Εισαγωγή στον Προγραμματισμό

Ενότητα C++ (75%) :

11η εβδομάδα

Παναγιώτης Τζουνάκης

(βασισμένο στις σημειώσεις που ευγενικά προσέφερε ο καθηγητής κ. Γεώργιος Ραχώνης)

Άνοιξη 2026

Συναρτήσεις

Παράδειγμα 0 (με επαναληπτικές δομές)

Να γράψετε πρόγραμμα που θα διαβάζει τους αριθμούς *n*, *m* και θα υπολογίζει τον αριθμό συνδυασμών *n* στοιχείων ανά *m*

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} \qquad n \geq m \geq 1$$

Συναρτήσεις

Σύνταξη

```
τύπος_συνάρτησης όνομα_συνάρτησης(δηλώσεις
    ορισμάτων)
{
    δηλώσεις μεταβλητών;
    προτάσεις;
}
```

όπου οι «δηλώσεις ορισμάτων» είναι κατ' αναλογία με τις «δηλώσεις μεταβλητών».

Παράδειγμα 1 (με συναρτήσεις C++)

Να γράψετε πρόγραμμα που θα διαβάζει τους αριθμούς *n*, *m* και θα υπολογίζει με τη βοήθεια συνάρτησης τον αριθμό συνδυασμών *n* στοιχείων ανά *m*

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} \qquad n \geq m \geq 1$$

```
#include<iostream>
using namespace std;
long int factorial(int);      //Πρότυπο συνάρτησης
int main ( )
{
   int n, m;
   long int f1, f2, f3, bifactorial;
   cout<<“\nAssign value to n :”;
   cin>>n;
   cout<<“\nAssign value to m :”;
   cin>>m;
```

```
    f1=factorial(n);    //Αντιστοιχεί τον n με τον k
    f2=factorial(m);
    f3=factorial(n-m);
    bifactorial=f1/(f2*f3);
    cout<<"\nThe result is :"<<bifactorial;
    return 0;
}
long int factorial(int k)
{
    int i;
    long int f;
    if (k==0)
        f=1;
```

```
    else
    {
        f=1;
        for (i=1; i<=k; i++)
            f=f*i;
    }
    return(f);          //Επιστρεφόμενη τιμή
}
```

Παράδειγμα 2

Να γράψετε πρόγραμμα που θα διαβάζει τους αριθμούς *n, m* και θα υπολογίζει με τη βοήθεια δύο συναρτήσεων τον αριθμό συνδυασμών *n* στοιχείων ανά *m*

$$\binom{n}{m} = \frac{n!}{m!(n-m)!} \qquad n \geq m \geq 1$$

```
#include<iostream>
using namespace std;
long int factorial(int);          //Πρότυπο συνάρτησης
long int bifactorial(int, int); //Πρότυπο συνάρτησης
int main ( )
{
    int n, m;
    long int p;
    cout<<“\nAssign value to n :”;
    cin>>n;
    cout<<“\nAssign value to m :”;
    cin>>m;
```

```
    p=bifactorial(n,m);
    cout<<"\nThe result is :"<<p;
    return 0;
}
long int factorial(int k)
{
    int i;
    long int f;
    if (k==0)
        f=1;
```

```
    else
    {
        f=1;
        for (i=1; i<=k; i++)
            f=f*i;
    }
    return(f);          //Επιστρεφόμενη τιμή
}
long int bifactorial(int a, int b)
{
    long int g;
    g=factorial(a)/(factorial(b)*factorial(a-b));
    return(g);                  //Επιστρεφόμενη τιμή
}
```

Παράδειγμα 3

Να γράψετε πρόγραμμα που θα ζητάει από το χρήστη 30 ζεύγη ακέραιων θετικών αριθμών και θα εκτυπώνει τους αριθμούς του κάθε ζεύγους και το Μ.Κ.Δ. των αριθμών του κάθε ζεύγους.

Τροποποιήστε το πρόγραμμα ώστε ο χρήστης να επιλέγει τον αριθμό των ζευγών στην αρχή.

```
#include<iostream>
using namespace std;
int mkd(int, int);
int main()
{
    int i,a,b;
    for (i=1;i<=30;i++)
    {
        cout<<"\nAssign values to the  "<<i<<"pair
                                    of numbers";
        cin>>a;
        cin>>b;
        cout<<"\n\n\nThe values of the "<<i<<"pair are"
                                    <<a<<" and "<<b;
        cout<<"\n\nThe gcd of ("<<a<<","<<b<<") is "<<
                        mkd(a,b);
    }
    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);
}
```

Άσκηση 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);
}
```

Άσκηση 2

Να γράψετε πρόγραμμα το οποίο θα διαβάζει 10 ζεύγη φυσικών αριθμών a, b (απαιτείται έλεγχος ορθότητας δεδομένων) και με την βοήθεια συνάρτησης θα επιστρέφει την λύση της εξίσωσης ax+b=0.

Παράδειγμα 4

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

$$x_n = ax_{n-1} + bx_{n-2}, \quad n>2.$$

Οι σταθερές *a, b* και οι δύο πρώτοι όροι x_1, x_2 θα καθορίζονται από το χρήστη.

Για a=b=1, n>1, η πασίγνωστη ακολουθία Fibonacci

βλ., https://en.wikipedia.org/wiki/Fibonacci_sequence

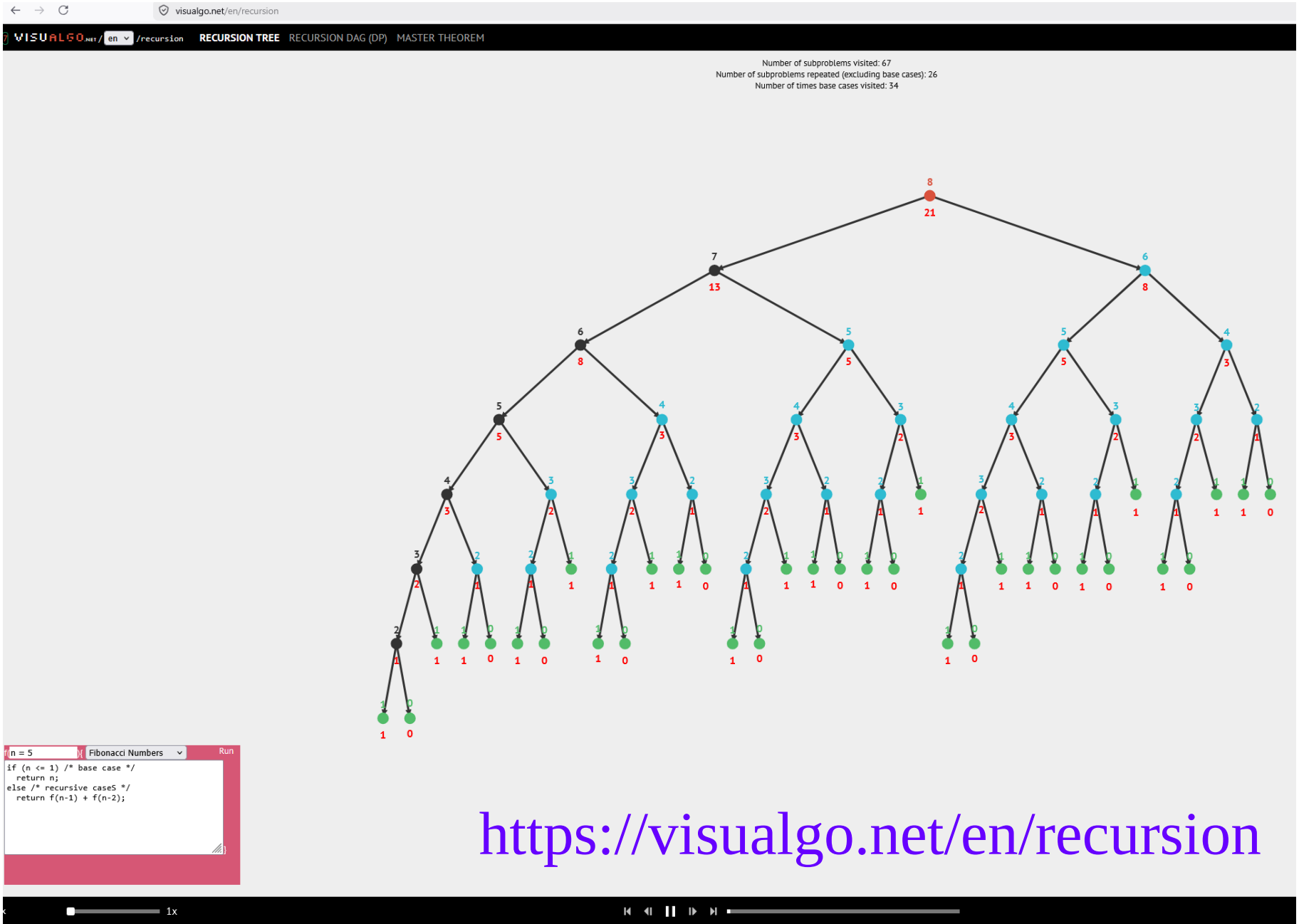
```
#include<iostream>
using namespace std;
int rec(int, int, int, int, int);
int main()
{
    int a,b,x1,x2,x,n;
    cout<<“\nAssign value to a:”;
    cin>>a;
    cout<<“\nAssign value to b:”;
    cin>>b;
```

```
cout<<“\nAssign value to the first term:”;
cin>>x1;
cout<<“\nAssign value to the second term:”;
cin>>x2;
cout<<”\nAssign value to n:”;
cin>>n;
while (n<=2)
{
    cout<<“\n n must be greater than 2”;
    cout<<“\n Assign value to n:”;
    cin>>n;
}
```

```
    x=rec(a,b,x1,x2,n);
    cout<<"\nThe "<<n<<" term of the sequence is :"
                                                <<x;
    return 0;
}
int rec(int c, int d, int y1, int y2, int m)
{
    int y;
    if (m==1)
        y=y1;
```

```
    else if (m==2)
          y=y2;
    else
    y=c*rec(c,d,y1,y2,m-1)+d*rec(c,d,y1,y2,m-2);
    return(y);
}
```

Οπτικοποίηση αναδρομικών κλήσεων functions



https://visualgo.net/en/recursion

Άσκηση 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 5

Να γράψετε πρόγραμμα που θα διαβάζει έναν αριθμό και θα υπολογίζει με τη βοήθεια συναρτήσεων και έμμεση αναδρομή αν είναι άρτιος ή περιττός.

Όπου στην έμμεση αναδρομή, δύο (ή περισσότερες) συναρτήσεις καλούν η μια την άλλη κυκλικά.

```
#include <iostream>
using namespace std;
// Forward declaration so isOdd can be called from isEven
bool isOdd(int n);
// Returns true if n is even, false otherwise
bool isEven(int n) {
    if (n == 0)
        return true;          // Base case: 0 is even
    else
        return isOdd(n - 1);  // Indirect call to isOdd
}
```

```
// Returns true if n is odd, false otherwise
bool isOdd(int n) {
    if (n == 0)
        return false;          // Base case: 0 is not odd
    else
        return isEven(n - 1); // Indirect call to isEven
}
int main() {
    int num;
    cout << "give number:" ;
    cin >> num;    cout << num << " is even? " <<
    boolalpha << isEven(num) << endl;
    cout << num << " is odd?  " << boolalpha <<
    isOdd(num) << endl;
    return 0;
}
```

https://en.cppreference.com/cpp/io/manip/boolalpha