη πρόταση>;
```

/*Εάν αρχικά η count=40, ο βρόχος δε θα εκτελείτο ούτε μία φορά.*/

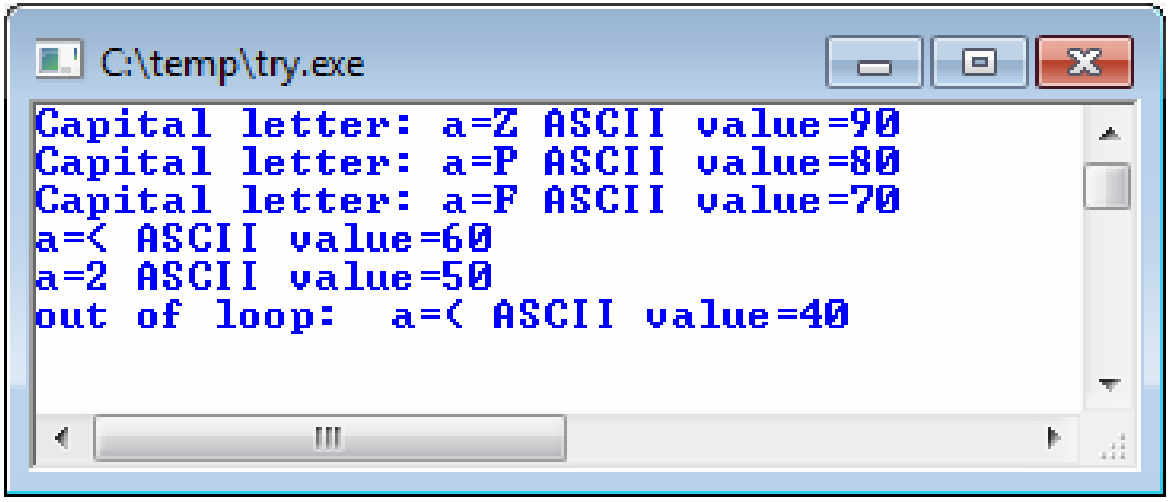


Άσκηση: *Δίνονται οι παρακάτω δύο προτάσεις:* ***(α)*** **while (++count<12) Π1** *και* ***(β)*** **while (count++<12) Π1**. *Να περιγραφεί ο τρόπος με τον οποίο ο υπολογιστής τις εκτελεί, εντοπίζοντας τη διαφορά τους, εάν υπάρχει.*

Λύση: *Υπάρχει διαφορά μεταξύ των προτάσεων κι αυτή εντοπίζεται στον αριθμό επαναλήψεων. Η* ***(α)*** *χρησιμοποιεί την προθεματική σημειογραφία ενώ η* ***(β)*** *τη μεταθεματική. Στην* ***(α)*** *πρόταση αυξάνεται πρώτα η τιμή της* **count** *και η νέα τιμή της συγκρίνεται με το* **12**, *ενώ στη* ***(β)*** *πρώτα συγκρίνεται η τιμή της* **count** *με το* **12** *και στη συνέχεια αυξάνεται η τιμή της. Αυτό σημαίνει πως η πρόταση* **Π1** *θα εκτελεσθεί μία φορά παραπάνω στην περίπτωση* ***(β)***.

Παράδειγμα 2:

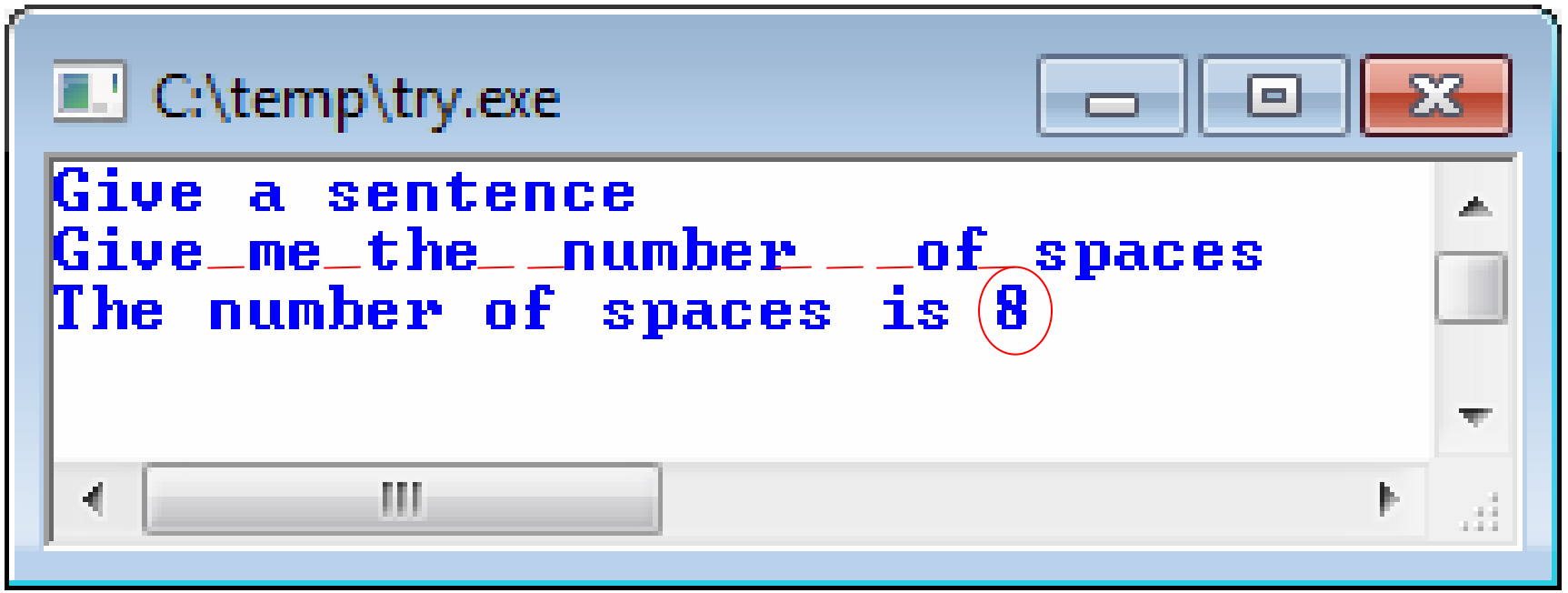
```
#include <stdio.h>
void main ()
{
  char a='Z';

  while(a>40)
  {
    if (a>'A')
    { printf("Capital letter:\t"); }
    printf( "a=%c ASCII value=%d\n",a,a );
    a=a-10;
   }
   printf( "out of loop:  a=%c ASCII value=%d\n",a,a );
}
```



Παράδειγμα 3: **Να γραφεί πρόγραμμα που να διαβάζει μία σειρά χαρακτήρων από την είσοδο, να μετρά τα κενά και να τυπώνει τον αριθμό τους.**

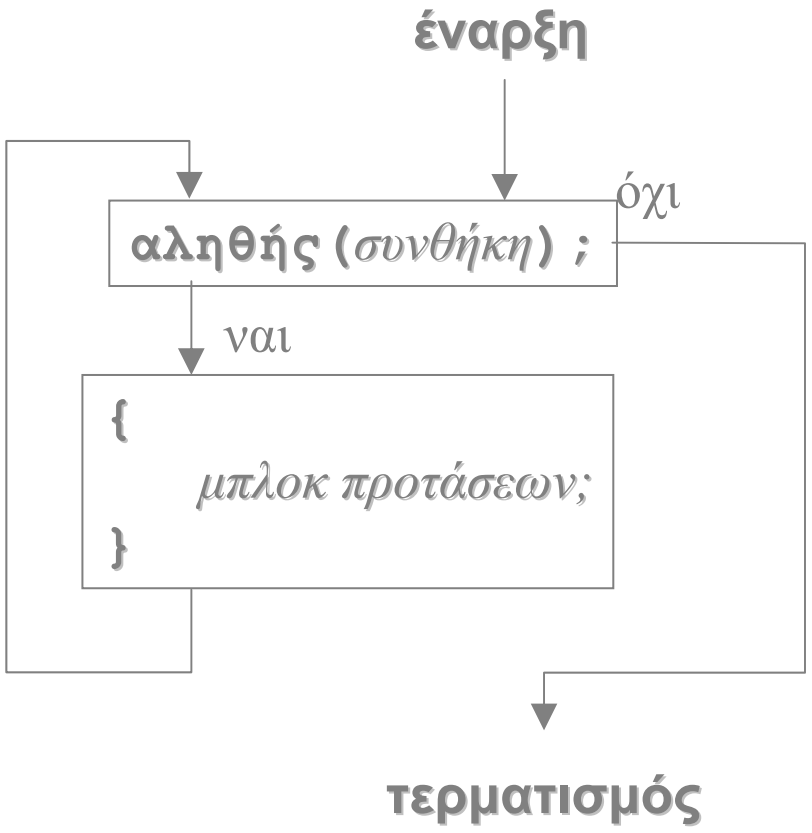
```
#include <stdio.h>
void main()  {
  int num_spaces=0;   char ch;
  printf( “Give a sentence\n” );
  ch=getchar();
  while (ch!=‘\n’)
  {
    if (ch = = ‘ ’) num_spaces++;
    ch=getchar();
  }
  printf(“The number of spaces is %d\n”,num_spaces);
}
```

Αποτελέσματα:



Βρόχος με συνθήκη εισόδου στη C, οδηγούμενος από μετρητή: for

```
for (αρχική; συνθήκη; μετρητής)
{
  προτάσεις;
}
```



Βρόχος με συνθήκη εισόδου στη C, οδηγούμενος από μετρητή: for

Η λειτουργία της πρότασης επανάληψης *for* μπορεί να μορφοποιηθεί σε δομημένα Ελληνικά ως εξής:

Αρχικοποίησε το μετρητή

Έλεγξε τη συνθήκη

Εάν είναι αληθής

    Εκτέλεσε τις προτάσεις

    Ενημέρωσε το μετρητή

    Επάνελθε στον έλεγχο της συνθήκης

Αλλιώς ενημέρωσε το μετρητή και σταμάτησε

Βρόχος με συνθήκη εισόδου στη C, οδηγούμενος από μετρητή: for

Ο βρόχος *for* στη γλώσσα C παρέχει μεγάλη ευελιξία καθώς οι εκφράσεις μέσα στις παρενθέσεις μπορούν να έχουν πολλές παραλλαγές:

- Μπορεί να χρησιμοποιηθεί ο τελεστής μείωσης για μέτρηση προς τα κάτω:

```
for (n=10; n>0; n- -) printf( "n=%d\n",n );
```

- Το βήμα καθορίζεται από το χρήστη:

```
for (n=0; n<60; n=n+13) printf( "n=%d\n",n );
```

- Ο μετρητής μπορεί να αυξάνει κατά γεωμετρική πρόοδο:

```
for (n=2; n<60.0; n=1.2*n) printf("n=%f\n",n );
```

Βρόχος με συνθήκη εισόδου στη C, οδηγούμενος από μετρητή: for

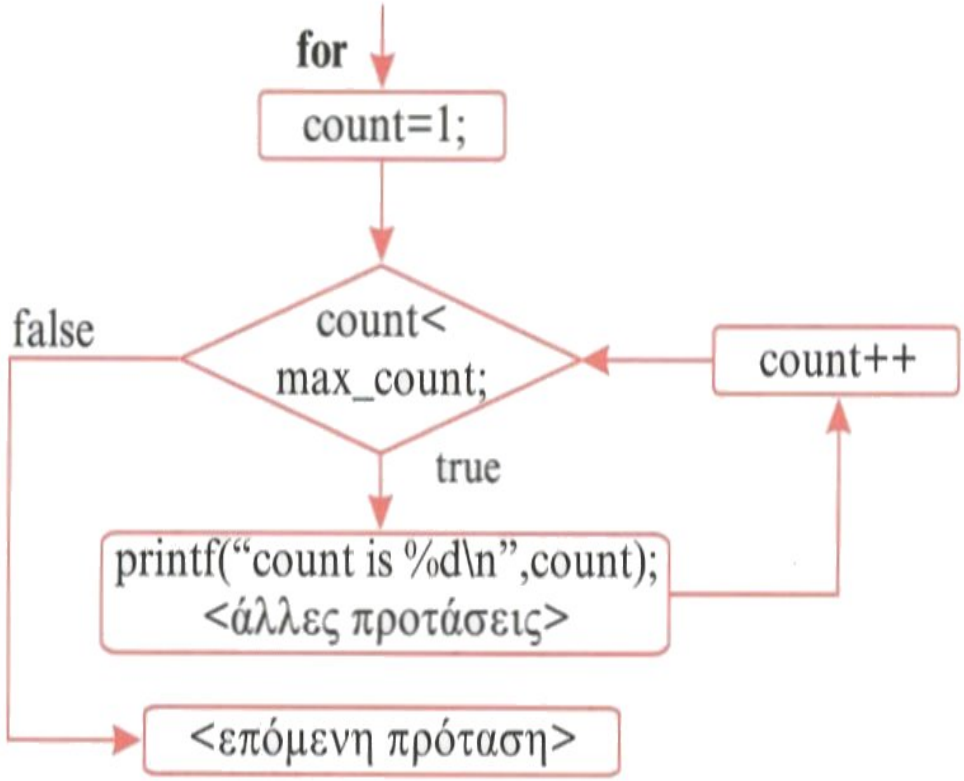
- Χρησιμοποιώντας την ιδιότητα ότι κάθε χαρακτήρας του κώδικα ASCII έχει μία ακέραια τιμή, ο μετρητής μπορεί να είναι μεταβλητή χαρακτήρα. Το παρακάτω τμήμα κώδικα θα τυπώνει τους χαρακτήρες από το 'a' έως το 'z' μαζί με τον ASCII κωδικό τους:

```
for (n='a'; n<'z'; n++)
        printf( "n=%c, the ASCII value is %d\n",n,n );
```

- Θα πρέπει να σημειωθεί ότι εάν το σώμα του βρόχου αποτελείται από μία πρόταση, δεν απαιτούνται {}. Ωστόσο προτείνεται η χρήση των αγκίστρων σε κάθε περίπτωση, ανεξάρτητα από τον αριθμό των προτάσεων που απαρτίζουν το σώμα του βρόχου.

Παράδειγμα 1:

```
int count, max_count=30;
for
(count=1;count<max_count;count+
+)
{
  printf("count is %d\n",count);
  <άλλες προτάσεις>;
}
<επόμενη πρόταση>;
```



Παράδειγμα 2: **Να γραφεί πρόγραμμα που να διαβάζει από την είσοδο ένα αλφαριθμητικό και το τυπώνει *times* φορές.**

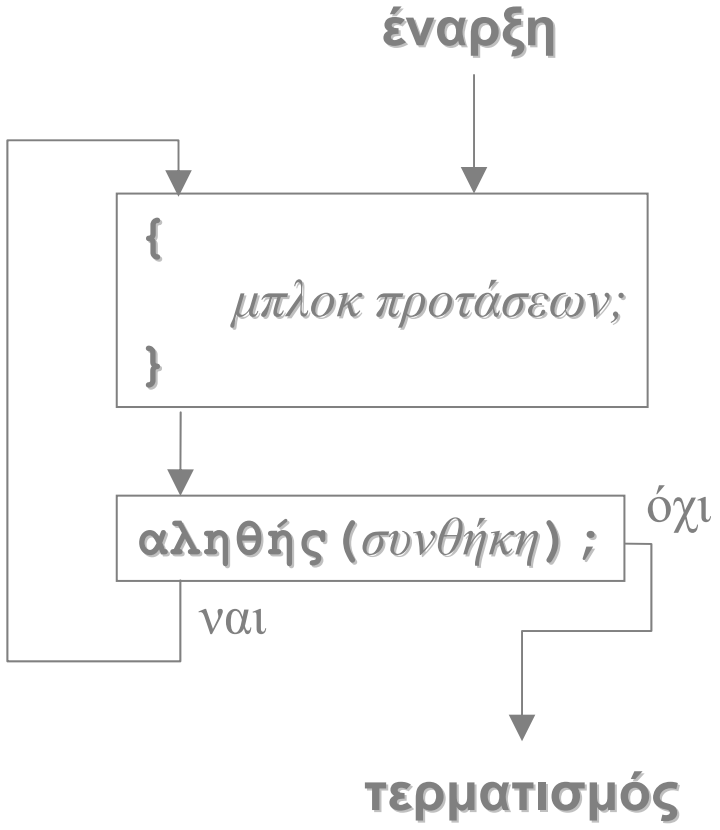
```
#include <stdio.h>
#define MAX_CHAR 80
void main()
{
  char str[MAX_CHAR];
  int i,times;
  printf( "\nEnter a string:" );
  scanf( "%s",str );
  printf( "\nEnter the number of repetitions:" );
  scanf( "%d",&times );
  for (i=0; i<times; i++) printf( "%s\n",str );
}
```

Δεν απαιτείται & για τους πίνακες χαρακτήρων

Βρόχος με συνθήκη εξόδου στη C, οδηγούμενος από γεγονός/μετρητή: do{}while()

```
do
{
    προτάσεις στις οποίες αλλάζει η
    συνθήκη;
}
while (συνθήκη);
```

// εναλλακτικά η while μπορεί να τοποθετηθεί αμέσως μετά το }



Βρόχος με συνθήκη εξόδου στη C, οδηγούμενος από γεγονός/μετρητή: do{}while()

Η λειτουργία της πρότασης επανάληψης *do-while* μπορεί να μορφοποιηθεί σε δομημένα Ελληνικά ως εξής:

Εκτέλεσε τις προτάσεις

Έλεγξε τη συνθήκη

Εάν είναι αληθής

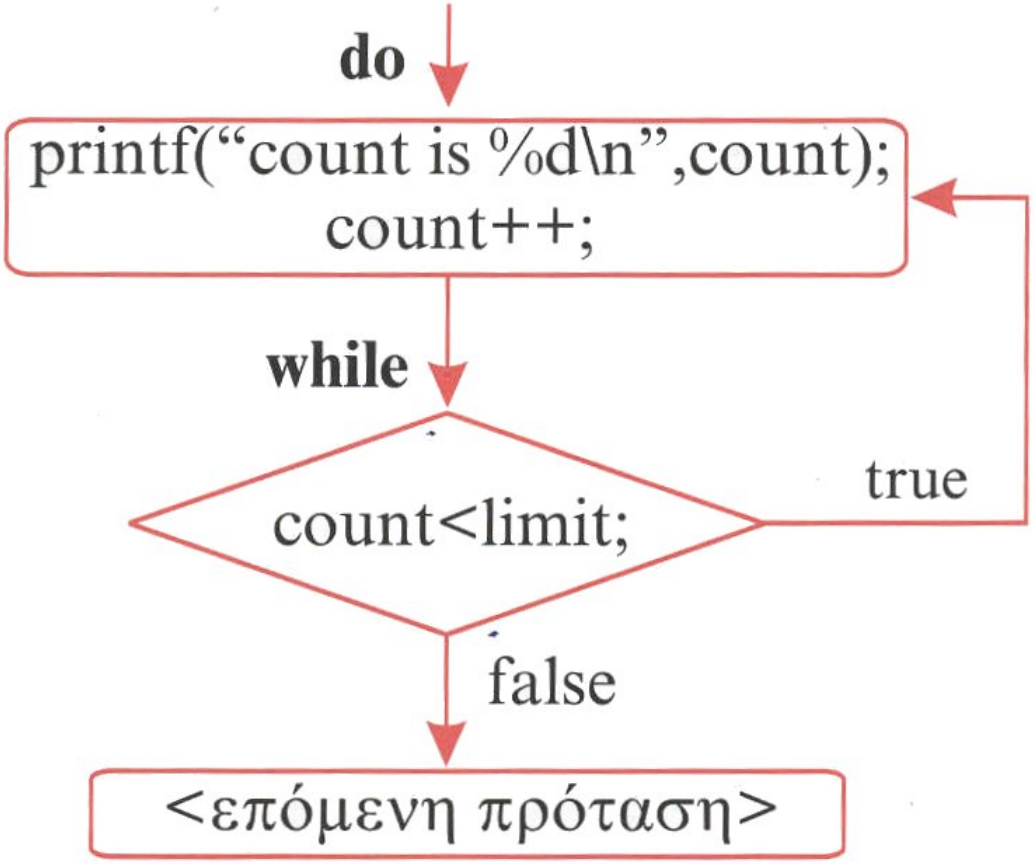
    Ξεκίνησε από την αρχή

Αλλιώς σταμάτησε

Παράδειγμα 1:

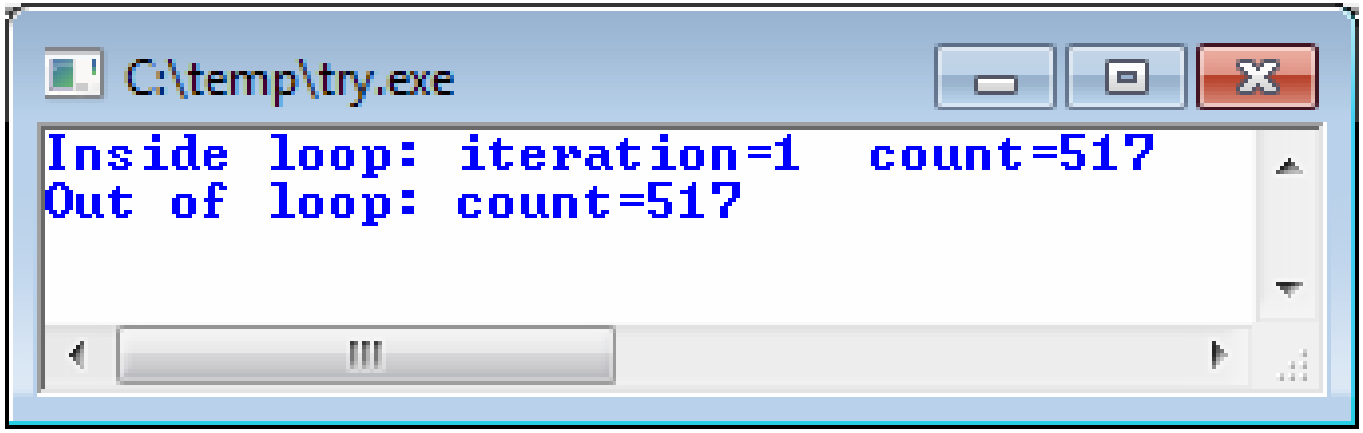
```
int count=30;
int limit=40;
do
{
  count++;
  printf("count is %d\n",count);
}
while (count<limit);
<επόμενη πρόταση>;
```

/*Το σώμα του βρόχου εκτελείται μία φορά έστω κι αν αρχικά count>=limit.*/



Παράδειγμα 2:

```
#include <stdio.h>
main()  {
  int iteration=0, count=516, limit=40;
  do   {
    count++;
    iteration++;
    printf("Inside loop: iteration=%d   count=%d\n",iteration,count);
  } while (count<limit);
  printf("Out of loop: count=%d\n",count);
}
```



Ένθετοι βρόχοι (nested loops)

Φώλιασμα (nesting): Τοποθέτηση ενός βρόχου μέσα σε άλλον. Ο εσωτερικός βρόχος είναι μία *πρόταση* μέσα στον εξωτερικό.

```
printf("\n");
for (i=0; i<4; i++)
{
   for (j=0; j<3; j++)
   {
      printf(" %d.%d ", i, j);
   }
   printf("\n");
}
```

Αποτέλεσμα:

```
>
 0,0   0,1   0,2
 1,0   1,1   1,2
 2,0   2,1   2,2
 3,0   3,1   3,2
>
```

Παράδειγμα 1: **Στο πρόγραμμα που ακολουθεί δημιουργούνται ένθετοι βρόχοι, όπου χρησιμοποιούνται και τα τρία είδη προτάσεων επανάληψης:**

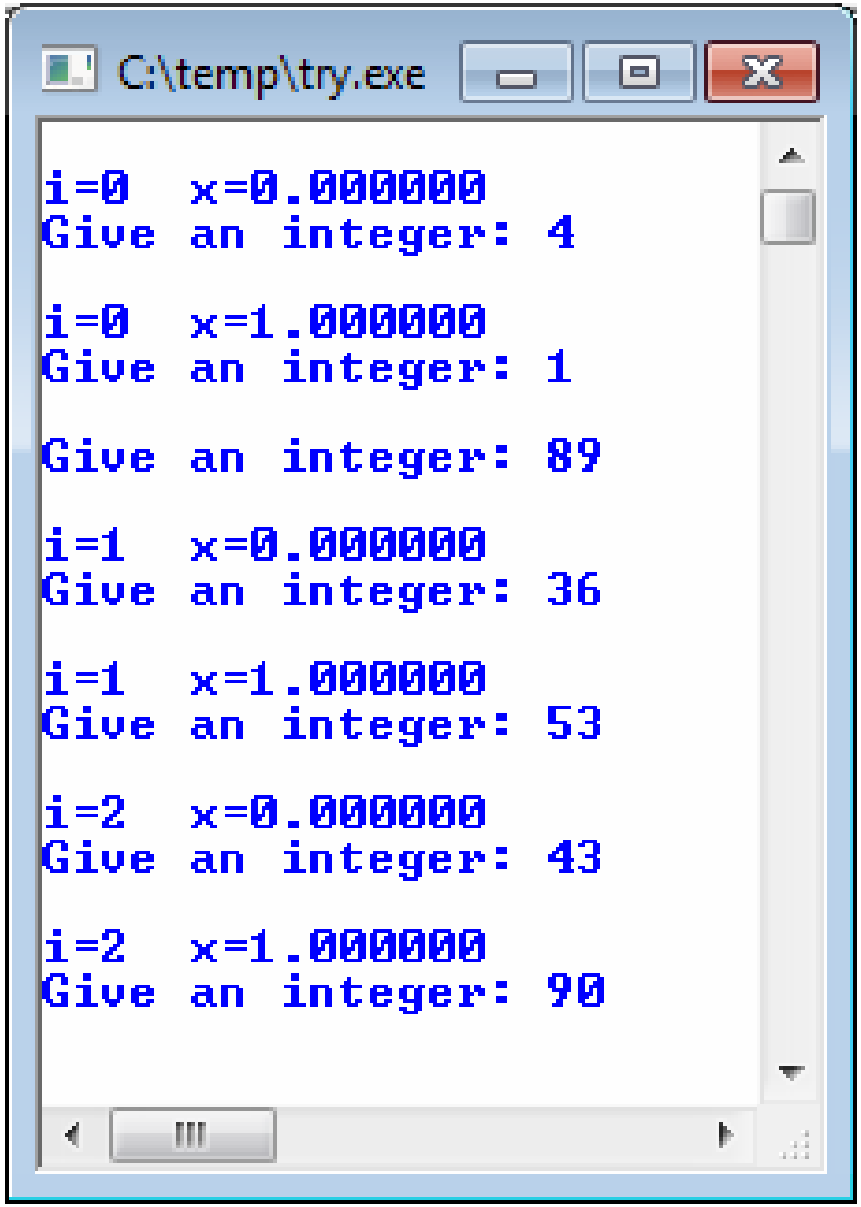
```
#include <stdio.h>
void main()  {
  int i,j;
  float x;
  for (i=0;i<=2;i++)  {
    x=0.0;
    while( x<2) {
      printf( "\ni=%d  x=%f",i,x );
      x++;
      do  {
        printf( "\nGive an integer: " );    scanf( "%d",&j );
      } while (j<4);
    }    // τέλος της while
  }    // τέλος της for
}    // τέλος της main
```

Εξωτερικός βρόχος

Πρώτο επίπεδο ένθεσης

Δεύτερο επίπεδο ένθεσης

Αποτελέσματα:

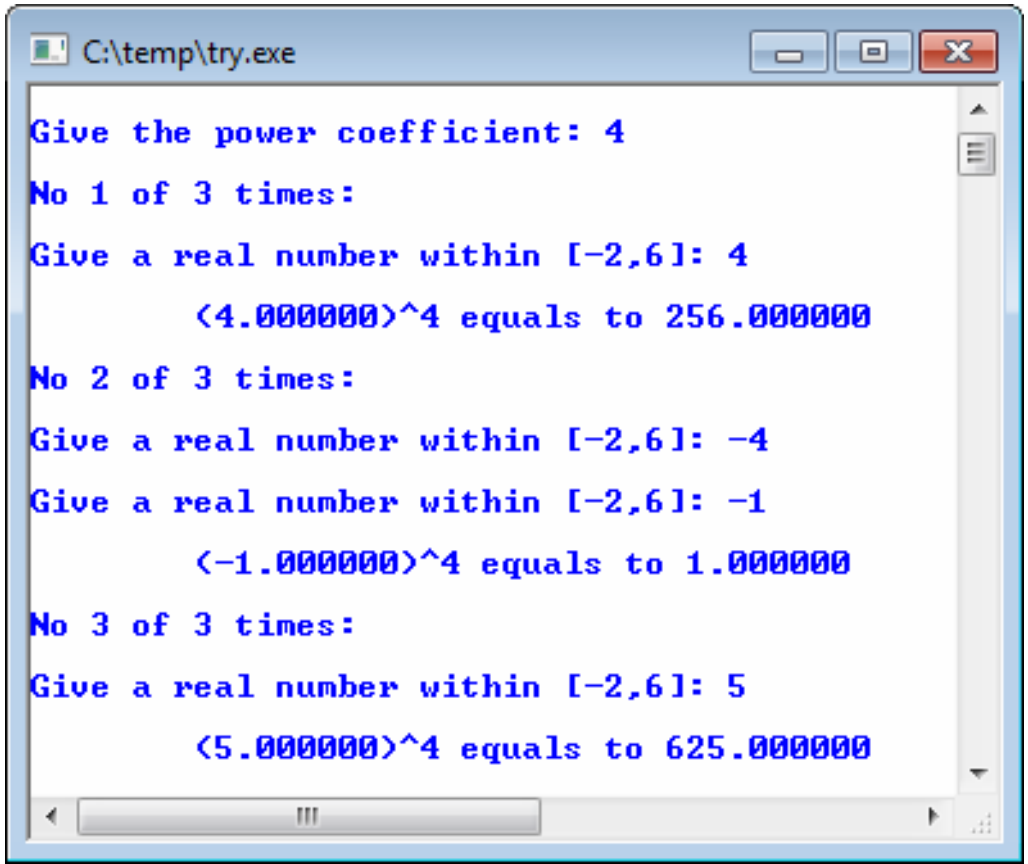


Παράδειγμα 2: **Να καταστρωθεί πρόγραμμα που θα δέχεται διαδοχικά N αριθμούς, κάθε φορά θα ζητά εκ νέου τον αριθμό εφόσον αυτός δεν υπάγεται σε συγκεκριμένο και δοθέν διάστημα [A,B]. Αρχικά ο χρήστης θα δίνει έναν ακέραιο για εκθέτη δύναμης. Το πρόγραμμα θα υψώνει κάθε φορά το δοθέντα αριθμό στη δοθείσα δύναμη και θα τυπώνει το αποτέλεσμα.**

```
#include <stdio.h>
#include <math.h>
#define N 3
#define A -2
#define B 6
#define MY_ZERO 1E-6
main()  {
  int i,j,pow_coef;
  float x,y;
  printf("\nGive the power coefficient: ");  scanf("%d",&pow_coef);
```

```
for (i=0;i<N;i++)  {
   printf("\nNo %d of %d times:\n",i+1,N);
   do  {
     printf("\nGive a real number within [%d,%d]: ",A,B);
     scanf("%f",&x);
   } while ((x<A) || (x>B));
   if (pow_coef = = 0) y=1.0;
   else if (fabs(x)<MY_ZERO)  {
     if (pow_coef>0) y=0.0;
     else  {
       printf("\n\nERROR!! The result can't be defined\n\n");     getch();
       break;
        }
   }  // end of internal ELSE-IF
   else  {
     y=x;
```

```
    for (j=1;j<=(abs(pow_coef)-1);j++)  {
      y=y*x;
    }
    if (pow_coef<0) y=1/y;
  }   // end of external ELSE-IF
  printf("\n\t(%f)^%d equals to %f\n",x,pow_coef,y);
 }  // end of FOR(i)
}
```



while==for

- **Είναι δυνατό να μετασχηματισθεί ένας βρόχος *FOR* σε βρόχο *WHILE*;**
- **Ναι!**

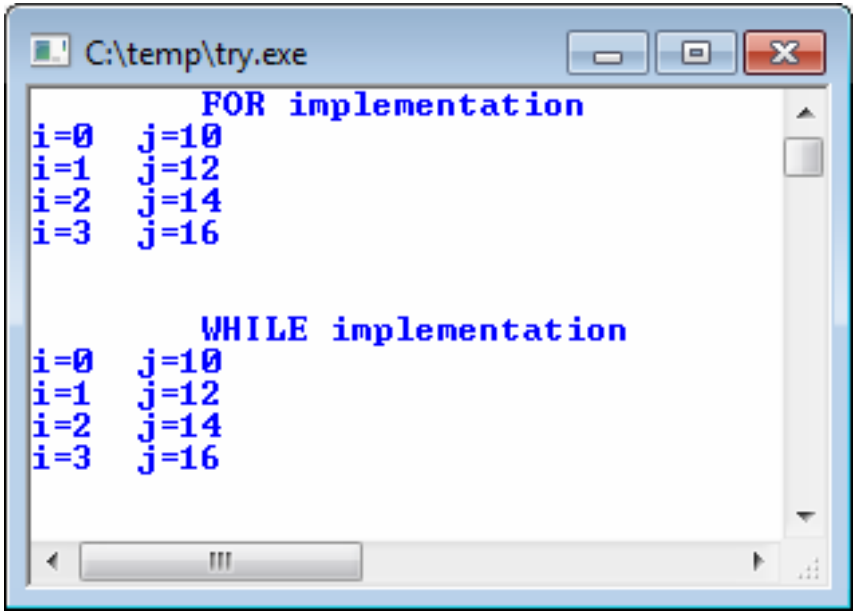
```
for (init; cond; step)
{
   προτάσεις;
}
```

≈

```
init;
while (cond)
{
   προτάσεις;
   step;
};
```

Παράδειγμα:

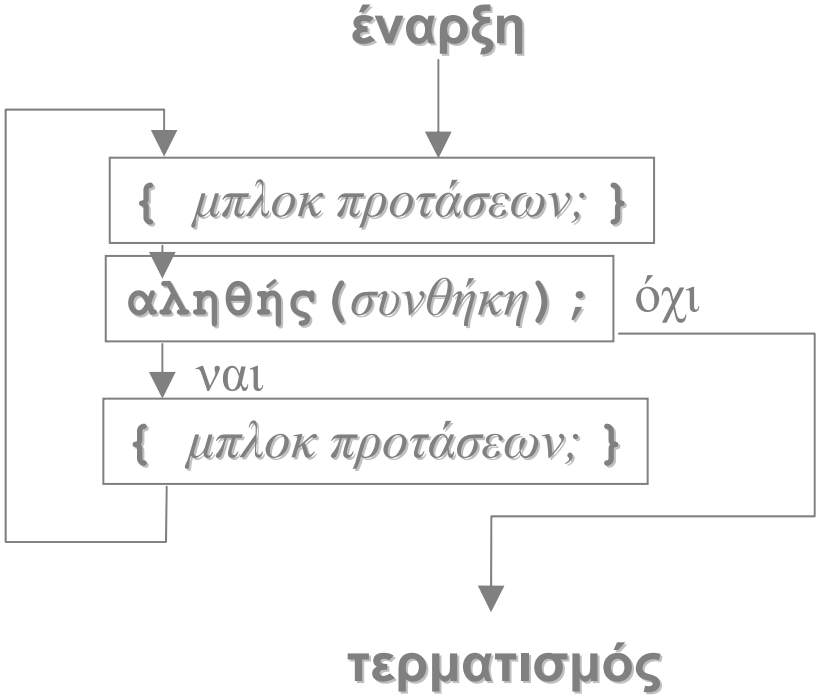
```
#include <stdio.h>
main()  {
  int i,j;
  printf("\tFOR implementation\n");
  for (i=0,j=10;((i<10) && (j<18));i++,j=j+2)  {
      printf("i=%d  j=%d\n",i,j);
  }
  printf("\n\n\tWHILE implementation\n");
  i=0;    j=10;
  while ((i<10) && (j<18))  {
      printf("i=%d  j=%d\n",i,j);
      i++;
      j=j+2;
  }
}
```



Διακοπτόμενοι βρόχοι στη C: κωδική λέξη break

3) Διακοπτόμενος βρόχος: (αποφύγετέ τον!)

```
float energy;
 ...
while(TRUE)
{
    drink_water();
    if(energy <= 2.384) break;
    jump_10_meters();
}
```

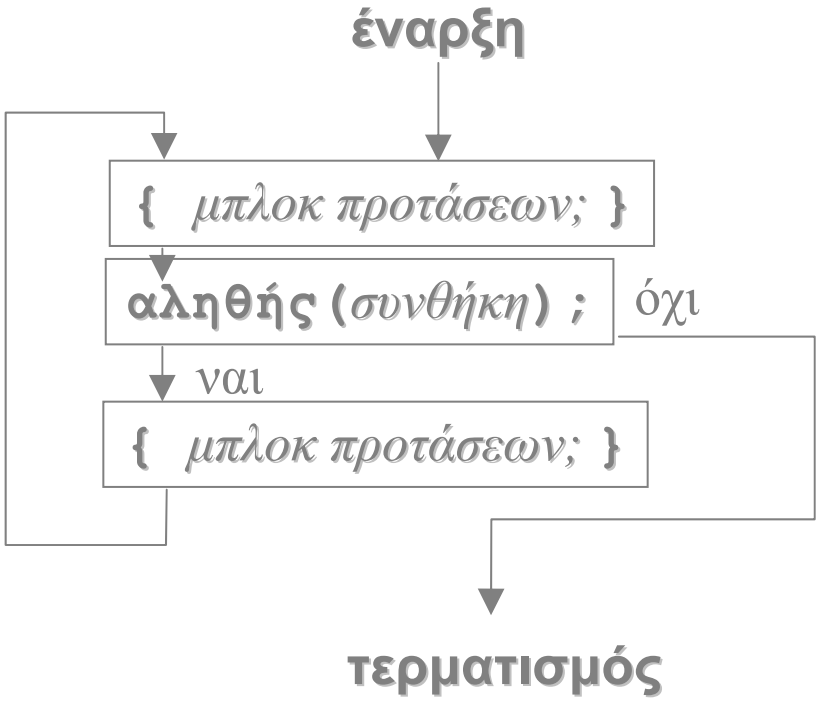


Διακοπτόμενοι βρόχοι στη C: κωδική λέξη break

3) Διακοπτόμενος βρόχος: (μερικές φορές απαραίτητος)

```
int days;
float food,fat;

for(days=155; days>0; days--)
{
    work_all_day();
    if( food+fat < 0.01) break;
    sleep_all_night();
}
die_quietly();
```

Έξοδος από τη for



Σύνοψη: Βρόχοι στη C

- **Τρία είδη βρόχων στη C,**
  - **WHILE**
  - **DO WHILE**
  - **FOR**

- **Κάθε είδος είναι κατάλληλο για διαφορετική εργασία, κατά συνέπεια η επιλογή βασίζεται στο είδος της εργασίας.**

→ Μη χρησιμοποιείτε βρόχους 'for' σε όλες τις περιπτώσεις και άκριτα! ←

WHILE(συνθήκη){προτάσεις;};

- **Βρόχος με συνθήκη εισόδου, οδηγούμενος από γεγονός**

Έλεγξε τη συνθήκη.
Εάν είναι αληθής
Προχώρησε στις προτάσεις
Ξεκίνησε από την αρχή
Αλλιώς σταμάτησε

- **Εάν φαίνεται δυσνόητο για βρόχους οδηγούμενους από μετρητή, χρησιμοποίησε *for*.**

DO{προτάσεις;}WHILE(συνθήκη;)

- **Βρόχος με συνθήκη εξόδου**

  Εκτέλεσε τις προτάσεις

  Έλεγξε τη συνθήκη

  Εάν είναι αληθής

  Ξεκίνησε από την αρχή

  Αλλιώς σταμάτησε

- **Για τους βρόχους που είναι οδηγούμενοι από μετρητή προσπάθησε να διευθετήσεις το πρόβλημα έτσι ώστε να μπορεί να χρησιμοποιηθεί *for*.**

FOR(αρχική; συνθήκη; μετρητής){προτάσεις;}

- **Βρόχος με συνθήκη εισόδου, οδηγούμενος από μετρητή**

Αρχικοποίησε
Έλεγξε τη συνθήκη
Εάν είναι αληθής
Εκτέλεσε τις προτάσεις
Ενημέρωσε το μετρητή
Επάνελθε στον έλεγχο της συνθήκης
Αλλιώς ενημέρωσε το μετρητή και σταμάτησε

Ρητή διακλάδωση (goto)

Η πρόταση

goto <ετικέτα>;

μεταφέρει τον έλεγχο στην πρόταση που σημειώνεται με την ετικέτα ως

<ετικέτα>: πρόταση

Η εντολή *goto* *πρέπει να αποφεύγεται* γιατί οδηγεί σε κώδικα "σπαγγέτι" και αίρει τα πλεονεκτήματα του δομημένου προγραμματισμού. Μπορεί να χρησιμοποιηθεί σε περιπτώσεις εξόδου από πολύ βαθιά ενσωματωμένη δομή, που μια προσεκτική χρήση της *goto* μπορεί να δώσει πιο συμπαγή κώδικα.

Η εντολή continue

- **Η εντολή *continue* μεταφέρει τον έλεγχο της ροής στην αρχή του βρόχου. Χρησιμοποιείται συνήθως όταν θέλουμε να μεταφέρουμε τον έλεγχο στην επόμενη επανάληψη του βρόχου, παραλείποντας την εκτέλεση του υπόλοιπου τμήματος του σώματος του βρόχου.**

- **Στους βρόχους *while* και *do–while* η εντολή *continue* υποχρεώνει τον έλεγχο του προγράμματος να περάσει κατευθείαν στη συνθήκη ελέγχου και να προχωρήσει κατόπιν στην επεξεργασία του βρόχου. Στην περίπτωση της *for* ο υπολογιστής εκτελεί πρώτα το τμήμα του βρόχου και κατόπιν τη συνθήκη ελέγχου, προτού συνεχισθεί η εκτέλεση του βρόχου.**

Παράδειγμα: **Το πρόγραμμα που ακολουθεί εμφανίζει στην οθόνη μόνο τους άρτιους αριθμούς.**

```
# include <stdio.h>
void main()  {
        int x;
        for ( x=0; x<100; x++ )  {
                if (x%2) continue;
                printf( "%d",x );
                <προτάσεις>;
        }
}
```

Κάθε φορά που παράγεται ένας περιττός αριθμός ενεργοποιείται η εντολή διακλάδωσης και εκτελείται η *continue*, παρακάμπτεται η *printf()* και οι υπόλοιπες προτάσεις, οπότε ο έλεγχος προχωρά στην επόμενη επανάληψη.

Κανόνες χρήσης προτάσεων ροής ελέγχου

1. Τοποθετείτε πάντοτε το σώμα των προτάσεων διακλάδωσης υπό συνθήκη και επανάληψης μία θέση στηλογνώμονα δεξιότερα, για αύξηση της αναγνωσιμότητας του κώδικα. Στην περίπτωση δε που το σώμα αποτελείται από περισσότερες της μίας προτάσεις, περικλείετε αυτές σε άγκιστρα.

2. Αποφεύγετε τη χρήση της πρότασης διακλάδωσης *goto*. Καταστρέφει τη δόμηση του προγράμματος και τις περισσότερες φορές προδίδει αδυναμία κατασκευής δομημένου κώδικα.

3. Προτιμήστε το βρόχο επανάληψης συνθήκης εισόδου (*while*) από τον αντίστοιχο συνθήκης εξόδου (*do-while*) γιατί οδηγεί σε πιο ευανάγνωστο κώδικα.

Κανόνες χρήσης προτάσεων ροής ελέγχου

4. Χρησιμοποιείστε την εντολή break σε προτάσεις *switch*. Γενικά αποφεύγετε τη χρήση των *break* και *continue* σε βρόχους επανάληψης, επειδή διακόπτουν την κανονική ροή ελέγχου και καθιστούν την παρακολούθησή της δύσκολη.

5. Ελέγξτε σχολαστικά και βεβαιωθείτε ότι κάθε συνθήκη βρόχου επανάληψης οδηγεί στην έξοδο μετά από πεπερασμένες επαναλήψεις (να μη δημιουργούνται *ατέρμονες βρόχοι (infinite loops)*).

Ο τελεστής κόμμα (,)

Ο τελεστής κόμμα (,) επιτρέπει την παράθεση περισσότερων της μίας εκφράσεων σε θέσεις όπου επιτρέπεται μία έκφραση. Η τιμή της έκφρασης είναι η τιμή της δεξιότερης των εκφράσεων. Συνήθως περιπλέκει τον κώδικα και για αυτό το λόγο η χρήση του είναι περιορισμένη, εκτός από την πρόταση *for*, στην οποία συνηθίζεται να χρησιμοποιείται ως συνθετικό των εκφράσεων αρχικοποίησης και ανανέωσης. Για παράδειγμα, η πρόταση

```
for (i=0,j=10;  i<8;  i++,j++) t[j]=s[i];
```

αντιγράφει τα οκτώ πρώτα στοιχεία του πίνακα s στον t, ξεκινώντας από το ενδέκατο στοιχείο του.

Ο τελεστής κόμμα (,)

Αποφύγετε προτάσεις όπως η

```
for (ch=getchar(),j=0;ch!='A';j++,putchar(ch),ch=getchar())
```

Η πρόταση αυτή διαβάζει χαρακτήρες και τους αποτυπώνει στην οθόνη ενόσω δεν πληκτρολογείται ο χαρακτήρας 'A'. Ο μετρητής j δεν παίζει ρόλο στο βρόχο.

Η πρόταση, αν και είναι συμπαγής ως προς τον κώδικα, μειώνει σε μεγάλο βαθμό την αναγνωσιμότητά του.