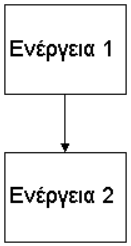
ΗΥ-150
Προγραμματισμός

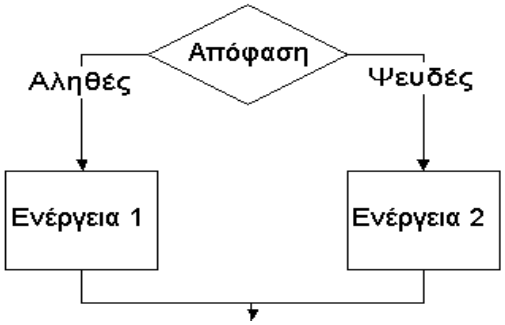
Επανάληψη

Σειριακή εκτέλεση εντολών

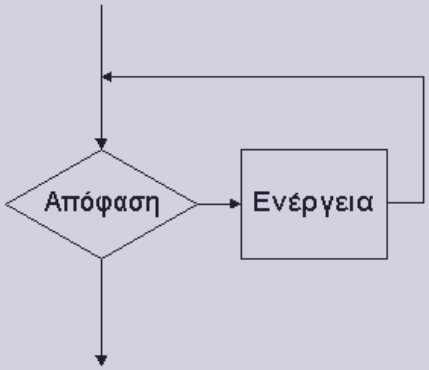
- Όλα τα προγράμματα «γράφονται» χρησιμοποιώντας 3 είδη ροής εντολών:
  - Σειριακή



  - Σε διακλάδωση
    - if, if/else, switch



  - **Επαναληψηπτικά**
    - **for, while, do-while**



Εντολές επανάληψης

- Οι εντολές επανάληψης μας επιτρέπουν να προσδιορίσουμε ένα block εντολών που θα **επαναλαμβάνεται** όσο μία **συνθήκη** παραμένει αληθής
- Το block εντολων μπορεί να περιέχει και μία μόνο εντολή και, σηνήθως, εκτελείται περισσότερο από μια φορά

Σειριακή εκτέλεση

- Θέλουμε να τυπώσουμε τους αριθμούς από το μηδέν έως τρία στην οθόνη

```
int main()
{
  cout << "0";
  cout << "1";
  cout << "2";
  cout << "3";
  return 0;
}
```

Πως θα τυπώσω τους αριθμούς ως το 10000;

Επαναληπτική εκτέλεση

```
int a = 0;
while (a<4)
{
   cout <<  a;
   a = a+1;
}
```

Συνθήκη – Condition (conditional statement)

Επαναλμβανόμενη ενέργεια – block εντολών.

Κατά τα γνωστά: εάν η εντολή είναι μία, τότε τα {} μπορούν να παραληφθούν.

Τυπώνει 0,1,2,3.

Εικονικό «τρέξιμο» του προγράμματος

a	a<4	output
0	1	0
1	1	1
2	1	2
3	1	3
4	0	

```
int a=0;
while (a<4)
{
        cout << a;
        a = a+1;
}
```

Επανάληψη σε αντίστροφη σειρά

```
int a = 4;
while (a>0)
{
        a = a-1;
        cout << a;
}
```

Τυπώνει 3,2,1,0.

Διακλάδωση - Επανάληψη

- **Επανάληψη**

```
int a = 0;
while (α<4)
{
   cout << a;
   a = a+1;
}
```

Τυπώνει 0,1,2,3.

- **Διακλάδωση**

```
int a = 4;
if (α>0)
{
   a = a-1;
   cout << a;
}
```

Τυπώνει 3.

- Η **τιμή** της έκφρασης (συνθήκης) είναι **αληθής, ή ισοδύναμα, η αποτίμηση της είναι *διάφορη του μηδενός***

- *Που είναι το πρόβλημα παρακάτω;*

```
while (a<6);
   f = 3 ;
```

Κατηγορίες Βρόχων Επανάληψης

- **Με μετρητή:** ρητά προσδιορισμένος αριθμός επαναλήψεων
  - επανέλαβε όσο ο μετρητής <> από μια τιμή
- **Με σημαία (flag):** μη-προσδιορισμένος αριθμός επαναλήψεων.
  - η συνθήκη τερματισμού εξαρτάται από τιμές μεταβλητών που μεταβάλλονται κατα την επανάληψη
- **Συνδυασμοί**
  - με χρήση λογικών τελεστών

```
int c =0;
while (c<2)
{
        cout << "Hello";
        c = c+1;
}
```

```
int a =0;
while (a!=-1)
{
        cin >> a;
        cout <<"a^2=" << a*a;
}
```

```
while ((c<2) || (a!=-1))
{
        ...
}
```

while

Επαναληπτικές Εντολές - while

while (έκφραση)

{

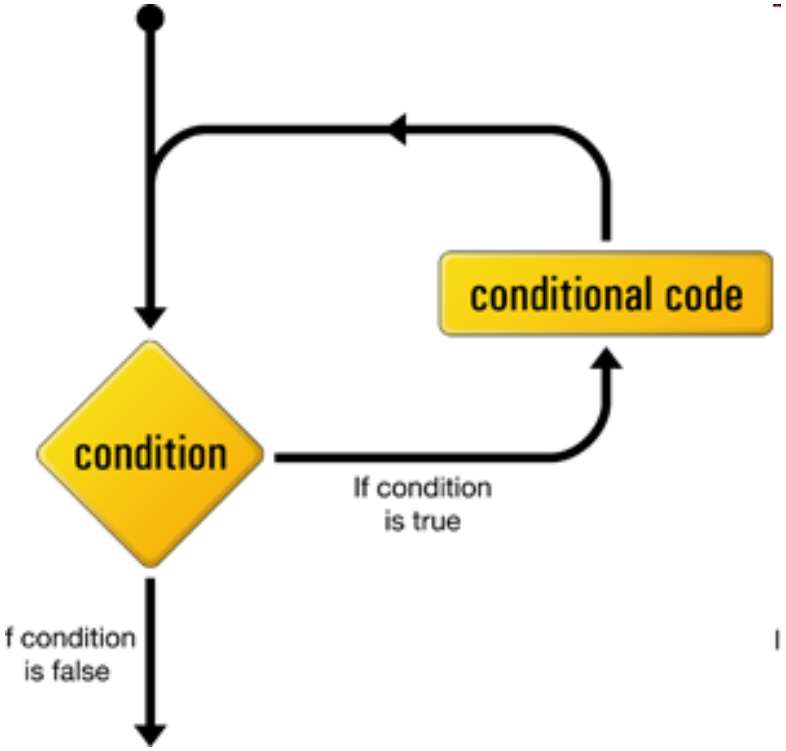
εντολή / block εντολών,

statement

}

while: Υπολόγισε την *έκφραση*

- Αν
  - *αληθής* εκτέλεσε την *εντολή*
  - *ψευδής* τερμάτισε
- Επέστρεψε στο while



Πίνακας μετατροπής F° – C°

```
int main()
{
   int a = 0;
   while (a <= 100)
   {
      cout << a << " deg F =" << (a - 32) * 5 /
   9 << " deg C";
      a = a + 10;
   }
   return 0;
}
```

```
0 degrees F = -17 degrees C
10 degrees F = -12 degrees C
20 degrees F = -6 degrees C
30 degrees F = -1 degrees C
40 degrees F = 4 degrees C
50 degrees F = 10 degrees C
60 degrees F = 15 degrees C
70 degrees F = 21 degrees C
80 degrees F = 26 degrees C
90 degrees F = 32 degrees C
100 degrees F = 37 degrees C
```

Άσκηση

- Γράψτε ένα τμήμα προγράμματος που τυπώνει τους ακέραιους αριθμούς από το **1 μέχρι το 12**. Κάθε αριθμός που **διαιρείται με το 3** να τυπώνεται ένα ***** δίπλα του. Ο κάθε αριθμός να τυπώνεται σε **ξεχωριστή γραμμή.**

- ***1***
- ***2***
- ***3****
- ***4***
- ***5***
- ***6****

...

Λύση

```
int i;                                  /* loop counter */
i = 1;
while (i <= 12)
{
   cout << i;
   if((i%3) == 0)
        cout << "*";
   cout << "\n";
   i = i + 1;
}
```

Υπολογισμός των τετραγώνων των πρώτων 100 φυσικών αριθμών και του αθροίσματός τους

1*1+2*2+3*3+…+10*100

```
int c=1,sum=0;
while (c<=100)
{
   cout << “C is” << c << “, square of C is” << c*c << endl;
   sum = sum + c*c;
   c=c+1;
}
cout << “sum is “ << sum;
```

Άθροισμα Σειράς

```
int number;                    /* κρατάει δεδομένο εισόδου κάθε φορά*/
int sum;                       /* υφιστάμενη τιμή αθροίσματος */
cin>>number;                   /* διάβασε πρώτο στοιχείο */
sum = 0;                       /* αρχικοποίηση αθροίσματος*/
while(number != 0)
{
    sum = sum + number;        /* άθροισμα με υφιστάμενο sum*/
    cin >> number;
}
cout >> “To athroisma tis seiras einai ” >> sum;
```

Παραδείγματα Βρόχων

```
/*Εκτυπώνει τους 10 πρώτους αριθμούς του N που το άθροισμα των ψηφίων τους
    είναι 7 και την μέση τους τιμή*/

#define N 10
    int x = 1;                  /*metablhth gia enan tyxaio ari8mo*/
    int cnt = 0;                /*metablhth gia to # tvn ari8mwn*/
    float mean = 0;             /*metablhth gia th mesh timh*/

    while (cnt < N)
    {
        if (sumOfDigits(x) == 7)
        {
            mean = x+mean;
            cnt = cnt + 1;
            cout << “O ” << cnt << “ ari8mos einai o ” << x;
        }
        x = x + 1;
    }
    mean = mean/N;
    cout << “H mesh tous timh einai ” << mean;
```

Παραδείγματα Βρόχων

```
/*Υπολογισμός του αθροίσματος των ψηφίων του αριθμού n*/

int sumOfDigits(int n)
{
    int pn,x,sum;
    pn = n;
    sum = 0;

    while (pn > 0)
    {
        x = pn % 10;
        pn = pn/10;

        sum += x;
    }

    return sum;
}
```

Επαναληπτικές Εντολές - while

Π.χ., Υπολογισμός των τετραγώνων των πρώτων 10 φυσικών αριθμών που είναι πολλαπλάσια του 7 ή του 9 και του αθροίσματός τους

```
int c=1,sum=0,pl = 0;
while (pl<=10)
{
    if (c % 7 == 0 || c % 9 == 0)
    {
        cout << “C is ” << c << “, square of C is ” << c*c;
        sum = sum + c*c;
        pl++;
    }
    c++;
}
cout << “Sum = ” << sum;
```

Εύρεση Μέγιστου

- Γράψτε ένα πρόγραμμα που αναγνωρίζει τον **πιο μεγάλο αριθμό** σε μια, απροσδιορίστου μεγέθους, σειρά ακέραιων αριθμών. Η σειρά εισάγεται από μονάδα εισόδου και τερματίζεται με την τιμή 0.
- π.χ.
- Enter series: 2 4 5236 6 31 978 5555 23 0
- >> 5555

Εύρεση Μέγιστου

```
int number;                    /* κρατάει δεδομένο εισόδου κάθε φορά*/
int maximum;                   /* υφιστάμενη τιμή μεγίστου αριθμού */
cin >> number;                 /* διάβασε πρώτο στοιχείο */
maximum = number;              /* αρχικοποίηση μεγίστου */

while(number != 0)
{
    cin >> number;       /* διάβασε επόμενο στοιχείο */
    if (number > maximum)     /* επεξεργασία */
        maximum = number;
}

cout << “To megisto stoixeio tis seiras einai ” << maximum;
```

- Γράψτε ένα πρόγραμμα το οποίο υπολογίζει το άθροισμα μιας ***απροσδιορίστου μεγέθους*** σειράς ακέραιων αριθμών.
  - *2 4 236 6 31 23* ▮
  - *3 549 33 29 12 190 2398 99 19 1* ▮
  - Η σειρά εισάγεται από μονάδα εισόδου και τερματίζεται με την τιμή 0.

Τη στιγμή μέταφρασης του κώδικα

- **π.χ.**
- **Enter series: 2 4 236 6 31 23 0**
- **>> 302**

Σειριακή είσοδος χαρακτήρων

```
char c;
while( (cin.get(c)) )
   cout.put(c);
```

cin.get()

```
/* letter_count.c A program to count letters in input. */
main()
{
   int c ;
   int count = 0;
   while ( cin.get(c))
        if ( ( c >= 'A' ) && ( c <= 'Z' )
          || ( c >= 'a' ) && ( c <= 'z' ) )
               count ++ ;
   cout << count << “ letters” ;
}
```

Increment operators

Τελεστές Αύξησης/Μείωσης

- ***Συχνά θέλουμε να αυξάνουμε/μειώνουμε μετρητές κατά 1.***
- Prefix Operator *(προ-σημειογραφική)*
  - ++i; i = i + 1;
  - --i; i = i - 1;
- Εκτελούμε την πράξη ***πρώτα*** και μετά χρησιμοποιούμε την **νέα** τιμή της μεταβλητής
- Postfix *Operator (μετά-σημειογραφική)*
  - i++; i = i + 1;
  - i--; i = i - 1;
- Χρησιμοποιούμε την **υφιστάμενη** τιμή της μεταβλητής και ***μετά*** εκτελούμε την πράξη.
- Και οι δυο τελεστές εκτελούν την ίδια πράξη (αφαίρεση / πρόσθεση) αλλά με διαφορετική σειρά.

Prefix vs Postfix Τελεστές

i = 5;

x = ++i; **Πρώτα αύξησε** και μετά **ανάθεσε**

y = i++; **Πρώτα ανάθεσε** και μετά **αύξησε**

x είναι 6, y είναι 6 και το i είναι 7

```
x = ++i; <==> { i=i+1;
                x=i;
```

```
y = i++; <==> { y=i;
                i=i+1;
```

```
int my_int = 3;
cout << my_int++;
cout << my_int++;
cout << my_int;
```

Εκτυπώνει: 3,4,5

```
int my_int = 3;
cout << ++my_int;
cout << ++my_int;
cout << my_int;
```

Εκτυπώνει: 4,5,5

Σύνθετοι Τελεστές Ανάθεσης

- *Συχνά θέλουμε να αυξάνουμε / μειώνουμε μετρητές* ***κατά κάποιο Κ (>1)****.*
- Πρόσθεση: **i+=k;** **i = i + k;**
- Πολλαπλασιασμός: **i*=k;** **i = i * k;**
- Διαίρεση: **i/=k;** **i = i / k;**
- Υπόλοιπο: **i%=k;** **i = i%k;**

- Π.χ.

i-= k + z; *i = i – (k+z);*

Increment and Decrement Operators

```
int c,n;

n = 0;
while( (cin.get(c)) )
   if( c == '\n' )
        n++;

cout >> n >> “lines”;
```

for

Επαναληπτικές Εντολές - for

Συντακτικό:

for (έκφραση1 ; έκφραση2 ; έκφραση3) εντολή

Σημασιολογία:

- Υπολόγισε την τιμή της *έκφραση1*
- Υπολόγισε την τιμή της *έκφραση2*
- Αν η *έκφραση2* είναι αληθής (μη μηδενική τιμή) τότε
- Εκτέλεση την *εντολή*
- Υπολόγισε την τιμή της *έκφραση3*
- Πήγαινε στο βήμα 2
- Ειδάλλως, εξήλθε του βρόχου

Ισοδύναμο με:

```
έκφραση1;
while (έκφραση2)
{
    εντολή
    έκφραση3;
}
```

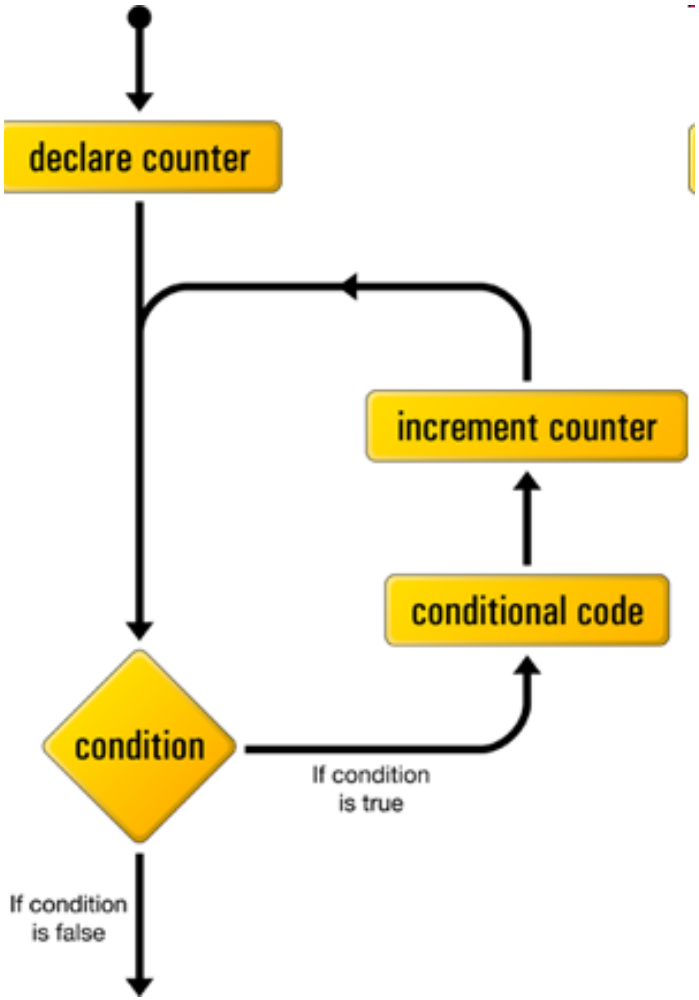
Σημασία **for**

- **for (αρχικοποίηση; συνθήκη επανάληψης ; ενημέρωση)**
- **εντολή;**
- ***Σειρά Εκτέλεσης:***
  - *Αρχικοποίηση. Συνθήκη*
    - εντολή
  - *Ενημέρωση. Συνθήκη*
    - εντολή
  - *Ενημέρωση. Συνθήκη*
    - έξοδος

1η

ν-ιοστή

τελευταία



Παράδειγμα

```
int x;
for (x=0; x<5; x++)
{
   cout << x;
}
```

x	x<5	έξοδος
0	1	0
1	1	1
2	1	2
3	1	3
4	1	4
5	0	

Ροή Έλεγχου for

```
int x,y;

for (x=0; x<5; x++)

{
   cout << x;
}
/* Εδώ το x=5 */
y = x*x + 10;
cout << y;
/* επομένως εκτυπώνεται 35 */
```

Αρχικοποίηση

Ενημέρωση

Συνθήκη

for vs. while

```
int x;
for (int x=0; x<5; x++)
{
        cout << x;
}
```

```
int x = 0;
while (x < 5)
{
        cout << x;
        x++;
}
```

Παραδείγματα

- Αντίστροφη μέτρηση
- Μέτρηση με βήμα (step increment)

```
int x;
for (x=5; x>0; x--)
{
        cout << x;
}
```

Εκτυπώνει 5,4,3,2,1

```
int i;
for (i = 0; i < 10; i += 2)
{
        cout << i;
}
```

Εκτυπώνει 0,2,4,6,8,

```
int i = 10;
for (i = 10; i > 0; i /= 2)
{
        cout << i;
}
```

Εκτυπώνει: 10,5,2,1

```
int i;
int sum = 0;
for (i = m; i <= n; i++)
    sum += f(i);
```

```
int f(int i)
{
        int result = -1;

        result = i*i;

        return result;
}
```

$$\sum_{i=m}^{n} x_i = x_m + x_{m+1} + x_{m+2} + \ldots + x_{n-1} + x_n .$$

Επαναληπτικές Εντολές - for

Συντακτικό:

for (έκφραση1 ; έκφραση2 ; έκφραση3) εντολή

Συνήθως οι παραπάνω εκφράσεις αντιστοιχούν σε:

```
for (
        αρχικοποίηση μετρητών;
        συνθήκη ;
        μεταβολή μετρητών
)
```

έτσι ώστε σε μια γραμμή να φαίνεται πόσες επαναλήψεις θα κάνει ο βρόχος

Επαναληπτικές Εντολές - for

Συγκρίνετε:

```
int c;
for (c=1; c<=10; c++)
{
      cout << “C is ” << c << “, square of C is ” << c*c;
}
```

με

```
int c;
c=1;
while (c<=10)
{
     cout << “C is ” << c << “, square of C is ” << c*c;
     c++;
}
```

- Είναι ισοδύναμα, αλλά το πρώτο προτιμάται από τους προγραμματιστές: μετά την for υπάρχουν όλες οι πληροφορίες μαζεμένες για να καταλάβουμε πως θα εκτελεστεί ο βρόχος

Παραδείγματα Βρόχων

```
/*Υπολογισμός Δύναμης*/

int power(int base, int n)
{
    int i, p;
    p =1;

    for (i = 1; i <= n; i++)
        p = p*base;

    return p;
}
```

```
/*Υπολογισμός παραγοντικού
    n! = n*(n-1)! */
int parag(int n)
{
    int i, p;
    p =1;

    for (i = 2; i <= n; i++)
        p = p*i;

    return p;
}
```

Επαναληπτικές Εντολές - for

Η γλώσσα δεν υποχρεώνει η πρώτη έκφραση να αφορά αρχικοποίηση, η δεύτερη συνθήκη κτλ. Έτσι τα παρακάτω είναι συντακτικά σωστά:

```
k=7;
for (; k; )
    cout << “K is ” << k--;
```

do-while

Εισαγωγή

- Στις εντολές for και while, προτού εκτελεστεί κάποια εντολή στο σώμα του βρόχου, πρέπει να ελεγχθεί κάποια συνθήκη.

- π.χ. int a =0; **while** (a<2) { /* **σώμα** */ a = a+1; }
- π.χ. int **for** (a=0;a<2;a++) {/***σώμα** */ }

Εισαγωγή

- Θέλουμε να διαβάσουμε αριθμούς από το χρήστη και να τους αθροίσουμε. Όταν ο χρήστης δώσει 0 τότε τερματίζει το πρόγραμμα.

```
cin >> a;
while (a!=0)
{
        sum+=a;
        cin >> a;
}
```

- χρειάζομαι 2 cin

Η Εντολή do-while

- do
- εντολή;
- while(συνθήκη);

```
do
{
        cin >> a;
        sum+=a;
}
while (a!=0);
```

- Πρώτα εκτελείται το σώμα της do { /* σώμα */ } while();
- Ο έλεγχος της συνθήκης γίνεται μετά το while

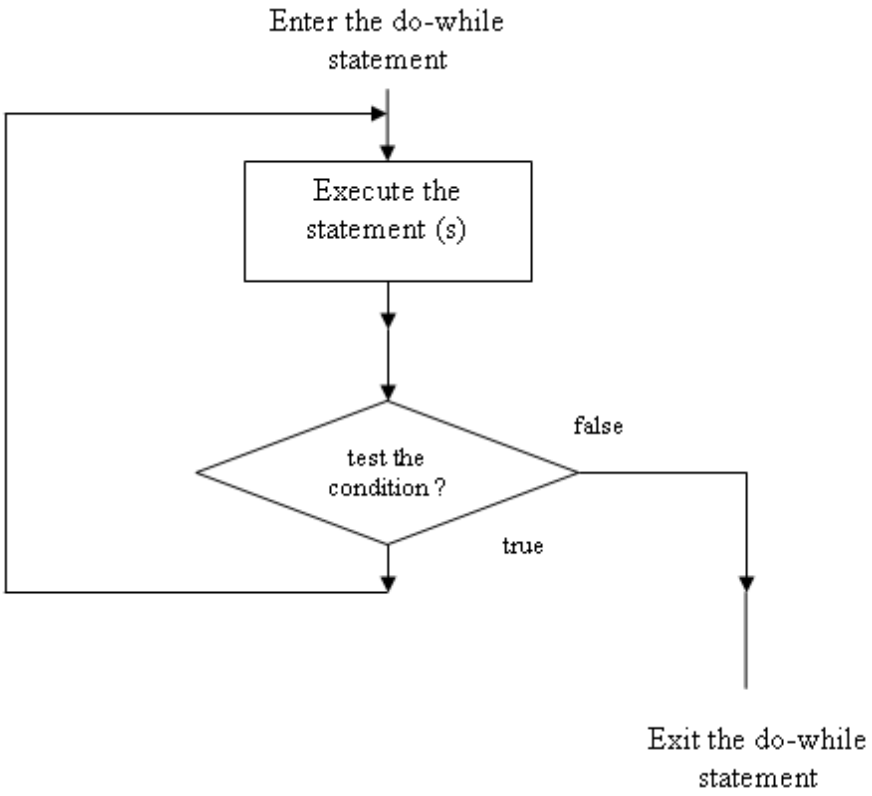
Ροη Ελέγχου

- Εκτέλεσε το σώμα
- Εφόσον η συνθήκη **ισχύει** επανέλαβε

την εκτέλεση των εντολών στο σώμα του βρόχου

- Στο *for* και *while* το σώμα μπορεί να μην εκτελεστεί, ενώ στο *do-while* εκτελείται ***τουλάχιστο μια φορά***

- ***ΠΡΟΣΟΧΗ: Να είμαστε σίγουροι ότι θέλουμε να εκτελεστεί το σώμα εντολών ΠΡΩΤΑ μια φορά και μετά να γίνει ο έλεγχος της συνθήκης…***



Παράδειγμα

```
int number;

do
{
  cout << “Enter a possitive value:”;
  cin >> number;
}
while(number<= 0);
```

```
int number;
cout << “Enter a possitive value: ”;
cin >> number;

while(number<= 0)
{
    cout  << “Enter a possitive value:”;
    cin >> number;
}
```

Επαναληπτικές Εντολές – do/while

```
do
{
    εντολή
}
while (έκφραση);
```

```
int calculator(int a,int b)
{
        int res = 0;
        char ch;
        cout << "Add, Substract, Multiply, Divide";
        do
        {
                cout >>"Enter first letter:";
                cin.get(ch);
        }
        while ( ch!='A' && ch!='S' && ch != 'M' && ch!='D');

        if (ch=='A') res = a+b;
        else if (ch=='S') res = a-b;
        else if (ch=='M') res = a*b;
        else if (ch=='D') res = a/b;

        cout << a << ch << b << "=" << res;
        return res;
}
```

Έξοδος από βρόχο

Έξοδος από βρόχο

- Εντολή **break**
  - Τερματισμός μόνο του πιο «μέσα» βρόγχου ή switch
  - Ειδικές περιπτώσεις
  - Π.χ., βγες από το μενού
- Εντολή **continue**
  - Η ροή πηγαίνει απευθείας στην *έκφραση3* παράσταση της for (αν πρόκειται για for)
  - Κατόπιν η ροή πηγαίνει στην συνθήκη ελέγχου της ανακύκλωσης (για for, while, do)
  - Π.χ., αγνόησε κάποιο στοιχείο του βρόχου

break

- **Η εντολή break** μπορεί να χρησιμοποιηθεί σε βρόχο **while, do-while** και **for** για να προκαλέσει πρόωρη έξοδο από το βρόχο.

```
int i;
for (i = 1; i < 10; i++)
{
        if (i == 5)
        {
                break;
        }
        cout << i;
}
cout << “broke out of loop at i = ” << i;
return 0;
```

ΕΞΟΔΟΣ:
1 2 3 4
Έξοδος από βρόχο με i = 5

break

- **Η εντολή break** εξάγει την εκτέλεση από τον πλησιέστερο βρόχο στην εμβέλεια (scope)

```
for (i = 1; i < 10; i++)
{
        for (j=0;j<5;j++)
        {
                if (j == 2)
                {
                        break;
                }
        }
}
```

Έξοδος από βρόχο του j

continue

- Η εντολή **continue** μπορεί να χρησιμοποιηθεί με **for, while, do-while** για να προκαλέσει τη ματαίωση της εκτέλεσης των επομένων εντολών του βρόχου, για την τρέχουσα επανάληψη.

```
int i;
for (i = 1; i < 10; i++)
{
        if (i == 5)
        {
                continue;
        }
        cout << i;
}
```

Παράλειψη των υπολοίπων εντολών σε μια τρέχουσα επανάληψη και εκ νέου εκτέλεση του σώματος εντολών

ΕΞΟΔΟΣ:
1 2 3 4 6 7 8 9
Έξοδος από βρόχο με i = 10

Παραδείγματα Βρόχων

/*Υπολογισμός της μικρότερης δύναμης του 2 που είναι μεγαλύτερη από τον αριθμό n*/

```
int powerOfTwo(int n)
{
    int p;
    p = 1;

    while (1)
    {
        p = p*2;

        if (p > n)
            break;
    }

    return p;
}
```

break

```
int main() {
   char key;

   cout<<"Press any key or E to exit:\n“<<endl;
   while(1) {
      cin>>key;
      // if E or e, exit
      if (key == 'E' ||  key == 'e')
          break;
   }
   cout<<"Goodbye!\n";
}
```

continue

```
1  int main()
2  {
3      int haystack[SIZE] = {1, 3, 2, 4, 7, 6, 9, 5, 8, 0};
4
5      int needle;
6
7      cout<<"Enter a number (0-9) to see its position:"<<endl;
8      cin>>needle;
9
10     int i;
11     for (i = 0; i < SIZE; i++)
12     {
13         if (needle != haystack[i])
14         {
15             cout<<"Finding at position ”<<i<<“: ”<< haystack[i]<< endl;
16             continue;
17         }
18
19         cout<<"Number ” needle << “ found at position ” <<i<<endl;
20         break;
21     }
22     return 0;
23 }
24
25
26
27
28
```

1/2

```
char c;
while(true) {
        cout << "MAIN MENU:" << endl;
        cout << "l: left, r: right, q: quit -> ";
        cin >> c;
        if(c == 'q')
                break; // Out of "while(1)"
        if(c == 'l') {
                cout << "LEFT MENU:" << endl << "select a or b: ";
                cin >> c;
                if(c == 'a') {
                        cout << "you chose 'a'" << endl;
                        continue; // Back to main menu
                }
                if(c == 'b') {
                        cout << "you chose 'b'" << endl;
                        continue; // Back to main menu
                }
                else {
                        cout << "you didn't choose a or b!" << endl;
                        continue; // Back to main menu
                }
        }
```

2/2

```
if(c == 'r') {
                        cout << "RIGHT MENU:" << endl;
                        cout << "select c or d: ";
                        cin >> c;
                        if(c == 'c') {
                                    cout << "you chose 'c'" << endl;
                                    continue; // Back to main menu
                        }
                        if(c == 'd') {
                                              cout << "you chose 'd'" << endl;
                                    continue; // Back to main menu
                        }
                        else {
                                    cout << "you didn't choose c or d!"
                                              << endl;
                                    continue; // Back to main menu
                        }
            }
            cout << "you must type l or r or q!" << endl;
}
cout << "quitting menu..." << endl;
```

Έξοδος από βρόχο

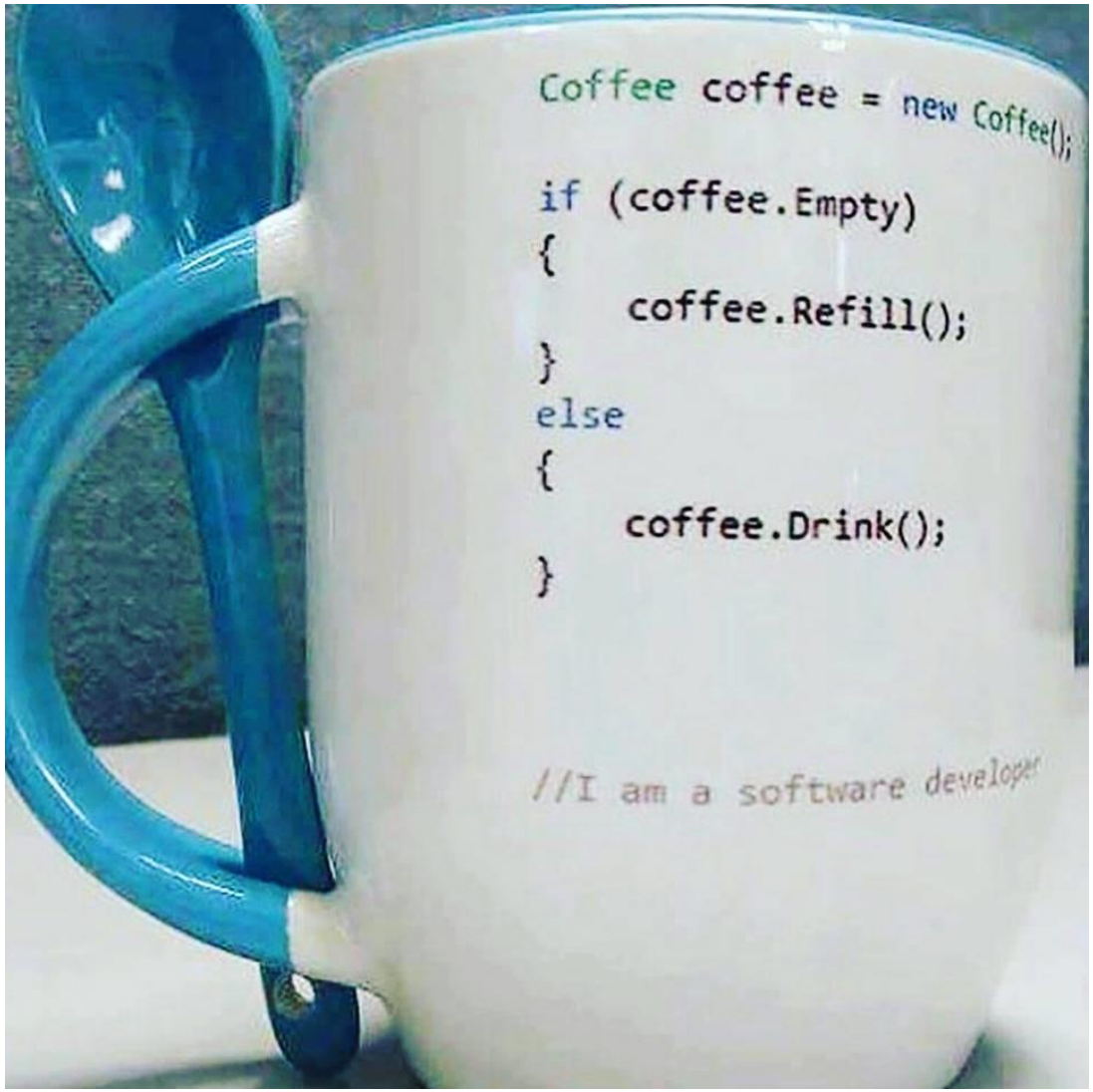
```
#include <stdio.h>
int main() /* Tameiakh mhxanh*/
{
    float sum = 0, SUM = 0, product = 0;
    int plh8os, id = 1;
    cout << “Give -1 for false,”;
    cout << “-2 for next customer”;
    cout << “and -3 for exit”;;
    while (1)
    {
        cout ("Enter product price :");
        cin >> product;

        if (product == -1)
        {
         cout << “Enter customer ” << id;
           SUM = SUM - sum;
           sum = 0;
           continue;
        }
        else if (product == -2)
        {
          cout <<“sum”<<id<<“=”<<sum;
          id++;
          sum = 0;
          continue;
        }
        else if (product == -3)
        {
          cout << “Balance of ” << id-1
    << “ = ” << SUM;
          break;
        }

        cout << "Enter item #:";
        cin >> plh8os;

        sum += plh8os* product;
        SUM += plh8os* product;
    }
    return 0;
}
```

Που είναι το λάθος και πως διορθώνεται;



Προβλήματα

Πρόβλημα Εύρεσης Παραγοντικού

- Σχεδιάστε αλγόριθμο ο οποίος υπολογίζει το παραγοντικό ενός θετικού ακέραιου αριθμού (>0) που δίνει ο χρήστης.

- **Δεδομένα εισόδου**: αριθμός x
- **Δεδομένα εξόδου** : το παραγοντικό
- Π.χ. 1! = 1, 2! = 1.2 = 2, 3! = 1.2.3 = 6
- 4!=1.2.3.4=24, 5!=1.2.3.4.5=120, ….

Λύση 1: Εύρεσης Παραγοντικού με **for loop**

```
int main()
{
    int i, factorial=1;
    int n;

    cout << "Enter Number (1-12):";
    cin >> n;

    for (i=1; i<=n; i++)
    {
      factorial *= i;
    }
    cout << “The factorial of” << n << “ is ” << factorial;
    return 0;
}
```

Λύση 2: Εύρεσης Παραγοντικού με έλεγχο δεδομένων εισόδου

```
int main()
{
    int i, factorial=1;
    int n;
    do                                  /* Έλεγχος ότι τα δεδομένα εισόδου είναι ορθά */
    {
      cout << "Enter Number (1-12):";
      cin >> n;
    }
    while (n<1 || n>12);
    for (i=1; i<=n; i++) {
      factorial *= i;
    }
    cout << “The factorial of ” n << “ is ” << factorial;
    return 0;
}
```

Λύση 3: Εύρεσης Παραγοντικού με **while loop**

```
int main()
{
    int i=1, factorial=1;
    int n;

    do                                  /*΄Ελεγχος ότι τα δεδομένα εισόδου είναι ορθά*/
    {
        cout << "Enter Number (1-12):";
        cin >> n;
    }
    while (n<1 || n>12);

    while (i<=n)
    {
        factorial *= i;
        i++;
    }
    cout << “The factorial of ” n << “ is ” << factorial;
    return 0;
}
```

Λύση 3: Εύρεσης Παραγοντικού με **do while loop**

```
main()
{
    int i=1, factorial=1;
    int n;
    do                                  /* Έλεγχος ότι τα δεδομένα εισόδου είναι ορθά*/
    {
      cout << "Enter Number (1-12):";
      cin >> n;
    }
    while (n<1 || n>12);
    do
    {
      factorial = factorial * i;
      i++;
    }
    while (i<=n)
    cout << “The factorial of ” n << “ is ” << factorial;
    return 0;
}
```

Εμφωλευμένοι (Nested) Βρόχοι :

```
int i=0;
while(i<5)
{
   int j=0;
   while(j<5)
   {
      cout << i << "-" << j;
      j++;
   }
   cout<<endl;
   i++;
}
```

Αποτέλεσμα:
0-0 0-1 0-2 0-3 0-4
1-0 1-1 1-2 1-3 1-4
2-0 2-1 2-2 2-3 2-4
3-0 3-1 3-2 3-3 3-4
4-0 4-1 4-2 4-3 4-4

```
for (i=0; i <5; i++)
{
        for (j=0; j < i; j++)
                cout << i*j;
        cout.put('\n');
}
```

- Πόσες φορές θα εκτελεστεί το 1ο cout;

  0+1+2+3+4 = 10
- Πόσες φορές θα εκτελεστεί το 2ο cout;

  5
- Ποια θα είναι η τελευταία τιμή που θα εκτυπωθεί;

  4*3=12

Πόσες φορές θα εκτελεστει;

```
for (i=0; i <= MAX1; i++)
   for (j=0; j < MAX2; j++)
      cout << "****************";
```

1) MAX1*MAX2
2) (MAX1+1)*(MAX2+1)
3) (MAX1+1)*MAX2
4) MAX1*(MAX2+1)

Example

```
for (i=0;i<50;i++)
{
    for (j=0;j<50;j++)
    {
        char c;
        double distance,dx,dy;

        dy = 25-i;
        dx = 25-j;
        distance = sqrt(dx*dx + dy*dy);
        if (distance < 25)
            c = '*';
        else
            c = ' ';
        cout.put(c);
    }
    cout.put('\n');
}
```

τρίγωνο

```
int i=0,j;
while(i<5)
{
    j=0;
    while(j<=i)
    {
        cout.put(i);
        cout.put('-');
        cout.put(j);
        j++;
    }
    cout.put('\n');
    i++;
}
```

Αποτέλεσμα:
0-0
1-0 1-1
2-0 2-1 2-2
3-0 3-1 3-2 3-3
4-0 4-1 4-2 4-3 4-4

Εμφωλευμένοι βρόχοι

```
for (i=1; i<=8; i++)
{
    for (j = 1; j <= i; ++j)
    {
        cout.put('*');
    }
    cout.put('\n');
}
```

Τι είναι μια «αναλογική» και «μη-αναλογική» γραμματισειρά;

```
*
**
***
****
*****
******
*******
********
```

Φωλιασμένοι βρόχοι

/*Πως θα εκτυπώναμε το παρακάτω σχήμα;*/

```
*                    *
**                  **
***                ***
****              ****
*****            *****
******          ******
*******        *******
**************************
```

```
for (i = 0; i < 10; i++)
    /* loop body */

i = 0;
for (; i < 10;)
{
    i++;
}

i = 0;
for (;; i++)
{
    if (i >= 10)
        break;
}
```