Classes and objects

Objectives:

1.Declaring variabiles
2.Declaring classes
3. public, protected and privat
5. Object instantiation
6. Implicit, explicit and copy constructor
7. Destructor
8. this operator

A short theoretical part:

Define and explain the notions above. (from the lecture).

- A class defines atributes and methods.

```
class X{
        //variables
        //functions
        };
```

- An object is an instantiation of a class and it has a certain state (represented by a value) and a certain behavior (represented by functions) in a certain moment.
- An object oriented program is a collection of objects which interact with one each other by messages (by applying a method).

1. Problem! Repare the existent error(s) from the next program:

```
int main( )
 {
 int i = 1;
 int& r = i; // r si i refers to the same int
 int& r2; // ERROR:
 int x = r; // x = 1
 r = 2; //  i = 2
 int* p = &r; // p points to i
  return 0;
 }

Which are the values for i, r, x after executing the code above?
```

2. What is the value of a after calling the function increment?

```
 void increment(int& i)
 {
      i++;
 }
 void main()
 {
      int a = 5;
      increment(a);
 }
```

- In C++ we define the classes by *class* or *struct*. When using structurs all the members will be public by default, when using classes the members will be private by default.

Suggestion : declare at first public members, then protected, then private.

3. Compile and run! Try to repair the errors!

```
class X
 {
     int x;      // private, by default
  public:
     int y, z; // public
     int f();  // public member function
  private:
     int w;    // private data
  protected:
     int f(int, int); // protected member function
};
int main()
{
     X obj; //the default constructor is called
     obj.x = 10; // ERROR: Cannot access private members
     obj.f();      // OK!
     obj.f(1,1);   // ERROR:Cannot access protected members
       return 0;
}
```

- The object instantiation: Class_Name object_name;

Example: Date myBirthday;
Date* pDate = new Date;

- We can access the members of a class by using the operators "." and "->"; new operators introduced ".*" and "$\rightarrow$*";

4. Write the corresponding class and the main function such that the following program works.

```
void f()
{
Date today;
Date *pdate = &today;

today.init(13,3,2013); // March 13, 2013
printf("Today is %d/%d/%d.",pdate->day,pdate->month,today.year);
}
```

5. Write the corresponding class and the main function such that the following program works

```
typedef int Date::*PM;  //pointer to an int member of the class Date

typedef void (Date::*PMF)(int, int, int);     //pointer to member functions
                                                of the class Date
                                              //which has 3 int arguments
```

```
void f() {
    Date today;
    Date *pdate = &today;

    PM pm = &Date::day;
    PMF pmf = &Date::init;

    (today.*pmf)(14, 3, 2012);       //   today.init(14, 3, 2012);
    today.*pm = 8;                //  today.day = 8

     (pdate->*pmf) (14, 3, 2012);    //  today.init(14, 3, 2012);
     pdate->*pm = 8;             //  today.day = 8
       cout<<today.*pm<<"    "<<pdate->*pm<<endl;
      }
```

- **default constructor** is of the form: X();
- **explicit constructor with default values** :
  Example: `complex(double re=0.0, double im=0.0);`
- **explicit constructor**
  Example: `complex(double re, double im);`
- **copy constructor** is of the form:
  X(const X&);
  Example: `complex(const complex& src);`

- **the destructor** of a class is of the form: ~X();
  A class can have only one single destructor.

DO NOT explicitly call the constructor and the destructor as in the following example:

```
void f()
  {
    String s1;
    s1.String::String("Explicit call");
    s1.~String();
   }
```

Problems:

1. Study the following problem

```
class complex
{
public:
// constructors
      complex(double re=0.0, double im=0.0); // default constructor as well
      complex(const complex& src); // copy constructor
// getters
double getReal() const;
double getImag() const
{
return im;
}
// setters
void setReal(double re);
void setImag(double im);
// auxiliary functions
```

```
void print(ostream& out);

private:
double re, im;
};

complex::complex(double re, double im)
{
this->re = re;
this->im = im;
}
complex::complex(const complex& src)
{
re = src.re;
im = src.im;
cout << "\nCopy-constructor";
}
inline double complex::getReal() const
{
return re;
}
void complex::setReal(double r)
{
re = r;
}
void complex::setImag(double im)
{
this->im = im;
}
void complex::print(ostream& out)
{
out << "\ncomplex [re=" << re << ", im=" << im << "]";
}
void f(complex c)
{
c.setReal(1);
c.print(cout);
}
void g(complex& c)
{
c.setReal(1);
c.print(cout);
}
int main()
{
complex c1, c2(3.4, 6), c3(100), c4=90;
complex c5=c2;
f(c3);
c3.print(cout);
g(c3);
c3.print(cout);
return 0;
} // the destructor is called for all the objects in the reversed order of
declaration
```

2. Study the following problem! Repare the errors (if they exist) and write the corresponding function main.

The file Complex. h

```
#pragma once
class Complex
{
   private double re;
   private double im;
public:
Complex(void);
~Complex(void);
void sum(Complex* z,Complex* zz);
void diff(Complex* z,Complex* zz);
};
```

The file Complex.cpp

```
#include "stdafx.h"
#include "Complex.h"
Complex::Complex(double re,double im)
{
   this.re=re;
   this.im=im;
}
Complex::~Complex(void)
{
   std::cout<<"Object destroyed";
}
void Complex::sum(Complex *z, Complex *zz)
{
     double re=(*z).re+(*zz).re;
     double im=(*z).im+(*zz).im;
      std::cout<<"The sum : ( "<<re<<","<<im<<")\n";
}
void Complex::diff(Complex *z, Complex *zz)
{
     double re=(*z).re -(*zz).re;
     double im=(*z).im -(*zz).im;
     std::cout<<"The difference : ( "<<re<<","<<im<<")\n";
}
```

3. Starting from the problems 1 and 2 create a program which has ca class Complex and use:
a) An explicit constructor.
b) A default constructor.
c) A copy constructor.
d) A destructor.
e) Methods which allow the modification and the access of a component of a complex number:

double getRe( )
void setRe(double re)
double getIm( )
void setIm(double im)

f) A method, which allows the addition of two complex numbers, of the form:
void sum(Complex c)
g) A method, which allows the addition of two complex numbers, of the form:

Complex sum(Complex a, Complex b)

4. Choose one of your daily tasks (not too complex) and describe it in an object oriented way. Try to describe it such that the objects interact one to each other.
Hint: create a class to eat, atributes: an array with what you want to eat, breakfast, lunch (using enum), a method eats and the objects.

5. Write a class Driver. Use your imagination for writing two methods which can describe the bahavior of a driver. This class will have a default constructor, an explicit one and a destructor. Take into account the following: the driver has one single car, cannot carry more than 4 persons in a car ride, has a certain schedule.

Extra problems

6. Create a program C++ which finds the description of a pc with components like: processor, video card, sound card, memory, hard, dvd-rw + optional accesories: webcam, microphone, headphones, wireless, bluetooth... Try to describe it such that the objects interact one to each other.