PCLP2

Laboratorul 5

1. **Compunerea claselor.** Există nenumărate exemple de obiecte din viața de zi cu zi care sunt alcătuite din alte obiecte mai mici. Spre exemplu, o maşină este alcătuită din caroserie, motor, transmisie, roți şi alte elemente mai mici. Acest concept este transpus în programarea orientată pe obiecte prin compunerea claselor. Dezvoltăm clase noi integrându-le obiecte din alte clase. Studiați exemplul clasei Employee care cuprinde două instanțe ale clasei Date. Pentru aceasta dezvoltați un proiect Dev-Cpp care cuprinde cele 5 fişiere de mai jos.

date1.h

```
#ifndef DATE1_H
#define DATE1_H
class Date
{
  public:
    //constructor implicit
    Date(int = 1, int = 1, int = 1990);
    void print() const;
    ~Date();
  private:
    int day;  //1-12
    int month;//1-31
    int year;
    //functie utilitara de testare a
    //corectitudinii zilei pentru month si year
    int checkDay(int);
};
#endif
```

date1.cpp

```
#include <iostream>
using std::cout;
using std::endl;
#include "date1.h"
//constructor fara verificarea valorilor
Date::Date(int d, int m, int y)
{
  if(m > 0 && m <= 12)
    month = m;
  else {
    month = 1;
    cout << "Luna " << m << " incorecta. "
         << "Valoarea implicita este 1.\n";
  }
  year = y;
  day = checkDay(d); //valideaza ziua
  cout << "Constructorul obiectului de tip Date pentru ";
  print();
  cout << endl;
}
```

```
//Tipareste data in forma zi-luna-an
void Date::print() const
  { cout << day << '-' << month << '-' << year; }

//Destructorul   folosit   pentru   confirmarea   stergerii
obiectului
Date::~Date()
{
  cout << "Destructorul obiectului de tip Date pentru ";
  print();
  cout << endl;
}

int Date::checkDay( int testDay )
{
  static const int daysPerMonth[13] =
    {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
  if( testDay > 0 && testDay <= daysPerMonth[month])
    return testDay;

  //Februarie: test pentru an bisect
  if(month == 2 && testDay == 29 &&
      (year % 400 == 0 ||
      (year % 4 == 0 && year % 100 != 0)))
    return testDay;

  cout << "Ziua " << testDay << " incorecta. "
       << "Valoarea implicita este 1.\n";
  return 1;
}
```

employee1.h

```
#ifndef EMPLOYEE1_H
#define EMPLOYEE1_H
#include"date1.h"
class Employee
{
  public:
    Employee(char*, char*, int, int, int, int, int, int);
    void print() const;
    ~Employee();//destructor folosit la confirmarea ordinii
                //in care sunt sterse obiectele
  private:
    char firstName[25];
    char lastName[25];
    const Date birthDate;
    const Date hireDate;
};
#endif
```

employee1.cpp

```
#include <iostream>
using std::cout;
```

```
using std::endl;

#include <cstring>
#include "employee1.h"
#include "date1.h"

Employee::Employee(char* fname, char*lname,
           int bday, int bmonth, int byear,
           int hday, int hmonth, int hyear)
  : birthDate(bday, bmonth, byear),
    hireDate(hday, hmonth, hyear){
  //copiaza fname in firstName
  //verificand daca lungimea corespunde
  int length = strlen(fname);
  length = (length < 25 ? length : 24);
  strncpy( firstName, fname, length);
  firstName[length] = '\0';

  //copiaza lname in lastName
  //verificand daca lungimea corespunde
  length = strlen(lname);
  length = (length < 25 ? length : 24);
  strncpy( lastName, lname, length);
  lastName[length] = '\0';

  cout << "Constructorul obiectului Employee: "
       << firstName << ' ' << lastName << endl;
}

void Employee::print() const
{
  cout << lastName << ", " << firstName << "\nAngajat: ";
  hireDate.print();
  cout << "  Data nasterii: ";
  birthDate.print();
  cout << endl;
}

//Destructorul folosit pentru
//confirmarea stergerii obiectului
Employee::~Employee()
{
  cout << "Destructorul obiectului de tip Employee: "
       << lastName << ", " << firstName << endl;
}
```

test_composition.cpp

```
#include <iostream>
using std::cout;
using std::endl;

#include "employee1.h"
```

```
int main()
{
  Employee e("Bob", "Jones", 24, 7, 1949, 12, 3, 1988);

  cout << '\n';
  e.print();

  cout << "\nTesteaza constructorul Date "
       << "pentru valori incorecte:\n";
  Date d(35, 14, 1994);
  cout << endl;

  return 0;
}
```

2. **Clasele Punct şi Dreapta**. Urmând modelul de mai sus, implementați clasele Punct şi dreaptă.

a. **Clasa Punct** conține:
   i. Două date membre reale x şi y care reprezintă coordonatele unui punct;
   ii. Funcții getter si setter pentru datele membre;
   iii. Un constructor implicit care inițializează datele membre cu 0;
   iv. Un constructor explicit.

b. **Clasa Dreapta** conține:
   i. Două date membre p1 şi p2 de tip Punct;
   ii. Funcții getter si setter pentru cele două date membre;
   iii. Un constructor explicit cu 4 parametri reali, câte doi pentru fiecare punct;
   iv. O funcție membră de afişare a coordonatelor celor două puncte într-un format ales de voi

c. Funcția **main** conține:
   i. Declarația unui obiect de tip Dreapta prin care transmiteți coordonatele celor două puncte care determină dreapta;
   ii. Afişarea obiectului;
   iii. Modificați poziția celor două puncte apelând funcțiile setter pentru p1 şi p2;
   iv. Afişarea din nou a obiectului.