Παράδειγμα 1

Το παραγοντικό ενός μη αρνητικού ακέραιου αριθμού *n* ορίζεται με αναδρομικό τρόπο από τις σχέσεις:

$$0!=1 \qquad \text{και} \qquad n!=n(n-1)! \quad \text{για } n>0.$$

Να γράψετε πρόγραμμα που ζητάει από το χρήστη να εισάγει ένα μη αρνητικό ακέραιο αριθμό *n* και υπολογίζει και εκτυπώνει το παραγοντικό του.

```
#include<iostream>
using namespace std;
long int factorial(int);
int main ( )
{
int n;
long int fact;
cout<<“\nAssign value to n :”;
cin>>n;
while (n<0)
{
    cout<<“\n The number n should be > 0”;
    cout<<“\nAssign again value to n:”;
    cin>>n;
}
```

```
fact=factorial(n);
cout<<"\nThe factorial of n is : "<<fact;
return 0;
}
long int factorial(int k)
{
long int f;
if (k==0)
    f=1;
else
    f=k*factorial(k-1);
return(f);
}
```

Παράδειγμα (άσκηση 10.1)

Να γράψετε πρόγραμμα που θα ζητάει από το χρήστη 30 ζεύγη ακέραιων θετικών αριθμών και θα τα αποθηκεύει σε ένα πίνακα Α 2x30 θέσεων, πχ το πρώτο ζεύγος θα το αποθηκεύει στα στοιχεία Α[0][0] και Α[1][0], κ.λ.π. Στη συνέχεια θα δημιουργεί και θα εκτυπώνει τον πίνακα Β 3x30 του οποίου οι δύο πρώτες γραμμές θα ταυτίζονται με τον πίνακα Α και η τρίτη θα ορίζεται από τον τύπο Β[2][i]=Μ.Κ.Δ.(Α[0][i],Α[1][i]) για κάθε i=0,…,29. Να χρησιμοποιήσετε συνάρτηση για την εύρεση των Μ.Κ.Δ.

```
#include<iostream>
using namespace std;
int mkd(int, int);
int main()
{
    int A[2][30],B[3][30],i,j;
```

```
for (j=0;j<30;j++)
{
        cout<<"\n Assign a positive value to A[0]["<<j<<"] :";
        cin>>A[0][j];
        while (A[0][j]<0)
        {
                cout<<"\n A[0]["<<j<<"] must be positive";
                cout<<"\nAssign again value to A[0]["<<j<<"] :";
                cin>>A[0][j];
        }
        cout<<"\n Assign a positive value to A[1]["<<j<<"]";
        cin>>A[1][j];
        while (A[1][j]<0)
        {
                cout<<"\n A[1]["<<j<<"] must be positive";
                cout<<"\nAssign again value to A[1]["<<j<<"] :";
                cin>>A[1][j];
        }
}
```

```
    for (i=0;i<2;i++)
    {
        for (j=0;j<30;j++)
            B[i][j]=A[i][j];
    }
    for (j=0;j<30;j++)
        B[2][j]=mkd(A[0][j],A[1][j]);
    for (i=0;i<3;i++)
    {
        cout<<endl;
        for (j=0;j<30;j++)
            cout<<B[i][j]<<“   ”;
    }
    return 0;
}
```

```
int mkd(int x, int y)
{
    int m;
    while(x!=y)
    {
    if (x>y)
        x=x-y;
    else
        y=y-x;
    }
    m=x;
    return (m);
}
```

Παράδειγμα (άσκηση 10.3)

Να γράψετε πρόγραμμα που θα υπολογίζει με τη βοήθεια συναρτήσεων τον *n-οστό* όρο της αναδρομικής ακολουθίας

$$x_n=(n-1)!x_{n-1}+(n-2)!x_{n-2}, \quad n>2.$$

Οι δύο πρώτοι όροι x_1,x_2 θα καθορίζονται από το χρήστη.

```
#include<iostream>
using namespace std;
long int factorial(int);
long int rec(int, int, int);
int main()
{
    int x1,x2,n;
    long int x;
    cout<<"\nAssign value to the first term :";
    cin>>x1;
    cout<<"\nAssign value to the second term :";
    cin>>x2;
```

```
    cout<<"\Assign a positive value to n :";
    cin>>n;
    while (n<=0)
    {
        cout<<"\n The number n must be positive";
        cout<<"\nAssign again value to n :";
        cin>>n;
    }
    x=rec(x1,x2,n);
    cout<<"\nThe "<<n<<"term of the
                                    sequence is :"<<x;
    return 0;
}
```

```
long int factorial(int k)
{
    long int f;
    if (k==0)
        f=1;
    else
        f=k*factorial(k-1);
    return(f);
}
```

```
long int rec(int y1, int y2, int m)
{
    long int y;
    if (m==1)
        y=y1;
    else if (m==2)
        y=y2;
    else
        y=factorial(m-1)*rec(y1,y2,m-1)+
                                factorial(m-2)*rec(y1,y2,m-2);
    return(y);
}
```

Παράδειγμα 2

Να γράψετε πρόγραμμα που θα διαβάζει ένα 5x5 πίνακα *A* θετικών ακεραίων μικρότερων ή ίσων του 20 (απαιτείται έλεγχος ορθότητας δεδομένων) και θα δημιουργεί τον 5x5 πίνακα *B* του οποίου τα στοιχεία δίνονται από τον τύπο

$$B_{ij} = \sum_{1 \le k \le j} A_{ik} \,!$$

για κάθε 1≤i,j≤5.

Ο υπολογισμός των παραγοντικών θα γίνεται με τη χρήση συνάρτησης. Το πρόγραμμα θα τυπώνει και τους δύο πίνακες.

```
#include<iostream>
using namespace std;
long int factorial(int);
int main()
{
    int A[5][5], i,j,k;
    long int B[5][5];
    for (i=0;i<5;i++)
    {
        for (j=0;j<5;j++)
        {
            do
            {
                cout<<"\nAssign a value greater than 0
                    and lower than 21 to the ("<<i+1<<","
                     <<j+1<<") element of A :";
                cin>>A[i][j];
            }
            while(A[i][j]<0 || A[i][j]>20);
        }
    }
```

```
for (i=0;i<5;i++)
{
        for (j=0;j<5;j++)
        {
                B[i][j]=0;
                for (k=0;k<=j;k++)
                        B[i][j]=B[i][j]+factorial(A[i][k]);
         }
 }
 cout<<"\n The matrix A  \n\n\n";
 for (i=0;i<5;i++)
{
        cout<<endl;
        for (j=0;j<5;j++)
                cout<<A[i][j]<<"        ";
 }
```

```
    cout<<"\n \n\n The matrix B  \n\n\n";
    for (i=0;i<5;i++)
   {
        cout<<endl;
        for (j=0;j<5;j++)
                cout<<B[i][j]<<"        ";
    }
   return 0;
}
long int factorial(int m)
{
   long int f;
   if (m==0)
        f=1;
   else
        f=m*factorial(m-1);
   return(f);
}
```

Παράδειγμα 3

Γράψτε πρόγραμμα που να εκτυπώνει το αποτέλεσμα υπολογισμού μιας από ένα σύνολο συναρτήσεων (τετράγωνο, κύβος, ύψωση σε δύναμη, αντίστροφος, λογάριθμος, απόλυτη τιμή) της μεταβλητής εισόδου x, κατόπιν αντίστοιχης επιλογής του χρήστη.

```
#include<iostream>
#include<math.h>
using namespace std;
int main()
{
        int choice, n;
        float x, y;

        cout<<"\n enter value of x:";
        cin>>x;
        cout<<"Choose function to calculate:\n";
        cout<<"Enter 1 for square of x,\n";
        cout<<"Enter 2 for cube of x,\n";
        cout<<"Enter 3 for power of x,\n";
        cout<<"Enter 4 for inverse of x,\n";
```

```
cout<<"Enter 5 for logarithm of x,\n";
cout<<"Enter 6 for absolute value of x,\n";
cout<<"choice=";
cin>>choice;
if (choice==1)
        y=pow(x,2);
else
        if (choice==2)
                y=pow(x,3);
        else
                if (choice==3)
                {
                        cout<<"Enter exponent:";
                        cin>>n;
                        cout<<"\n";
```

```
                y=pow(x,n);
        }
        else
                if (choice==4)
                {
                        if (x!=(0.0))
                                y=1.0/x;
                        else
                        {
                                cout<<"Can not
calculate inverse of zero!\n";
                                return -1;
                        }
                }
                else
```

```
if (choice==5)
{
        if (x>(0.0))
                y=log(x);
        else
        {
                cout<<"Can not calculate logarithm of non positive number!\n";
                return -1;
        }
}
else
        if (choice==6)
                y=fabs(x);
        else
```

```
                                        {

        cout<<"invalid choice "<<choice<<", exiting without
calculation\n";
                                                return -1;
                                        }
        cout<<"result y="<<y<<"\n";
        return 0;
}
```

Επιπλέον προτάσεις-εντολές ελέγχου ροής προγράμματος

- Switch
- Βοηθητικές εντολές:
  - break
  - continue
  - goto

Εντολή switch

Υλοποιεί εκτέλεση κώδικα «υπό συνθήκη» αναλόγως της τιμής μιας έκφρασης ελέγχου

```
switch (expression) {
case const expr 1 :
statements 1
case const expr 2 :
statements 2
...
case const expr n :
statements n
default:
statements
}
```

- Η έκφραση ελέγχου πρέπει να είναι ακέραιου ή διακριτού τύπου (ή δυνάμει διακριτού τύπου π.χ. Bool
- Οι εκφράσεις κάθε περίπτωσης (case) πρέπει να είναι σταθερές (να επιδέχονται αποτίμησης κατά το χρόνο μεταγλώττισης).
- Το πρόγραμμα εκτελεί τις εντολές της περίπτωσης που αντιστοιχεί στην έκφραση ελέγχου
- ΚΑΙ ΣΥΝΕΧΙΖΕΙ ΣΤΗΝ ΕΠΟΜΕΝΗ περίπτωση, εκτός κι αν υπάρχει εντολή αλλαγής ροής (break, goto, return, …)

Εντολή break

Χρησιμοποιείται **<u>μόνο</u>** α) με την εντολή switch β) εντός μπλοκ εντολών βρόγχου (loop) for, while, do-while

Ο έλεγχος ροής προγράμματος μεταφέρεται στην επόμενη εντολή από αυτή που περιέχει την εντολή break.

Παράδειγμα break

```
…
int i;
while (true) {
cout << "Δώσε ένα θετικό ακέραιο: ";
cin >> i;
cout << '\n';
if (i > 0) {
break;
}
cout << “αρνητικός ακέραιος, δοκίμασε ξανά\n";
}
…
```

Εντολή continue

Χρησιμοποιείται **<u>μόνο</u>** εντός μπλοκ εντολών βρόγχου (loop) for, while, do-while

Ο έλεγχος ροής προγράμματος μεταφέρεται στο τέλος του μπλοκ που περιέχει την εντολή break και εκτελείται η επόμενη επανάληψη του μπλοκ (αν αυτό προβλέπεται από την αντίστοιχη τιμή δείκτη ή συνθήκη ελέγχου).

Παράδειγμα continue

```
…
int i, j=0;
for (i=0; i < 1000; ++i) {
if ((i %5) != 0) {
continue;
}
j=j+i;
cout << “i=“i<<“ j=“<<j<< “\n”;
}
…
```

Εντολή goto

Δηλώνουμε μια ετικέτα (label) η οποία αντιστοιχεί σε κάποια εντολή ως εξής

labelname: statement;

…

goto labelname;

ΣΗΜΕΙΩΣΗ: περιττή και δυνητικά επικίνδυνη εντολή, χρησιμοποιείτε μετά προσοχής!

Παράδειγμα 3Β

```
#include<iostream>
#include<math.h>
using namespace std;

int main()
{
        int choice, n;
        float x, y;
```

```
cout<<"\n enter value of x:";
cin>>x;
cout<<"Choose function to calculate:\n";
cout<<"Enter 1 for square of x,\n";
cout<<"Enter 2 for cube of x,\n";
cout<<"Enter 3 for power of x,\n";
cout<<"Enter 4 for inverse of x,\n";
cout<<"Enter 5 for logarithm of x,\n";
cout<<"Enter 6 for absolute value of x,\n";
cout<<"choice=";
cin>>choice;
```

```
switch (choice) {
        case 1:
                y=pow(x,2);
                break;
        case 2:
                y=pow(x,3);
                break;
        case 3:
                cout<<"Enter exponent:";
                cin>>n;
                cout<<"\n";
```

```
                y=pow(x,n);
            break;
        case 4:
            if (x!=(0.0))
                y=1.0/x;
            else
            {
                cout<<"Can not calculate
inverse of zero!\n";
                return -1;
            }
```

```
                break;
        case 5:
                if (x>(0.0))
                        y=log(x);
                else
                {
                        cout<<"Can not calculate
logarithm of non positive number!\n";
                        return -1;
                }
                break;
```

```
            case 6:
                y=fabs(x);
                break;
            default:
                cout<<"invalid choice
"<<choice<<", exiting without calculation\n";
                return -1;
        }
        cout<<"result y="<<y<<"\n";
        return 0;
}
```