## Lab work #7

**Topics**

* Inheritance

**Exercises**

1. Using inheritance, rewrite the program developed in session 6 for geometric figures. Besides the classes Point, Circle, Square and Rectangle, it is intended to create a new class named Figure that should contain all the shared data of the previous classes. Moreover, Square and Rectangle can also be improved using inheritance.

Overload the operator << in order to be possible to print the information regarding the objects of the referred figures. At this point consider that all the information should be shown. Overload also the operator >> in order to ask the user information about a specific figure.

Test the implemented classes with the following program:

**int** main() {

*// Testing class Point*

Point p1(2,2), p2;

cout << **"Insert Point: x y"** << endl;

cin >> p2;

cout << **"P1: "** << p1 << endl << **"P2: "** << p2 << endl;

cout << **"Distance P1<->P2: "** << p1.dist(p2)<< endl;

*// Testing class Figure*

Figure f1(p1, **"green"**), f2;

cout << **"Insert Figure Info: Centre(x y) Color"** << endl;

cin >> f2;

cout << **"F1: "** << f1 << endl << **"F2: "** << f2 << endl;

*// Testing class Circle*

Circle c1(p1, **"blue"**, 4), c2;

cout << **"Insert Circle Info: Centre(x y) Color Radius"** << endl;

cin >> c2;

cout << **"C1: "** << c1 << endl << **"C2: "** << c2 << endl;

cout << **"Area: AC1 = "** << c1.area() << **"; AC2 = "** << c2.area() << endl;

cout << **"Perimeter: PC1 = "** << c1.perimeter() << **"; PC2 = "** << c2.perimeter() << endl;

cout << **"C1 intersects C2: "** << c1.intersect(c2) << endl;

*// Testing class Rectangle*

*// ... add here code lines for testing class Rectangle*

*// Testing class Square*

*// ... add here code lines for testing class Square*

**return** 0;

}

1. Implement a class Person who is characterized by name, citizen's card number and date of birth. Start with the following settings:

| class Person {  private:  String name;  int cc;  Date dateBirth;  // .....  public:  // .....  }; | class Date{  private:  int day;  int month;  int year;  // .....  public:  // .....  }; |
| --- | --- |

Make use of access modifiers to ensure that all class attributes are not accessible from the outside. If necessary, define new methods to include in the class's public interface. Create default and copy constructors and at least another one to allow the initialization of the class attributes when each object is created in the following form:

Date d(5, 10, 1988);

Person p("Ana Santos", 98012244, d);

Implement a new class Student, derived from the class Person, adding the necessary methods and attributes to access and save the id number (int) and the date of enrollment (Date) in the educational institution. Note that the id number must be assigned automatically (and sequentially from 10000) when creating a new student (explore the use of static attributes).

class Student : public Person{

private:

unsigned int id;

Date dateEnrollment;

// .....

public:

// .....}

Create default and copy constructors and at least two more to allow the initialization of the Student attributes when created in the following form:

Student s1("Maria", 9855678, Date(18, 7, 1990), Date(1, 9, 2014));

Student s2("Maria", 9855678, Date(18, 7, 1990));

Student s2 should consider the current date as the date of enrollment. There are several ways to obtain the current date, but we suggest the use of the module ctime (<https://www.tutorialspoint.com/cplusplus/cpp_date_time.htm>)

#include <iostream>

#include <ctime>

using namespace std;

int main() {

// current date/time based on current system

time\_t now = time(0);

tm \*ltm = localtime(&now);

// print various components of tm structure.

cout << "Year:" << 1900 + ltm->tm\_year<<endl;

cout << "Month: "<< 1 + ltm->tm\_mon<< endl;

cout << "Day: "<< ltm->tm\_mday << endl;

}

Create the Fellow class, derived from the Student class, which should include an attribute with the amount of the scholarship (double). Define new methods or rewrite methods that seem appropriate.

class Fellow: public Student{

private:

double scholarship;

// .....

public:

// .....

}

Create default and copy constructors and at least two more to allow the initialization of the Fellow attributes when created in the following form:

Fellow f1("Maria", 9855678, Date(18, 7, 1990), Date(1, 9, 2014), 1500.5);

Fellow f2("Maria", 9855678, Date(18, 7, 1990));

Fellow f2 should consider the current date as the date of enrollment and 1000.0 as the amount of the scholarship.

Implement all the members functions to set and get values of the attributes for all the classes and overload the operator << in order to be possible to print the information regarding the objects of the referred classes. At this point consider that all the information should be shown. Overload also the operator >> in order to ask the user information about a date, a student or a fellow. Test the implemented classes with the following program:

int main(){

vector <Student> students;

students.push\_back(Student("Maria", 9855678, Date(18, 7, 1990), Date(1, 9, 2014)));

students.push\_back(Student("Maria", 9855678, Date(18, 7, 1990)));

vector <Fellow> fellows;

fellows.push\_back(Fellow("Maria", 9855678, Date(18, 7, 1990), Date(1, 9, 2014), 1500.5));

fellows.push\_back(Fellow("Maria", 9855678, Date(18, 7, 1990)));

Student s;

cout << "Info for a new student:" << endl;

cin >> s;

students.push\_back(s);

Fellow f;

cout << "Info for a new fellow:" << endl;

cin >> f;

fellows.push\_back(f);

for(auto x : students)

cout << x;

for(auto x : fellows)

cout << x;}

1. Continue working on existing Person and Student classes and add the following classes:

* Professor derived from the Person class. The Professor class must have an additional attribute that is the primary research area.
* PosGradStudent derived from the Student class. The PosGradStudent class must have an additional attribute - the supervisor that should be a professor.
* Discipline that contains information on the subject's name, responsible professor and enrolled students. Imagine that in one discipline you can enroll regular students, fellows and postgraduate students and that the number of students is unlimited. Ensure that the student vector grows dynamically.
* Implement a method String getClassName() in all the hierarchy of the classes derived from Person, that should return a string with the name of the class of the object. Example:

Student s("Maria", 9855678, Date(18, 7, 1990));

cout << s.getClasseName() << endl; // should print Student

Fellow f("Maria", 9855678, Date(18, 7, 1990));

cout << f.getClasseName() << endl; // should print Fellow