========== Strfile.cpp =========
```
#include <fstream.h>
#include <iostream.h>
#include <assert.h>

int main() {
  const int sz = 100;
  char buf[sz];
  {
    ifstream in("Strfile.cpp");
    assert(in);
    ofstream out("Strfile.out");
    assert(out);
    int i = 1;

    while(in.get(buf, sz)) { // lasa \n la intrare
      in.get(); // extrage urmatoarele caractere (\n)
      cout << buf << endl;
      out << i++ << ": " << buf << endl;
    }
  }

  ifstream in("Strfile.out");
  assert(in);

  while(in.getline(buf, sz)) { // elimina \n
    char* cp = buf;
    while(*cp != ':')
      cp++;
    cp += 2;
    cout << cp << endl;
  }
}
```

========== seek.cpp ============
```
#include <iostream.h>
#include <fstream.h>
#include <assert.h>

int main(int argc, char* argv[]) {
  ifstream in(argv[1]);

  assert(in);
  in.seekg(0, ios::end); // sfirsit fisier
  streampos sp = in.tellg(); // dimensiune fisier
  cout << "dimensiune fisier = " << sp << endl;

  in.seekg(-sp/10, ios::end);
  streampos sp2 = in.tellg();
  in.seekg(0, ios::beg);
  cout << in.rdbuf();

  in.seekg(sp2);
  cout << endl << endl << in.rdbuf() << endl;
}
```

============= Datalog.h ===========
```
#ifndef DATALOG_H
#define DATALOG_H
#include <time.h>
```

```
#include <iostream.h>

const int bsz=10;

class DataPoint {
            tm Tm; // timp & data
            char Latitude[bsz], Longitude[bsz];
            double Depth, Temperature;
      public:
            tm Time(); // citeste timpul
            void Time(tm t); // seteaza timpul
            const char* latitude();
            void latitude(const char* l);
            const char* longitude();
            void longitude(const char* l);
            double depth();
            void depth(double d);
            double temperature();
            void temperature(double t);
            void print(ostream& os);
};
#endif
```

========== Datalog.cpp ==========

```
#include "datalog.h"
#include <iomanip.h>
#include <string.h>

tm DataPoint::Time(){
  return Tm;
}

void DataPoint::Time(tm t){
  Tm = t;
}

const char* DataPoint::latitude() {
  return Latitude;
}

void DataPoint::latitude(const char* l) {
  Latitude[bsz - 1] = 0;
  strncpy(Latitude, l, bsz - 1);
}

const char* DataPoint::longitude() {
  return Longitude;
}

void DataPoint::longitude(const char* l) {
  Longitude[bsz - 1] = 0;
  strncpy(Longitude, l, bsz - 1);
}

double DataPoint::depth(){
  return Depth;
}

void DataPoint::depth(double d){
  Depth = d;
}
```

```
double DataPoint::temperature() {
  return Temperature;
}

void DataPoint::temperature(double t) {
  Temperature = t;
}

void DataPoint::print(ostream& os) {
  os.setf(ios::fixed, ios::floatfield);
  os.precision(4);
  os.fill('0'); // umple la stinga cu '0'
  os << setw(2) << Time().tm_mon << '\\'
     << setw(2) << Time().tm_mday << '\\'
     << setw(2) << Time().tm_year << ' '
     << setw(2) << Time().tm_hour << ':'
     << setw(2) << Time().tm_min << ':'
     << setw(2) << Time().tm_sec;
  os.fill(' '); // umple la stinga cu ' '
  os << " Lat:" << setw(9) << latitude()
     << ", Long:" << setw(9) << longitude()
     << ", depth:" << setw(9) << depth()
     << ", temp:" << setw(9) << temperature()
     << endl;
}
```

========== Datagen.cpp ==========

```
#include <fstream.h>
#include <stdlib.h>
#include <string.h>
#include "datalog.h"

int main() {
  ofstream data("data.txt");
  ofstream bindata("data.bin", ios::binary);
  time_t timer;

  // setarea generatorului de numere aleatoare
  srand(time(&timer));
  for(int i = 0; i < 100; i++) {
    DataPoint d;
    // converteste data/timpul la o structura
    d.Time(*localtime(&timer));
    timer += 55; // citeste fiecare 55 secunde
    d.latitude("45*20'31\"");
    d.longitude("22*34'18\"");

    double newdepth  = rand() % 200;
    double fraction = rand() % 100 + 1;
    newdepth += double(1) / fraction;
    d.depth(newdepth);

    double newtemp = 150 + rand()%200; // Kelvin
    fraction = rand() % 100 + 1;
    newtemp += (double)1 / fraction;

    d.temperature(newtemp);
    d.print(data);
    bindata.write((unsigned char*)&d,sizeof(d));
  }
```

```
}
```

========== Datascan.cpp ==========

```
#include <iostream.h>
#include <fstream.h>
#include <strstream.h>
#include <iomanip.h>
#include "datalog.h"

int main() {
  ifstream bindata("data.bin", ios::binary);
  ofstream verify("data2.txt");
  DataPoint d;

  while (bindata.read((unsigned char*)&d, sizeof d))
    d.print(verify);
  bindata.clear(); // resetarea starii
  // numarul inregistrarii care se va afisa
  int recnum = 0;
  // aliniere la stinga
  cout.setf(ios::left, ios::adjustfield);
  // afisare cu precizie pe 4 pozitii
  cout.setf(ios::fixed, ios::floatfield);
  cout.precision(4);
  for(;;) {
    bindata.seekg(recnum * sizeof d, ios::beg);
    cout << "Inregistrarea " << recnum << endl;
    if (bindata.read((unsigned char*)&d, sizeof d)) {
      cout << asctime(&(d.Time()));

      cout << setw(12) << "Latitudine"
           << setw(12) << "Longitudine"
           << setw(10) << "Adincime"
           << setw(12) << "Temperatura"
           << endl;

      cout << setfill('-') << setw(43) << '-'
           << setfill(' ') << endl;

      cout << setw(12) << d.latitude()
           << setw(12) << d.longitude()
           << setw(10) << d.depth()
           << setw(12) << d.temperature()
           << endl;
    } else {
      cout << "Numar de inregistrare incorect" << endl;
      bindata.clear(); // reset
    }

    cout << endl
         << "Introduceti numarul inregistrarii, x pentru terminare ";

    char buf[10];
    cin.getline(buf, 10);
    if (buf[0] == 'x')
      break;
    istrstream input(buf, 10);
    input >> recnum;
  }
}
```

Temă

1. Să se realizeze o aplicație ce păstrează evidența produselor într-un depozit (nume depozit, adresă, listă produse).
Pentru fiecare produs se consideră următoarele informații:

- nume produs;
- cod produs (unic pentru fiecare produs din depozit);
- număr bucăți din acelaşi produs;
- preț unitar;

Operații:

- actualizare produse:
- intrarea în depozit a unui produs
- ieşirea din depozit a unui produs
- sortare după cantitate produs
- sortare după nume produs
- sortare produse după cod
- determinarea cantității unui anumit produs

- eliminare produse cu cantitatea zero
- determinarea produsului cu cantitate maximă
- afişare produse sortate după cantitate
- afişare produse sortate după cod
- afişare produse sortate după nume
- afişarea produselor cu o cantitate mai mare decât una specificată

Aplicația îşi va gestiona toate informațiile cu ajutorul unui fişier (supraâncărcarea operatorilor << şi >> pentru clasa Depozit).

```
#ifndef _PRODUS_
#define _PRODUS_

#include <iostream.h>
#include <string.h>

class Produs{
            char* denumire;
            char* cod;
            int  nrBucati;
            long pret;
      public:
            Produs(char* , char*, int, long);
            ~Produs();

            char* getCod();
            int modificNrBucati(int );
            int getNrBucati();
            long getPret();

            …
};

#endif
```

2. Să se implementeze o ierarhie de clase compusă din clasele *Publicatie*, *Carte*, *Revista*.
Informațiile aferente publicațiilor unei biblioteci sunt:
- titlu publicație (şir de 20 caractere)
- nume editor/editură (şir de 20 caractere)
- anul publicării
- cota publicației

Clasa *Carte* va fi derivata din clasa *Publicație*, şi va avea în plus:
- nume autor(i)
- cod ISBN
- greutate

Clasa *Revista* va fi derivata din clasa *Publicatie*.
Să se pună în evidență metode specifice pentru accesul la membrii acestor clase, constructori, destructori, moştenirea metodelor, redefinirea unei metode moştenite.

3. Să se scrie un program care să gestioneze publicațiile dintr-o bibliotecă, utilizându-se ierarhia de clase implementată în cadrul problemei anterioare.
Operații:
- actualizare publicație (modificare numelui clientului care a împrumutat-o şi/sau număr cod)
- adăugare publicație nouă
- ştergere publicație din bibliotecă
- afişarea tuturor cărților
- afişarea cărților împrumutate de un client
- afişarea cărților aparținând unui anumit domeniu sortate după nume autor.

Bibliografie

http://inf.ucv.ro/~mirel/courses/poo/poo.html

1. *The C++ Programming Language*, B. Stroustrup, Addison-Wesley.
(Cap. 21 Streams)

2. *C++ Coding Standard*, Todd Hoff
http://www.possibility.com/Cpp/CppCodingStandard.html