Cursul 13. Operaţii de intrare/ieşire în C++

(*plan de curs*)

1. Generalităţi despre clase în C++

```
#include <iostream>
using namespace std;

class Punct{
public:
    double x, y;
    static char * mesaj;
    Punct(double xx=13, double yy=13); //constructor
    Punct opus();

};
char* Punct::mesaj = "punctul";
Punct::Punct(double xx, double yy){
    x = xx;
    y = yy;
}
Punct Punct :: opus(){
    return Punct(-x, -y);

}
ostream& operator<< (ostream& xout, const Punct& p){
    xout << Punct::mesaj<<"("<< p.x << "," << p.y<<") ";
    return xout;
}
class PunctNumit : public Punct{
public:
    char nume;
    PunctNumit(Punct a, char ch){
        x = a.x;
        y = a.y;
        nume = ch;
    }
    PunctNumit(double x, double y, char ch){
        this->x = x;
        this->y = y;
        this->nume = ch;
    }
};
ostream& operator<< (ostream& xout, const PunctNumit& p){
    xout << Punct::mesaj <<" "<<p.nume<< "(" << p.x << "," << p.y << ") ";
    return xout;
}
template <class TIP> class PunctGeneric{
public:
    TIP x, y;
    static char * mesaj;
    PunctGeneric(TIP xx, TIP yy){
        x = xx;
        y = yy;
    }
    PunctGeneric opus(){
        return PunctGeneric(-x,-y);
    }
};
```

```
typedef PunctGeneric<double> Punctt;
//identificatorul Punctt este un sinonim cu specializarea
//sablonului PunctGeneric pentru tipul double

int main()
{
    Punct a;
    cout << a.x << endl; //13
    cout << a.mesaj << endl;//punctul
    cout << "a=" << a << endl;//a=punctul(13,13)
    cout << Punct::mesaj << endl; //punctul
    //cout << Punct::x << endl;//Error: illegal reference to non-static member 'Punct::x'

    Punct b;
    b = Punct(1, 2);
    cout << b.opus().x << endl; //-1

    Punct c(11, 22);

    cout << c;  //punctul(11,22)
    cout << endl;
    operator<<(cout, c);    //punctul(11,22)

    endl(cout);//endl este o functie:
    /*  ostream& endl(ostream& os)
    {
        os << '\n';
        os.flush();
        return os;
    }
    */
    cout.operator << (endl);  // apelul metodei operator<< a clasei ostream
    /*ostream& ostream::operator<<(ostream& (*m)(ostream&))
    {
        return (*m)(*this);
    }
    */
    cout << b << c<< endl;
    (operator<<(operator<<(cout, b), c)).operator<<(endl);

    Punct q(111, 222);
    PunctNumit qA(q, 'A');
    cout << qA << endl; //punctul A(111,222)

    PunctGeneric<int> aaa(1, 2);
    cout << aaa.opus().x << endl;//-1
    Punctt z(1.1, 2.2);
    cout << z.x << endl; //1.1

    return 0;
}
```

Exemplu de clasă şablon şi de specializări:

```
template <class CharT, class Traits = char_traits<class CharT>> class basic_ostream :
virtual public ios_basic<CharT, Traits>{...};

typedef basic_ostream<char, char_traits<char>> ostream;
typedef basic_ostream<wchar_t, char_traits<wchar_t>> wostream;
```

2. Stream-urile **cin** şi **cout**

```
#include<iostream>
using namespace std;
int main(){
    double a = 1.0 / 7.0;
    int old_prec, new_prec = 16;
    cout << "a=" << a << endl;
    old_prec = cout.precision(new_prec);
    cout << "vechea afisare era cu " << old_prec << " cifre" << endl;
    cout << "acum afisam cu " << new_prec << " cifre:" << endl;
    cout << "a=" << a << endl;
    return 0;
}

//a = 0.142857
//vechea afisare era cu 6 cifre
//acum afisam cu 16 cifre :
//a = 0.1428571428571429
//Press any key to continue .
```

3. Citirea şi scrierea fişierelor pe disc. Prezentare rapidă

```
#include<iostream>  //pentru cout
#include<fstream>   //pentru ofstream
using namespace std;

int main(){
    ofstream xout("felicitare.txt");
    xout << "Sa traiti bine!" << endl;
    cout << "Am scris." << endl;
    return 0;
}
```

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
    ifstream xin("felicitare.txt");
    char ch;
    do{
        xin >> ch;
        cout << ch;
    } while (ch != '!');//
    cout << "\nAm citit si am scris." << endl;
    return 0;
}

//Satraitibine!
//Am citit si am scris.
//Press any key to continue . . .
```

```
#include<iostream>
#include<fstream>
using namespace std;
int main(){
    ifstream xin("felicitare.txt");
    int ch;
    while ((ch = xin.get()) != EOF)
        cout << (char)ch;
    cout << "\nAcum e OK!." << endl;
    return 0;
}
//Sa traiti bine!
//
//Acum e OK!.
//Press any key to continue . . .
#include<iostream>
#include<fstream>
using namespace std;
int main(){
    ifstream xin("felicitare.txt");
    char ch;
    while (!xin.get(ch).eof())
        cout << ch;
    cout << "\nGata." << endl;
    return 0;
}

//Sa traiti bine!
//
//Gata.
//Press any key to continue . . .
```

4. Citirea şi scrierea fişierelor pe disc. Prezentare în amănunt.

Exemplu: şirul (x_n), dat de recurenţa

$$x_{n+1} = 4x_n(1 - x_n),$$

rămâne în intervalul [0,1] dacă x_0 este din [0,1].

```
#include<iostream>
#include<fstream>
#include<assert.h>          //pentru assert()
#include<stdlib.h>          //pentru atof()
using namespace std;

//void pauza()
//{   //system("pause");//Press any key to ...
//    char ch;
//    cout << "\n\tPauza!\n\nPentru start tastati [ENTER]\n" << endl;
//    do{
//        cin.get(ch); //cin>>ch nu "simte" tasta ENTER
//    }
//    while (ch != '\n');
//}
```

```
void pauza()
{
    char ch;
    cout << "\n\tPauza!\n\nPentru start tastati [ENTER]\n" << endl;
    cin.get(ch);            //cin>>ch nu "simte" tasta ENTER
    cin.putback(ch);        //refacem buffer-ul tastaturii
    cin.ignore(4096, '\n'); //si apoi il golim
}

void creareFisier(char caleFis[], double x, int nrMax){
    ofstream xout(caleFis);
    assert(xout.good());
    cout << "Asteptati va rog!" << endl;
    for (int i = 0; i < nrMax; i++){
        xout << "x[" << i << "]=" << x << endl;
        x = 4.0*x*(1.0 - x);
    }
    cout << "Gata, am creat fisierul " << caleFis << endl;
    xout.close();
}

void afisareFisier(char caleFis[]){
    ifstream xin(caleFis);
    assert(xin.good());
    const int dim = 1024;
    char buffer[dim];
    cout << "Iata fisierul " << caleFis << ":" << endl;
    while (xin.good()){
        xin.getline(buffer, dim);
        cout << buffer << endl;
    };              //Atentie: la final avem un buffer gol!
    cout << "Gata, acesta a fost fisierul " << caleFis << endl;
    xin.close();
    return;
}

void consultareFisier(char caleFis[]){
    ifstream xin(caleFis);
    assert(xin.good());
    const int dim = 100;
    int i, j;
    double x;
    char buffer[dim], bufNum[dim];
    int cont[11] = {};
    while (xin.good()){
        xin.getline(buffer, dim);
        if (buffer[0] == '\0') break;    //nu analizam buffer-ul gol.
        for (i = 0; i < dim && buffer[i++] != '=';);
        for (j = 0; i + j < dim && (bufNum[j] = buffer[i + j]) != '\0'; j++);
        x = atof(bufNum);
        cont[(int)(10 * x)]++;
    }
    cout << "In fisier avem: " << endl;
    for (i = 0; i < 10; i++) cout << cont[i] << " numere in [ " << i / 10.0
        << " , " << (i + 1) / 10.0 << " )" << endl;
    xin.close();
    return;
}
```

```
int main(){
    char nume[] = "numere.txt";
    int nrMax = 1000;
    double x0 = 0.12345;

    creareFisier(nume, x0, nrMax);
    pauza();

    afisareFisier(nume);
    pauza();

    /*
    x[0]=0.12345
    x[1]=0.43284
    x[2]=0.981958
    x[3]=0.0708646
    ............
    x[999]=0.434603
    */

    consultareFisier(nume);
    /*
    In fisier avem:
    217 numere in [ 0 , 0.1 )
    81 numere in [ 0.1 , 0.2 )
    81 numere in [ 0.2 , 0.3 )
    60 numere in [ 0.3 , 0.4 )
    60 numere in [ 0.4 , 0.5 )
    62 numere in [ 0.5 , 0.6 )
    74 numere in [ 0.6 , 0.7 )
    76 numere in [ 0.7 , 0.8 )
    92 numere in [ 0.8 , 0.9 )
    197 numere in [ 0.9 , 1 )
    Press any key to continue . . .
    */
    return 0;
}
```