LABORATOR IX
STREAMURI DE INTRARE-IEŞIRE

IX.1 SCOPUL LUCRĂRII

Lucrarea are ca scop ilustrarea utilizării streamurilor de intrare ieşire în programe orientate obiect. În primul rând sunt ilustrate elemente ale bibliotecilor corespunzătoare ale limbajului C++, principiile rămînând valabile în general. Nu în ultimul rând, structura bibliotecii IO este un bun exemplu de dezvoltare a unei arhtecturi de clase. Atenţie la relaţiile de derivare şi la moşteniri.

IX.2 BIBLIOECA DE STREAMURI C++

Implementarea strategiilor pentru realizarea operaţiilor de intrare ieşire pentru un program poate reprezenta una din problemele dificile cu care se confruntă proiectantul aplicaţiei. Bibliotecile IO sunt foarte complexe şi conţin implementează multe funcţionalităţi. De multe ori poate fi necesară utilizarea unor soluţii IO care se pot implementa numai prin extinderea funcţionalităţilor furnizate de biblioteci.

De fiecare dată când este nevoie consultaţi documentaţia bibliotecilor IO pe care le utilizaţi. În această lucrare de laborator este prezentată sumar structura de clase utilizată pentru implementarea operaţiilor IO, precum şi câteva exemple semnificative, fară a putea acoperi întregul spectru de posibilităţi ale bibliotecii IO. De alfel şi în lucrările de laborator precedente s-au utilizat operaţii IO şi chiar supradefinirea operatorilor de inserţie şi de extracţie.

Stadarul C++ conţine o serie de biblioteci standard moşenite din C şi o serie de biblioteci specifice. Concetul de *stream* (curent) nu a apărut odată cu C++. Implementarea stream-urilor în C++ a fost făcută încă de la începutul existenţei limbajului C++, ştiut fiind faptul că bibliotecile de intrare-ieşire reprezintă un element sensibil pentru orice limbaj. În figura IX.1 este reprezentată structura de clase utilizată pentru implementarea stream-urilor de intrare-ieşire.

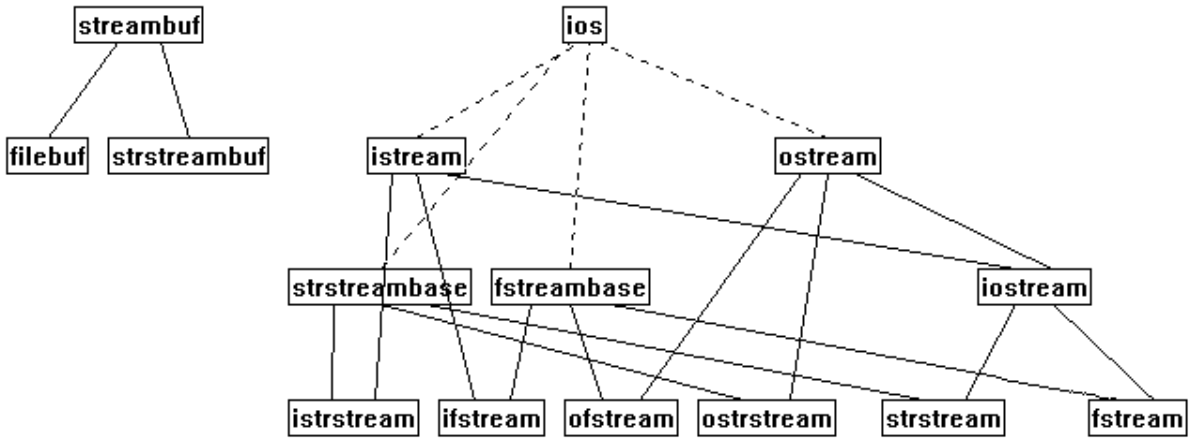

Figura IX.1 Structura de clase care formează biblioteca IO pentru C++

Clasa streambuf

Clasa virtuală (abstractă) streambuf este responsabilă de gestiunea buffer-ului utilizat pentru crearea unei implementări eficiente a streamurilor. Prin intermediul metodelor pur virtuale clasa streambuf furnizează cadrul de implementare a comunicării cu dispozitivele fizice asociate streamurilor (fişiere pe diverse suporturi, consolă: monitor – tastatură).

Stream-urile gestionează mereu un pointer la obiectul asociat de un tip derivat din streambuf.

Clasa filebuf

Clasa filebuf este derivată din clasa streambuf şi adaugă funcţionalităţile necesare pentru comunicarea cu fişiere.

Clasa strstreambuf

Clasa strstreambuf este derivată din clasa streambuf şi adaugă funcţionalităţile necesare pentru citirea şi scrierea caracterelor din, respectiv într-un buffer de tip şir de caractere. Citirea şi scrierea se pot face aleatoriu în interiorul buffer-ului. Operaţii de căutare sunt de asemenea implementate. Zona rezervată de un obiect strstreambuf poate fi de dimensiune fixă sau dinamică. În general stream-urile de intrare sunt de dimensiune fixă, iar cele de ieşire, putând avea dimensiune imprevizibilă, pot fi şi dinamice.

Diferenţa faţă de fstreambuf este că în cazul unui obiect strstreambuf nu există o sursă sau destinaţie reală pentu operaţiile de scriere, respectiv citire, astfel încât chiar buffer-ul preia unul sau ambele roluri.

Clasa ios

Clasa ios este utilizată pentru a grupa o serie de funcţionalităţi comune, necesare pentru clasele derivate. Nu s-a intenţionat crearea de obiecte de tipul ios.

Clasa gestionează informaţii despre stream (ios – I/O State): flag-uri de eroare, flag-uri şi valori de formatare, precum şi conexiunea cu buffer-ele utilizate la intrare/ieşire (pointerul către structura de informaţii a buffer-ului).

Clasa istream

Clasa istream furnizează funcţionalităţi pentru citirea la nivel de caracter şi cu format. În general un obiect istream nu este construit explicit într-un program deoarece la acest nivel nu există un mecanism de deschidere a unui dispozitiv. Singurul obiect implicit de tipul istream într-un program este **cin**, care este asociat cu dispozitivul standard de intrare (de obicei tastatura).

Clasa istream suportă două moduri de intrare: intrare formatată (conversie de la şiruri de caractere la formatul intern de reprezentare) şi intrare neformatată (formatul intern de reprezentare).

La acest nivel este supradefint operatorul de extracţie (>>) pentru tipurile predefinite. Operatorul de extracţie suportă operaţiile de intrare formatată. Celelalte metode se ocupă cu operaţii de intrare neformatată, gestionând starea membrilor moşteniţi de la ios.

Clasa ostream

Clasa ostream furnizează funcţionalităţi pentru scrierea la nivel de caracter şi cu format. În general un obiect de tipul ostream nu este construit explicit într-un program deoarece nu există la acest nivel un mecanism de deschidere al unui dispozitiv. Singurele obiecte implicite de tipul ostream sunt: **cout**, **cerr** şi **clog** care scriu la dispozitivul standard de ieşire şi de eroare (de obicei monitorul).

Clasa ostream suportă două moduri de ieşire: ieşire formatată şi ieşire neformatată.

La acest nivel operatorul de inserţie (<<) este supradefinit pentru tipurile de date predefinite. Operatorul de extracţie suportă operaţiile de intrare formatată. Celelalte metode se ocupă cu operaţii de intrare neformatată, gestionând starea membrilor moşteniţi de la ios.

Clasa iostream

Clasa iostream furnizează funcţionalităţi pentru scrierea şi citirea la nivel de caracter şi cu format la, şi respectiv de la dispozitivele standard de ieşire, respectiv intrare. În general un obiect de tipul iostream nu este construit explicit într-un program deoarece nu există la acest nivel un mecanism de deschidere al unui dispozitiv.

Nu există nici un obiect implicit de tipul iostream deoarece nu există dispozitive IO standard care să permită atât operaţii de ieşire cât şi operaţii de intrare. Clasa iostream este clasă de bază pentru alte clase derivate care au posibilitatea de a realiza operaţii IO, de intrare şi ieşire în acelaşi timp, utilizând acelaşi stream. Calsele fstream şi strstream sunt exemple de astfel de clase.

Clasa fstreambase

Clasa fstreambase este o clasă de bază care furnizează funcţionalităţi comune pentru cele trei clase pentru operaţii IO cu fişiere: ifstream, ofstream şi fstream. Clasa fstream base adaugă moştenirii de la ios funcţiile de deschidere (conectare la) şi închidere (deconectare de la) a fişierelor.

Nu s-a intenţionat crearea de obiecte de tipul fstreambase. Utilizatorul va crea obiecte de tipul ifstream, ofstream şi/sau fstream.

Clasa strstreambase

Clasa strstreambase este o clasă de bază care furnizează funcţionalităţi comune pentru cele trei clase pentru operaţii IO stream-uri de şiruri de caractere: istrstream, ostrstream şi strstream. Clasa fstream base adaugă moştenirii de la ios constructorii şi destructorul pentru streamuri de şiruri de caractere.

Clasa ifstream

Clasa ifstream este utilizată pentru accesarea fişierelor existente pentru operaţii de citire. Aceste fişiere pot fi deschise şi închise, şi se pot executa operaţii de citire şi căutare.

Crearea unui obiect de tipul ifstream este metoda preferată pentru accesarea unui fişier pentru operaţii exclusiv de intrare.

Clasa ofstram

Clasa ofstream este utilizată pentru crearea fişierelor noi sau accesare celor deja existente pentru operaţii de scriere. Fişierul poate fi deschis şi închis, şi se pot executa operaţii de scriere şi căutare.

Crearea unui obiect de tipul ofstream este metoda preferată pentru accesarea unui fişier pentru operaţii exclusiv de ieşire.

Clasa fstream

Clasa fstream este utilizată pentru a accesa fişiere pentru operaţii de citire, respectiv scriere. Un fişier poate fi deschis şi închis, şi se pot executa operaţii de citire, scriere şi căutare.

Crearea unui obiect de tipul fstream este metoda preferată pentru accesarea unui fişier pentru operaţii de intrare şi ieşire.

Clasa istrstream

Clasa istrstream este utilizată pentru crearea şi citirea din stream-uri şiruri de caractere.

Crearea unui obiect de tipul istrstream este metoda preferată pentru operaţii de citire dintr-un stream şir de caractere.

Clasa ostrstream

Clasa ostrstream este utilizată pentru crearea şi scrierea în stream-uri şiruri de caractere.

Crearea unui obiect de tipul ostrstream este metoda preferată pentru operaţii de scriere într-un stream şir de caractere.

Clasa strstream

Clasa strstream este utilizată pentru crearea şi citirea şi scrierea din, respectiv în stream-uri şiruri de caractere.

Crearea unui obiect de tipul strstream este metoda preferată pentru operaţii de citire şi scriere într-un stream şir de caractere.

IX.3 OBIECTE IO PREDEFINITE

Cele mai multe programe interacţionează într-un fel sau altul cu tastatura şi monitorul. Limbajul C furnizează trei *valori*: stdin, stdout şi stderr care sunt utilizate pentru comunicarea cu aceste *dispozitive standard* care sunt *deschise* înainte ca programul să înceapă să execute funcţia main. Aceste trei valori au tipul FILE * şi pot fi utilizate pentru orice operaţii cu fişiere.

Similar C++ pune la dispoziţie şapte obiecte pentru comunicarea cu aceleaşi dispozitive standard: cele trei obiecte C de tipul FILE *: stdin, stdout şi stderr, şi patru noi obiecte: cin, cout, cerr şi clog care corespund funcţional respectiv lui stdin, stdout, stderr şi stderr cu buffer.

cin

cin este un obiect de tipul istream care este conectat la *intrarea standard* înainte de începerea execuţiei funcţiei main. Operaţiile de intrarea pe cin se fac formatat.

cout

cout este un obiect de tipul ostream care este conectat la *ieşirea standard* înainte de începerea execuţiei funcţiei main. Operaţiile de ieşire pe cout se fac formatat, şi prin intermediul unui buffer.

cerr

cerr este un obiect de tipul ostream care este conectat la *ieşirea standard* înainte de începerea execuţiei funcţiei main. Operaţiile de ieşire pe cerr se fac formatat, şi fără buffer.

clog

cout este un obiect de tipul ostream care este conectat la *ieşirea standard* înainte de începerea execuţiei funcţiei main. Operaţiile de ieşire pe clog se fac formatat, şi prin intermediul unui buffer.

IX.4 PROGRAMUL I

Programele ilustrative din acest laborator nu-şi propun o trecere exhaustivă în revistă a funcţionalităţilor bibliotecilor IO ale C++. Ele sunt mai degrabă simple şi vă invită să consultaţi documentaţia în momentul în care simţiţi nevoia.

Programul care urmează ilustrează scrierea şi respectiv citirea în, şi din fişiere cu format şi binare.

Urmăriţi pas cu pas rularea programului pentru un număr de valori scrise/citite mic: 2-3. Comparaţi modul de rulare al programului cu rezultatele anticipate de dumneavoastră acasă, pe hârtie. Observaţi atitudinea defensivă în faţa posibilelor erori. Identificaţi mijloacele utilizate pentru depistarea şi rezolvarea erorilor.

Rulaţi programul pentru un număr mare de valori scrise/citite (>= 1000). Observaţi fişierele create de program. Observaţi dimensiunile şi conţinutul acestora. Încercaţi rularea pentru numere mari de valori scrise/citite şi corelaţi dimensiunile fişierelor cu tipul acestora şi numărul de elemente scrise. Relizaţi acelaşi experiment, pentru un număr mare şi fixat de valori scrise/citite şi modificaţi numărul de cifre semnificative pentru mantisă. Reprezentaţi grafic rezultatele.

ioformatbinar.cpp

```
#include <iostream.h>
#include <iomanip.h>
#include <fstream.h>
#include <math.h>
#include <string.h>

enum Error
{
   Success,
   Error
};

//Constanta pi cu valoarea sugerata de nume.
const double pi(3.14159265358979323846);

int main(void)
{
   //Siruri de caractere utilizat in mesajele de eroare
   //la manipularea streamurilor conectate la fisiere.
   char      *OpenError("Eroare la deschiderea fisierului ");
   char      *WriteError("Eroare la scrierea in fisierul ");
   char      *ReadError("Eroare la citirea din fisierul ");
   char      *KbdError("Eroare la citirea de la tastatura !!!");
   //Stream in care se va scrie formatat ...
   fstream   FileFormat;
   //Stream in care se va scrie binar ...
   fstream   FileBinar;
   //Sir de caractrere in care se citeste numele
```

```
//fisierului in care se va scrie formatat ...
char        FileNameFormat[256];
//Sir de caractrere in care se citeste numele
//fisierului in care se va scrie binar ...
char        FileNameBinar[256];
//Numarul de valori scrise / citite.
int         N;
//"Precizia" cu care se face iesirea cu format.
int         Prec;

cout << "Numele fisierului cu format : ";
cin >> FileNameFormat;
//Nu este deloc rau sa verificam succesul operatiei.
if(cin.fail())
{
   cerr << KbdError << endl;
   return Error;
}
cout << "Numarul de cifre semnificative pentru mantisa : ";
cin >> Prec;
if(cin.fail())
{
   cerr << KbdError << endl;
   return Error;
}
cout << "Numele fisierului binar : ";
cin >> FileNameBinar;
if(cin.fail())
{
   cerr << KbdError << endl;
   return Error;
}
//Ne asiguram ca numele fisierelor nu coincid ...
if(!strcmp(FileNameFormat, FileNameBinar))
{
   cerr << "Numele fisierelor coincid ! " << FileNameFormat << " == " << FileNameBinar << endl;
   return Error;
}

//Deschiderea fisierului cu format.
//Implicit streamurile sunt formatate.
//S-a setat formatul cu mantisa si exponent (scientific)
//Pentru valorile in virgula mobila.
FileFormat.open(FileNameFormat, ios::out | ios::scientific);
if(!FileFormat)
{
   cerr << OpenError << FileNameFormat << endl;
   return Error;
}
//Se seteaza "precizia" pentru iesirea cu format
//la valoarea 10. Pentru formatul "scientific" reprezinta
//numarul de cifre semnificative reprezentate pentru mantisa.
//S-a utilizat functia membru precision(int)
```

```
FileFormat.precision(Prec);

//Deschiderea fisierului binar.
FileBinar.open(FileNameBinar, ios::out | ios::binary);
if(!FileBinar)
{
   cerr << OpenError << FileNameFormat << endl;
   //Urmatoarea instructiune poate lipsi,
   //in acest context fiind oricum apelat
   //destructorul obiectului FileFormat
   FileFormat.close();
   return Error;
}

cout << "Numarul de valori scrise / citite : ";
cin >> N;
if(cin.fail())
{
   cerr << KbdError << endl;
   return Error;
}

double Val;
for(int i = 0; i < N; i++)
{
   Val = tan(pi / (i + 1));
   //Testarea erorii se face cu utilizarea referintei
   //returnate de operatorul de insertie.
   if((FileFormat << Val << ' ').fail())
   {
      cerr << WriteError << FileNameFormat << endl;
      return Error;
   }
   //Conversia pointerului la Val(double)
   //in pointer la sir de caractere cu lungimea
   //egala cu numarul de octeti ocupati de un
   //obiect de tip double. Metoda write copie la
   //iesire, octet cu octet formatul intern de
   //reprezentare al lui Val
   FileBinar.write((char *)&Val, sizeof(Val));
   //O alta varianta de testare a erorii
   //fara utilizarea referintei returnate de write.
   if(FileBinar.fail())
   {
      cerr << WriteError << FileNameBinar << endl;
      return Error;
   }
}
//Aici este necesara inchiderea fisierelor.
FileFormat.close();
FileBinar.close();

//Se folosesc aceleasi obiecte pentru a testa citirea
```

```
//cu format si binara. Operatiile care se fac reprezinta
//exact operatiile inverse de la scriere.
FileFormat.open(FileNameFormat, ios::in);
//Alta varianta de testare a starii obiectului
if(!FileFormat)
{
   cerr << OpenError << FileNameFormat << endl;
   return Error;
}

FileBinar.open(FileNameBinar, ios::binary | ios::in);
if(FileBinar.fail())
{
   cerr << OpenError << FileNameBinar << endl;
   return Error;
}

double     ValBinar, ValFormat;
//Se utilizeaza i pentru a contoriza numarul
//de operatii de citire realizate.
i = -1; //De ce s-a atribuit valoarea -1 lui i ?
//Se evalueaza si eroare maxima absoluta.
double     MaxError(0);
double     MaxErrorTemp;
double     MaxErrorFormat;
double     MaxErrorBinar;
//Alta varianta de testare a validitatii
//operatiilor SI pentru un stream SI pentru
//celalalt
while(FileBinar && FileFormat)
{
   FileFormat >> ValFormat;
   FileBinar.read((char *)&ValBinar, sizeof(ValBinar));
   i++;
   MaxErrorTemp = fabs(ValFormat - ValBinar);
   if(MaxError < MaxErrorTemp)
   {
      MaxError = MaxErrorTemp;
      MaxErrorFormat = ValFormat;
      MaxErrorBinar = ValBinar;
   }
}
if(i != N)
{
   cerr << "S-au citit corect " << i << " valori din " << N << '!' << endl;
   return Error;
}
cout << "Eroarea maxima obtinuta la citire este : " << MaxError << ',' << endl;
cout << "si s-a obtinut pentru valorile : " << endl << "Format\t\t\tBinar" << endl;
//O alta forma de setare a numarului de cifre semnificative ...
//S-a utilizat manipulatorul setprecision(int)
cout << setprecision(Prec+1) << MaxErrorFormat << '\t' << MaxErrorBinar << endl;
//Este elegant ca si aici fisierele sa fie inchise, desi va urma imediat
```

```
   //apelul la destructorul clasei fstream pentru cele doua obiecte.
   FileFormat.close();
   FileBinar.close();

   cin.get();
   return Success;
}
```

IX.5 PROGRAMUL II

Programul care urmează realizează copierea unui fişier ASCII. Construiţi în directorul curent un fişier text care va avea rolul de fişier sursa în timpul operaţiilor de copiere. Rulaţi programul pas cu pas, simulaţi erori de utilizare.

asciicopy.cpp

```
#include <iostream.h>
#include <fstream.h>

enum Err
{
   Success,
   Error
};

//Programul are parametri in linia de comanda
int main(int argc, char *argv[])
{
   //Sunt sau nu furnizate corect argumentele ?!?
   switch(argc)
   {
      case 3:
            cerr << argv[0] << " realizeaza copia ASCII : " << argv[1] << " -> " << argv[2] << ' ';
            break;
      default :
          cerr << "Utilizare : " << argv[0] << " sursa destinatie" << endl;
            return Error;
   }

   char       *OpenError("Eroare la deschiderea fisierului ");
   char       *WriteError("Eroare la scrierea in fisierul ");
   char       *ReadError("Eroare la citirea din fisierul ");
   ifstream    In(argv[1]);
   if(!In)
   {
      cerr << OpenError << argv[1] << endl;
      return Error;
   }
   ofstream    Out(argv[2]);
```

```
    if(!Out)
    {
       cerr << OpenError << argv[2] << endl;
       return Error;
    }

    char      CTemp;
    //Cat timp nu s-a ajuns la sfarsitul fisierului de intrare...
    while(!In.eof())
    {
       CTemp = In.get();
     //...si nu a aparut o eroare la intrare
       if(In.good())
          Out.put(CTemp);
       else if(!In.eof())
       {
          cerr << ReadError << argv[1] << endl;
          return Error;
       }
     //...si nu a aparut o eroare la iesire ...
       if(Out.fail())
       {
          cerr << WriteError << argv[2] << endl;
          return Error;
       }
    }
    //Foarte sanatos ...
    In.close();
    Out.close();

    cerr << "O.K." << endl;
    return Success;
}
```

IX.6 TEMĂ

Analizaţi în scris comportarea programelor ilustrative. Tot în scris răspundeţi la întrebările:

1. De ce nu se pot utiliza obiecte de tipul streambuf?

2. De ce apare diferenţe la citirea valorilor în virgulă flotantă din fişiere cu format şi binare? Urmăriţi programul corespunzător.

3. De ce programul de copiere pentru fişiere ASCII nu este utilizabil în cazul fişierelor binare?

Concluzionaţi rezultatele experimentelor realizate la primul program ilustrativ şi formulaţi concluziile în scris. Furnizaţi şi o explicaţie.

Construiţi un program (foarte simplu) de copiere pentru fişiere binare.

Modificaţi clasa MatrixInt2 din lucrarea de laborator cu numărul IV pentru a realiza operaţii IO cu format şi binar cu fişiere. Construiţi un mic program de test care să ilustreze modul în care aţi rezolvat problema.