

Παράδειγμα:

Να ελεγχθεί κατά πόσον τα ακόλουθα τμήματα κώδικα είναι *λειτουργικά ισοδύναμα* δηλαδή δεχόμενα τα ίδια δεδομένα, οδηγούν στα ίδια αποτελέσματα.

```
int george, john;  
george=10;  
john=george+5;  
if (george<20)  
{  
    george*=2;  
}  
else  
{  
    john=1500;  
}  
printf( "john=%d\n", john );
```

```
int george, john;  
george=10;  
john=george+5;  
if (george<20)  
{  
    george*=2;  
}  
if (george>=20)  
{  
    john = 1500;  
}  
printf( "john=%d\n", john );
```

Οι πρώτες πέντε γραμμές των δύο τμημάτων κώδικα είναι ίδιες, στο τέλος των οποίων η μεταβλητές **george** και **john** έχουν λάβει τις τιμές **20** και **15**, αντίστοιχα. Στην έκτη γραμμή υπάρχει διαφοροποίηση: στο πρώτο τμήμα κώδικα το **else** δε θα εκτελεσθεί, καθώς η συνθήκη του **if** ήταν αληθής, και η μεταβλητή **john** θα διατηρήσει την τιμή της (**15**). Στο δεξί τμήμα κώδικα όμως, η διακλάδωση **if** είναι καινούρια, η συνθήκη είναι αληθής (**george=20>=20**) και η μεταβλητή **john** θα λάβει νέα τιμή (**1500**). Κατά συνέπεια τα δύο τμήματα κώδικα δεν είναι λειτουργικά ισοδύναμα.

convert_time.c

Υπό συνθήκη διακλάδωση *switch()*

```
switch(έκφραση)
{
    case (σταθ.-έκφρ. 1):
        προτάσεις;
        break;
    case (σταθ.-έκφρ. 2):
        προτάσεις;
        break;
    default:
        προτάσεις;
        break;
}
```

•Όταν υπάρχουν πολλές ένθετες προτάσεις *if* ή *elseif*, υπάρχει δυσκολία στην ανάγνωση και καθυστέρηση στην εκτέλεση.

•Τότε προτιμάται η χρήση της *switch()*.

switch() (συνέχεια)

- Η πρόταση **switch** επιτρέπει τον προσδιορισμό απεριόριστου αριθμού διαδρομών ανάλογα με την τιμή της έκφρασης.
- Υπολογίζεται η έκφραση και η τιμή της συγκρίνεται διαδοχικά με τις σταθερές εκφράσεις (**σταθ.-έκφρ. 1, σταθ.-έκφρ. 2, ...**). Ο έλεγχος μεταφέρεται στις προτάσεις που είναι κάτω από τη σταθερά έκφραση, με την οποία ισούται η τιμή της **έκφρασης**.
- Εάν δεν ισούται με καμία από τις σταθερές εκφράσεις, ο έλεγχος μεταφέρεται στις προτάσεις που ακολουθούν την ετικέτα **default**, εάν αυτή υπάρχει, αλλιώς στην πρόταση που ακολουθεί το σώμα της **switch**.

switch() (συνέχεια)

- Η πρόταση ελέγχου **break**, η οποία υποδηλώνει άμεση έξοδο από τη **switch**, είναι προαιρετική. Εάν αυτή λείπει, η εκτέλεση των προτάσεων που ακολουθούν την επιλεγείσα ετικέτα θα ακολουθήσει από την εκτέλεση και των προτάσεων και των επόμενων **case** ετικετών.
- Στην πράξη η **break** συναντάται σχεδόν πάντοτε, ακόμη και μετά τις προτάσεις τής **default**. Το τελευταίο γίνεται για να προστατευθούμε από το δύσκολο στην ανεύρεση σφάλμα που θα προκύψει από μελλοντική προσθήκη μίας νέας ετικέτας, με ταυτόχρονη παράλειψη προσθήκης πριν από αυτή τής **break**.

switch() (συνέχεια)

Η λειτουργία της *switch* διέπεται από το ακόλουθο σύνολο κανόνων:

- Κάθε *case* πρέπει να έχει μία *int* ή *char* σταθερά έκφραση.
- Δύο *case* δεν μπορούν να έχουν την ίδια τιμή.
- Οι προτάσεις κάτω από την ετικέτα *default* εκτελούνται όταν δεν ικανοποιείται καμία από τις *case* ετικέτες.
- Η *default* δεν είναι απαραίτητα η τελευταία ετικέτα.
- Οι *case* και *default* μπορούν να τοποθετηθούν με οποιαδήποτε σειρά.
- Η *break* μετά την τελευταία ετικέτα αποτελεί καλή τακτική, αν και δεν είναι απαραίτητη.

Παράδειγμα: Να γραφεί τμήμα κώδικα, το οποίο να δίνει τη δυνατότητα στο χρήστη να εισάγει δύο αριθμούς και στη συνέχεια να εκτελεί επί αυτών επιλεκτικά μία από τις τέσσερις αριθμητικές πράξεις

Λύση:

Χρησιμοποιώντας “δομημένα Ελληνικά”, η διεργασία περιγράφεται ως εξής:

1. **πάρε δύο αριθμούς**
2. **ενημέρωσε τον χρήστη για δυνατές επιλογές**
3. **πάρε την επιλογή του χρήστη**
4. **ανάλογα με την επιλογή**
5. **εκτέλεσε την αντίστοιχη πράξη**
6. **εμφάνισε το αποτέλεσμα**
7. **τερμάτισε**

Ο κώδικας του προγράμματος είναι ο ακόλουθος:

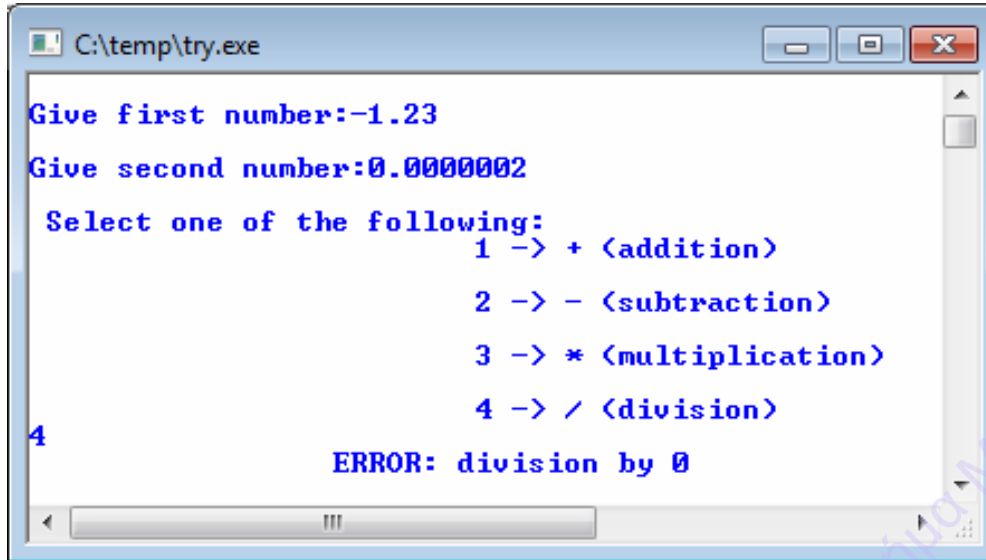
```
#include <stdio.h>           #include <math.h>
#define MY_ZERO 1E-5
#define ADD 1
#define SUB 2
#define MUL 3
#define DIV 4

int main(){
    float num1, num2,result;
    int choice, flag=0;
    printf( "\nGive first number:"); scanf("%f",&num1 );
    printf( "\nGive second number:"); scanf("%f",&num2 );
    printf( "\n Select one of the following:" );
    printf( "\n\t\t\t %d -> + (addition)\n",ADD );
    printf( "\n\t\t\t %d -> - (subtraction)\n",SUB );
    printf( "\n\t\t\t %d -> * (multiplication)\n",MUL );
    printf( "\n\t\t\t %d -> / (division)\n",DIV );
    scanf( "%d",&choice );
```

*Κάθε μία από τις πράξεις
αντιστοιχίζεται σε έναν ακέραιο.*

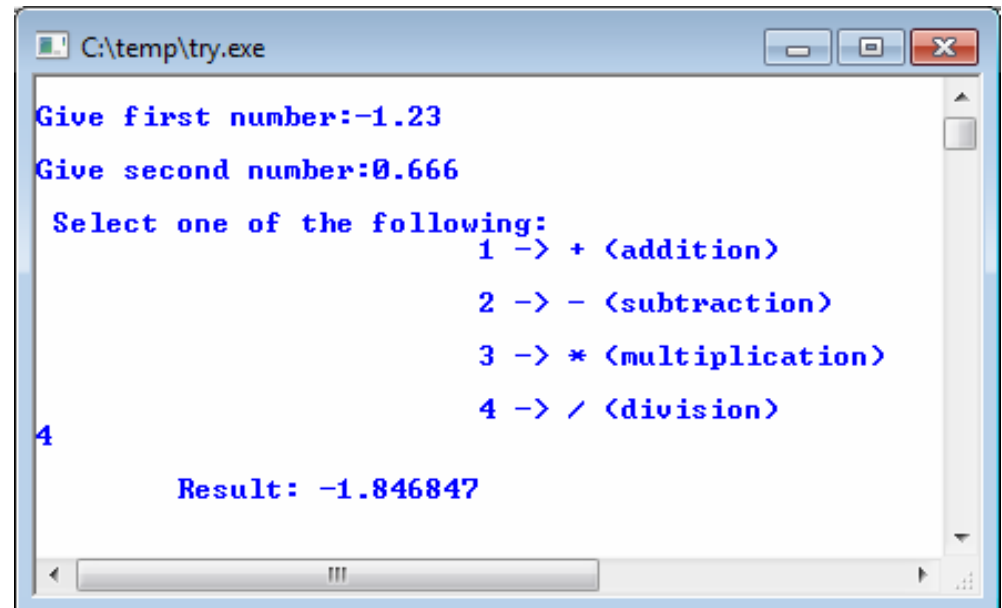

```
switch(choice){  
    case ADD:  
        result=num1+num2;  
        break;  
    case SUB:  
        result=num1-num2;  
        break;  
    case MUL:  
        result=num1*num2;  
        break;  
    case DIV:  
        if (fabs(num2)>MY_ZERO) result=num1/num2; // num2 != 0  
        else { printf( "\t\t ERROR: division by 0" ); flag=1; }  
        break;  
    default:  
        printf( "This selection is not supported" ); flag=1;  
        break;  
} // τέλος της switch  
if (!flag) printf( "\n\tResult: %f\n",result );  
return 0; } // τέλος της main
```

Αποτελέσματα:



```
C:\temp\try.exe

Give first number:-1.23
Give second number:0.0000002
Select one of the following:
    1 -> + <addition>
    2 -> - <subtraction>
    3 -> * <multiplication>
    4 -> / <division>
4
ERROR: division by 0
```



```
C:\temp\try.exe

Give first number:-1.23
Give second number:0.666
Select one of the following:
    1 -> + <addition>
    2 -> - <subtraction>
    3 -> * <multiplication>
    4 -> / <division>
4
Result: -1.846847
```

Θεματική ενότητα 5: **Προτάσεις επανάληψης – βρόχοι**

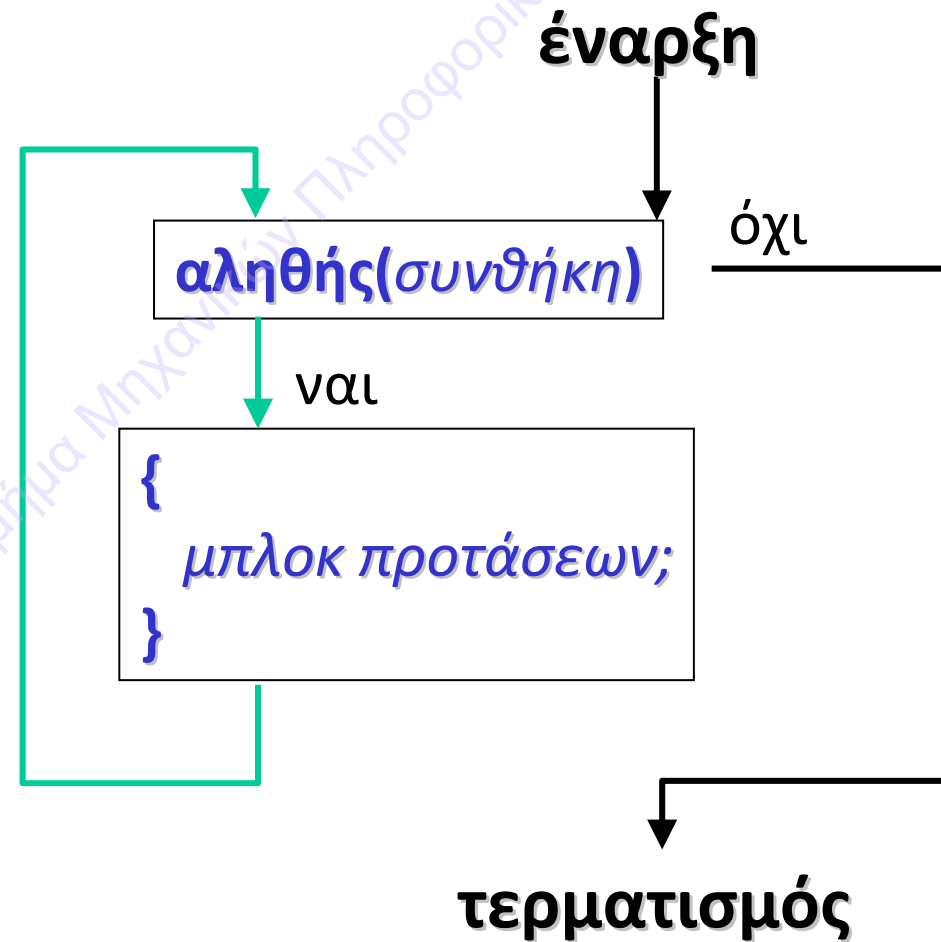
Προτάσεις επανάληψης - γενικά

- 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

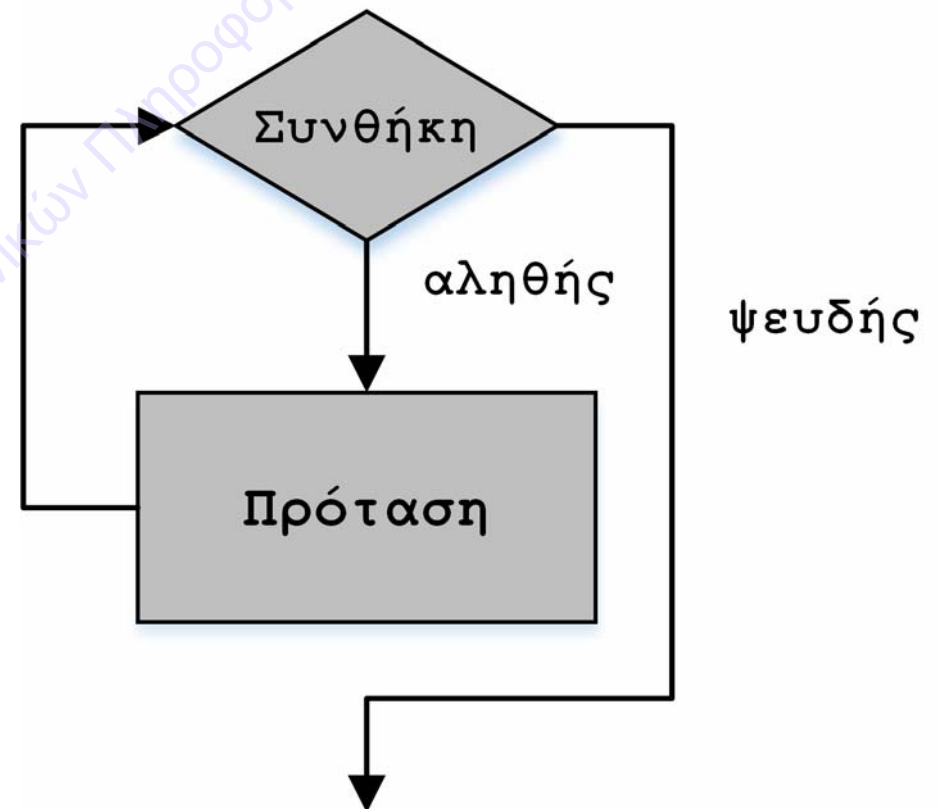
1) Βρόχος με συνθήκη εισόδου (pre-test loop):

- α) οδηγούμενος
από γεγονός
- β) οδηγούμενος
από μετρητή



while - do

while Συνθήκη **do** Πρόταση

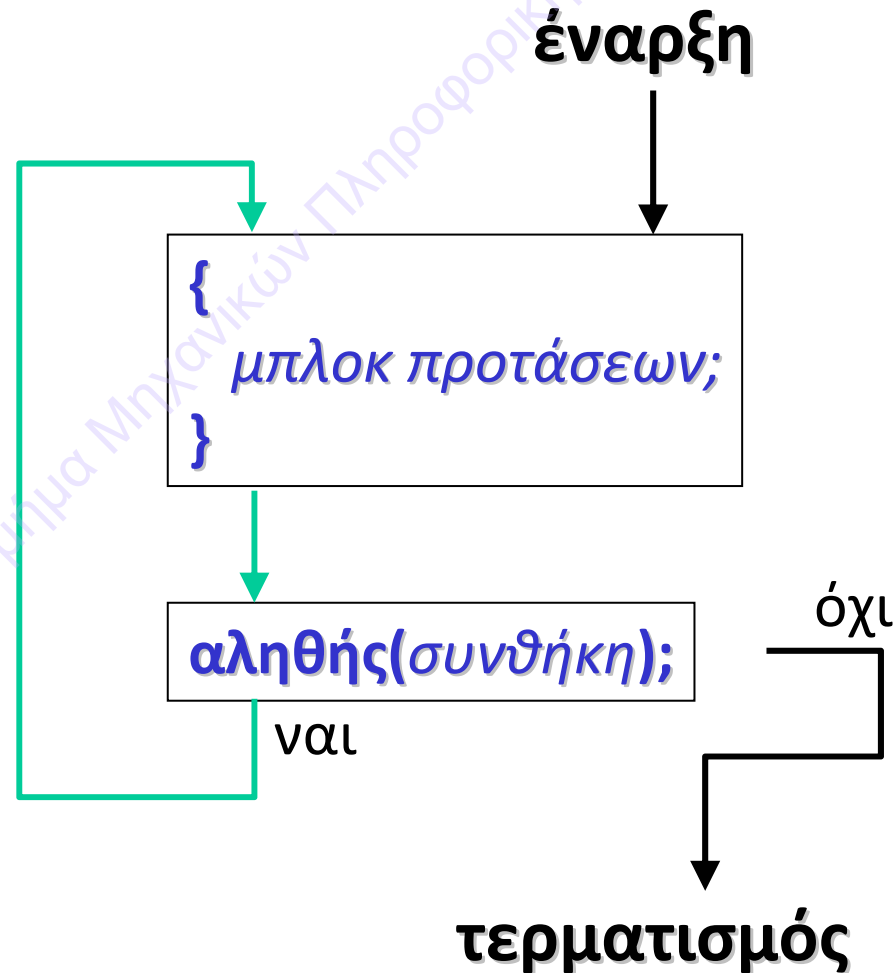


do - while

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

α) οδηγούμενος
από γεγονός

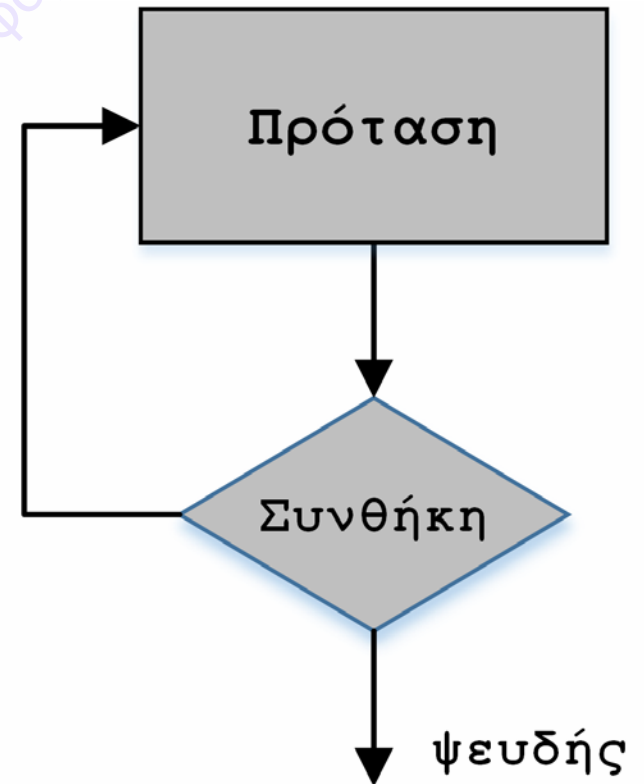
β) οδηγούμενος
από μετρητή



do - while

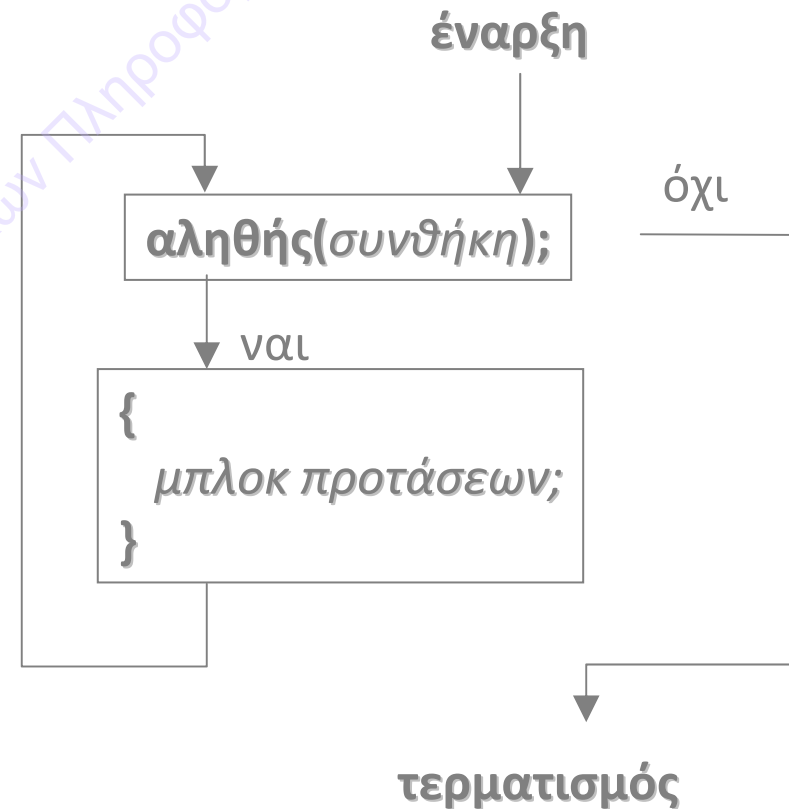
do Πρόταση **while** Συνθήκη

αληθής



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:

Έλεγε 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.*/
```

