Lucrarea 4. Programare orientată pe obiecte. Clase compuse

Scopul lucrării: Familiarizarea cu noţiunile de clasa compusă, lista de iniţializare, pseudo-constructori.

Desfăsurarea lucrării: Se vor scrie programe în cadrul cărora se vor utiliza elementele principale ale compunerii claselor:

• Clase compuse;

• Folosirea listei de iniţializare în cadrul funcţiilor constructor ai unei clase compuse;

• Pseudo-constructori;

• Utilizarea modificatorilor de acces în cadrul claselor compuse.

Se vor utiliza paradigmele de abstractizare a datelor si compunere a claselor, aplicaţiile având fisiere header pentru declararea claselor sifisiere sursă pentru implementarea funcţiilor membre, precum si un fisier sursă pentru testare. Se vor identifica si proiecta modulele corespunzătoare pentru rezolvarea fiecărei aplicaţii.

Declararea si folosirea claselor compuse:

Prin agregarea unor obiecte instanţiate din clase deja construite, într-o clasă nouă se defineste o clasă compusă.

Recomandari:

- Accesul la membrii clasei se face prin getteri si setteri. Expuneti cit mai putine metode membru si nici o variabila membru
- prefix _ pentru variabilele membru private
- ptr variabile membru de tip bool, numite de exemplu xxx, getter-ul va fi isXxx(), in loc de getXxx()

De studiat:

- naming conventions

Adaptati urmatorul cod, mediului Dvs de programare

```
#include <iostream>
#include <string>
using namespace std;

class Student {
private:
    int note;
    string name;
public:
    Student(int n = 0){
        note = n;
    }
    void setNote(int n){
        note = n;
    }
    int getNote(){
        return note;
    }
```

```
};

class StudentsGroup {
private:
    int studentsNumber;
    Student *studentsList;
public:
    StudentsGroup(int n){
        studentsNumber = n;
        studentsList = new Student[studentsNumber];
    }
    void showStudentsInGroup(){
        for(int i=0; i<studentsNumber;i++){
            cout<<i<<" "<<studentsList[i].getNote()<<endl;
        }
    }
};

int main()
{
    StudentsGroup *group = new StudentsGroup(3);
    group->showStudentsInGroup();
    return 0;
}
```

Exercitii:

1.1 Adaptati codul de mai sus astfel incit sa folositi fisiere header

1.2 Modificati codul astfel incit in constructorii Student si StudentsGroup sa folositi cuvintul cheie this.

1.3 Adaugati o variabila membru name, in clasa Student si modificati aplicatia astfel incit name sa fie initializat cu „Joe Doe”

1.4 Implementati o metoda readStudents in clasa StudentsGroup pentru citirea studentilor

Codul obtinut este asemanator celui de mai jos:

```
#include <iostream>
#include <string>
using namespace std;

class Student {
private:
    int note;
    string name;
public:
    Student(int note = 0, string name="Joe Doe"){
        this->note = note;
        this->name = name;
    }
    void setNote(int n){
        this->note = n;
    }
    int getNote(){
        return this->note;
    }
    string getName(){
        return this->name;
    }
    void setName(string name){
        this->name = name;
    }
};

class StudentsGroup {
private:
    int studentsNumber;
    Student *studentsList;
public:
    StudentsGroup(int studentsNumber){
        this->studentsNumber = studentsNumber;
        studentsList = new Student[studentsNumber];
    }
    void showStudentsInGroup(){
        for(int i=0; i<this->studentsNumber;i++){
            cout<<i<<" "<<studentsList[i].getName()<<" "<<studentsList[i].getNote()<<endl;
        }
    }
```

```
    void readStudentGroup(){
        int note;
        string studentName;
        for(int i=0; i<this->studentsNumber;i++){
                cout<<"student "<<i<<" name:";
                cin>>studentName;
                cout<<"note ";
                cin>>note;
                studentsList[i].setNote(note);
                studentsList[i].setName(studentName);
        }
    }
};

int main()
{
    StudentsGroup *studentsGroup = new StudentsGroup(3);
    studentsGroup->readStudentGroup();
    studentsGroup->showStudentsInGroup();
    return 0;
}
```

Exercitii Suplimentare:

2.1 Optimizati codul de mai sus astfel incit in loc de studentsList[i].setNote(note) si studentsList[i].setName(studentName); sa folosim studentsList[i] = Student(note, studentName);

2.2 Studiati daca este posibila extinderea tabelului de elemente Student cu cite un element cind se doreste introducerea unui nou student.

2.3 Implementati studentsList cu lista simplu sau dublu inlantuita de Student

2.4 Scrieti o metoda care sorteaza tabloul / lista de Student, studentsList, dupa name/note

2.5 Scrieti o metoda care returneaza Student cu nota maxima

2.6 Scrieti o metoda care returneaza primul element Student cu note egala cu 5. Considerati cazurile studentsList ca tabloul / lista. Ce returneaza in caz ca nu a gasit un asemenea element ?

Pseudo-constructori

Adaptati urmatorul cod, mediului Dvs de programare

```
#include <iostream>
#include <string>

using namespace std;
class A{
int i;
string s;
public:
A(int i1, string s1): i(i1), s(s1){}            // notati modul de apel ptr i si s din A !
int getI(){
    return this->i;
}
string getS(){
    return this->s;
}
};

class B{
A a;
string s;
public:
B(int i2,string s2): a(i2,s2), s("init value for s in A") {}        // notati modul de apel ptr constructor
string getS(){
    return this->s;
}
A getA(){
    return this->a;
}
};
```

```
int main()
{
   B *b = new B(5, "init value for s in class A");
   cout<<"B s:"<<b->getS()<<endl;
   cout<<"A s:"<<b->getA().getI()<<" i:"<<b->getA().getS()<<endl;
   return 0;
}
```

Exercitii

3.1 Modificati codul de mai sus astfel incit sa folositi B b in loc de B *b

Exercitii Suplimentare:

4.1 Investigati folosirea pseudo-constructorilor impreuna cu membrii variabile ca si pointeri