Clase si obiecte

Obiective:

1.Declararea variabilelor
2.Declararea claselor
3.Nivelurile de acces public, protected si privat
5.Instantierea obiectelor
6. Constructor implicit, explicit şi de copiere
7. Destructor
8.Operatorul this

Scurt teoretic:

Definiti si explicati notiunile de mai sus. (trebuie stiute de la curs).

- O clasa defineste atribute si metode.

```
class X{
        //variabile
        //functii
        };
```

- Un obiect este o instanta a unei clase care are o anumita stare (reprezentata prin valoare) si are un comportament (reprezentat prin functii) la un anumit moment de timp.
- Un program orientat obiect este o colectie de obiecte care interactioneaza unul cu celalalt prin mesaje (aplicand o metoda).

1. Problema! Reparati eroarea (erorile) din urmatorul program:

```
int main( )
 {
 int i = 1;
 int& r = i; // r si i refera la acelasi int
 int& r2; // ERROR:
 int x = r; // x = 1
 r = 2; // defapt i = 2
 int* p = &r; // p pointeaza la i
  return 0;
 }
```

Care vor fi valorile la finalul executiei prelucrarilor pentru i, r, x?

2. Ce valoare va avea a dupa apelul functiei increment?

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

- In C++ clasele le definim prin *class* sau *struct*. In cadrul structurilor membrii vor fi implicit publici, iar in clase membrii clasei vor fi impliciti private.

Este indicat sa declarati mai intai membrii publici, apoi protected, apoi private.

3. Compilati si rulati! Incercati sa reparati erorile!

```
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;
     obj.x = 10; // ERROR: Cannot access private members
     obj.f();       // OK!
     obj.f(1,1);    // ERROR:Cannot access protected members
       return 0;
}
```

- Obiectul se instantiaza: Nume_clasa nume_obiect;

De exemplu: Date myBirthday;

Date* pDate = new Date;

- Accesul la membrii unei clase se face cu operatorii "." si "->"; operatori noi introdusi ".*" si "→ *";

4. Scrieti clasa corespunzatoare si functia main astfel incat programul urmator sa functioneze.

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

today.init(14,3,2012); // March 14, 2012
printf("Today is %d/%d/%d.",pdate->day,pdate->month,today.year);
}
```

5. Scrieti clasa corespunzatoare si functia main astfel incat programul urmator sa functioneze

```
typedef int Date::*PM;  //pointer la un membru de tip int din clasa Date

typedef void (Date::*PMF)(int, int, int);     //pointer la functii membre
                                                 ale clasei Date
                                              //care au 3 argumente de tip
                                                int

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;
  }
```

-constructorul implicit va avea forma: X();
Exemplu: `complex(double re=0.0, double im=0.0);`
-constructor explicit
-constructorul de copiere va avea forma: X(const X&);
Exemplu: `complex(const complex& src);`
-destructorul unei clase are forma: ~X(); O clasa poate avea un singur destructor.

NU este recomandata apelarea explicita a constructorului si a destructorului ca in exemplul urmator: void f()

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

Probleme:

1. Studiati urmatoarea problema

```
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(int, char*[]) {
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;
} // destructorii sunt apelati pentru toate obiectele in ordinea inversa a
declaratiei
```

2. Studiati urmatoarea problema! Reparati erorile (daca exista) si scrieti functia main corespunzatoare.

Fisierul 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);
};
```

Fiserul 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<<"Obiect Distrus";
}
void Complex::sum(Complex *z, Complex *zz)
{
     double re=(*z).re+(*zz).re;
     double im=(*z).im+(*zz).im;
      std::cout<<"Suma : ( "<<re<<","<<im<<")\n";
}
void Complex::diff(Complex *z, Complex *zz)
{
     double re=(*z).re -(*zz).re;
     double im=(*z).im -(*zz).im;
     std::cout<<"Diferenta : ( "<<re<<","<<im<<")\n";
}
```

3. Pornind de la problemele 1 si 2 creati un program care sa contina o clasa Complex si utilizati:
a) Un constructor explicit.
b) Un constructor implicit. Rescrieți constructorul explicit utilizând parametrii impliciți.
c) Un constructor de copiere.
d) Un destructor.
e) Metodele care permit modificarea şi aflarea valorilor componentelor unui număr complex:

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

f) O metodă care permite adunarea unui număr complex care va arata astfel:
void sum(Complex c)
g) O metodă care să permită adunarea a două numere complexe care va arata:
Complex sum(Complex a, Complex b)

4. Alegeti una din sarcinile voastre zilnice (nu foarte complexa) si descrieti-o in forma procedurala si orientata obiect. Incercati sa o descrieti astfel incat obiectele sa interactioneze. Indicatie: creati o clasa a manca, atribute: un vector cu ceea ce vrei sa mananci, tip pranz, mic dejun (cu enumeratie), metoda mananca si obiect.

Probleme suplimentare

5. Creati un program C++ care sa depisteze modelul unui pc cu componente: procesor, placa video, placa de sunet, placa de baza, memorie, hard, dvd-rw + accesorii optionale: webcam, microfon, casti, wireless, bluetooth... Incercati sa descrieti astfel incat obiectele sa interactioneze intre ele.

6. Scrieti o clasa numita Sofer. Utilizati-va imaginatia pentru a concepe doua metode care sa simuleze comportamentul unui sofer. Aceasta clasa va avea un constructor implicit, unul explicit si un destructor. Tineti cont de urmatoarele: soferul are o singura masina, nu poate transporta in masina lui mai mult de 4 persoane, are un anumit program.