// Programming Techniques
Exercises Class # 10

Master Programme in Mathematical Finance
1st Semester 2018/2019
ISEG-UL

Sara Lopes
{
//sblopes@iseg.ulisboa.pt
//saradutralopes@gmail.com
}

Overview

- Newton's Method
- Object Oriented Programming
    - Encapsulation
    - Inheritance
    - Polymorphism

Bibliography:

- Stroustrup, Bjarne, Programming Principles and Practice Using C++, Second Edition, Addison-Wesley, 2014.

Newton's Method

Newton's method is a method for finding the roots of a differentiable real-valued function:

$$x : f(x) = 0$$

Iteration Scheme:

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}}$$

We obtain a series of sucessive approximations of the root starting with the initial guess x_0.

Newton's Method Exercise

Write a program that, using Newton's method, computes an approximation of $\sqrt{2}$ with absolute error smaller than 10^{-5}. How many iterations of the method are necessary?
Obs: Note that $\sqrt{2}$ is the solution of the equation $x^2 - 2 = 0$

Object Oriented Programming - Characteristics

- Everything is an object. An object is a type of variable that stores information and that receives messages
- A program is a set of objects that communicate between them.
- Every object has its own memory created from other objects.
- Every object has a type (class).
- Every object of the same type can receive the same messages

Encapsulation

Class member access specifications:

- **public:** A member declared as public can be accessed outside the class
- **private:** A member declared as private can only be acessed inside the class
- **protected:** Protected members can be accessed by class members and by members of sub classes.

Inheritance

Inheritance allows us to define a class in terms of another class.

When creating a class, instead of writing completely new data members and member functions we can designate that the new class should inherit the members of an existing class. This existing class is called the **base class**, and the new class is referred to as the **derived class**.

Cpolygon.h

```
class CPolygon {
protected:
 int width, height;
public:
 void set_values(int a,int b)
 {widht=a; height=b;}
 int area() { throw errPoli{"Don't␣know␣what␣to␣do"}; }
};

class CRectangle:public CPolygon{
public:
 int area(){return width*height;}
};

class CTriangle: public CPolygon{
public:
 area(){return width*height/2;}
};
```

main.cpp

```
#include "std_lib_facilities.h"
#include "Cpolygon.h"

int main()
{
   CPolygon A;
   A.set_values(1,2);

   CRectangle B;
   B.set_values(3,4);
   cout<<B.area()<<endl;

   CTriangle C;
   C.set_values(3,4);
   cout<<C.area()<<endl;
   return 1;
}
```

Result: 12 6

Polymorphism

Polymorphism is the possibility that a call to a member function can cause a different function to be executed depending on the type of object that invokes the function.

```cpp
void printarea(CPolygon &C){
    cout<<C.area();
}
CRectangle B;
B.set_values(3,4);
printarea(B);
```

Result: "I don't know what to do"

To solve this we use virtual functions:

```cpp
virtual int area() {throw errPol{"Dont␣know␣what␣to␣do"};}
```

Exercises:

- Construct the class Cpolygon and the derived classes in order to run all the code in this slides.
- Construct the class Vehicle and a derived class Car. The Vehicle should have as members: number of wheels, number of doors and a brand. Create a function to show the information about the Vehicle. The Car should have an additional member: color.
  Your code should run with the following commands in the main function:

```
string s1("Mercedes");
    Vehicle V(8,2,s1);
    showinfo(V);
    string s2("BMW");
    string c1("black");
    Car C(4,2,s2,c1);
    showinfo(C);
```