Clase, obiecte. Membri statici ai unei clase. Probleme

Membri statici trebuie definiti si initializati undeva (in interiorul clasei ei sunt doar declarati).

-functiile statice nu primesc pointerul "*this"*

-accesul la membri nestatici nu se poate face din functii de tip static

-crearea initializarea si accesul la acesti membrii sunt independente de existenta obiectelor.

Exemplu:

```
class Date {
    int day, month, year;
    static Date today; // declare the static data member
public:
    // …
    static void initToday();
};

Date Date::today; // create the static data member
```

Membri const -datele de tip const nu pot fi modificate;

-sintaxa: *const tipul_datei nume_data;*

- functiile de tip const nu pot modifica starea unui obiect

-cand functia este definita in afara clasei, este cerut sufixul const

-sintaxa: *tip_functie nume_functie(lista_arg) const;*

Exemplu:

```
class Date {
    int day, month, year;
    static Date today; // declare the static data member
public:
    // const, inline member function
    int getDay() const {
        day = 0 ; // ERROR: we're in const function
        return day;
    }
    int getMonth() const {
```

```
        return month;
    }
    int getYear() const {
        return year;
    }
};

int main() {
    Date d;
    cout << d.getDay() << d.getMonth() << d.getYear();
    return 0;
}
```

Membri mutable –pot fi intotdeauna modificati, chiar si din functii constante!

-este indicat sa folosim mutable atunci cand vrem ca valoarea sa se modifice in cadrul unei functii const

-sintaxa: *mutable tipul_datei nume_data;*

Probleme

1. **Creati un program C++ care sa functioneze pentru urmatoarea functie main:**

```
void main()
   {
   cout<<"/////////// ----- clasa Mate ------ \\\\\\\\"<<endl;
      Mate ob1(3,4);
      ob1.inmultire();
      ob1.impartire();

      Mate ob2,ob3(5,5);
      ob2.inmultire();
      cout<<"Aduna "<<ob3.adunare()<<endl;
    }
```

2. **Creati un program C++ care sa functioneze pentru urmatoarea functie main:**

```
void main()
{
   Persoana p1;
   p1.get_pers(); //se vor afisa numele, orasul, sexul,
                  //inaltimea si cati ani are persoana p1
   Persoana p2("Ana Maria","Deva","F",1.63,20);
   p2.get_pers();

   p2.calatoreste();  //se va introduce de la tastatura daca
                   //persoana calatoreste (D/N), daca 'D'
                   //atunci se va afisa unde merge si daca 'N'
                   //atunci se va afisa un mesaj
   cout<<"Prima persoana se ocupa cu "<<p1.se_ocupa_cu()<<endl;
   p2.mananca(); //se vor afisa felurile de mancare (in functie
                   // de ce se optiune se alege) mic dejun,
                   // pranz sau cina.
}
```

3. **Reparati (adaugati si modificati) programul C++ urmator in asa fel incat la apelarea variabilei nrobiecte in functia main sa se afiseze pe ecran cate obiecte sunt create la clasa Obiect:**

3.3

```
class Obiect
{
public:
  static int nrobiecte;
      Obiect(int);
      Obiect(const Obiect&);
};
int Obiect::nrobiecte=0;
Obiect::Obiect(int a)
{
      Obiect::nrobiecte++;
}
Obiect::Obiect(const Obiect& b)
{
      cout<<"Obiect: constructor de copiere "<<endl;
      Obiect::nrobiecte++;
}
```

In functie de cate obiecte la clasa Obiect veti crea atata va fi valoarea nrobiecte.
Indicatie: testate urmatoarele programe pentru a vedea utilitatea cuvantului cheie static:

3.1 Exemplu membru static: Efect

```
#include <iostream>

using namespace std;
void afisstatic( int c )
{
static int nStatic;   // Valoarea lui nStatic este retinuta
// intre fiecare apel de functie
   nStatic += c;
   cout << "nStatic este " << nStatic << endl;
}

int main() {
   for ( int i = 0; i < 5; i++ )
      afisstatic( i );
}
Output:
nStatic este 0
nStatic este 1
nStatic este 3
nStatic este 6
nStatic este 10
```

3.2 Utilizarea in clase:

```
#include <iostream>

using namespace std;
class CMyClass {
public:
   static int m_i;
};

int CMyClass::m_i = 0;

int main() {
   cout << CMyClass::m_i << endl;
   cout << CMyClass::m_i << endl;
```

```
    CMyClass::m_i = 1;
    cout << CMyClass::m_i << endl;
    cout << CMyClass::m_i << endl;
}
```

Output:

```
0
0
1
1
```

4. 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.

TEMA

1. Creati o clasa Masina, care sa aiba culoare, motor, cp, carburant, an fabricatie, tip, locuri, imbunatatiri. Avand in vedere ca masina are doar un singur volan, are doar 4 roti si e condusa de o singura persoana creati un program orientat obiect care sa contina toate cele enuntate mai sus.

2. (Optional) Creati un program care sa contina: o clasa Student care contine nume, an, grupa,..., o clasa Cursuri care contine cursurile la care participa un student si o clasa Sali care contine salile unde are cursuri un student. Tineti cont ca doua sau mai multe cursuri nu se pot desfasura in aceeasi sala si afisati cel putin 3 studenti la ce cursuri participa si in ce sala. Utilizati toate cele enumerate mai sus (constructori, destructor, ...).

3. Fiind dat programul:

```
class Date
{
     int d,m,y;
public:
     int day() const { return d; }
     int month() const { return m; }
```

```
     int year() const;
     //......
};
inline int Date::year() const
{
     return y++; //error: attempt to change member value in
const function
}
inline int Date::year()
{                          // error: const missing in member
function type
     return y;
}
void f(Date& d, const Date& cd)
{
     int i=d.year();//ok
    d.add_year(1); //ok
    int j=cd.year(); //ok
     cd.add_year(1); //error: cannot change value of const
cd
}
```

Reparati erorile!!

4. (Optional) Pentru clasa Sir de mai jos scrieti constructorii potriviti, alte metode si cateva obiecte la clasa Sir.

```
class Sir
{
     enum {DIM=100};
     int n;
     char s[DIM];
public:
     //constructor
   };
```

Lecturati cap.10 din cartea „The C++ Programming Language”, Bjarne Stroustrup 3rd Edition!!!!!!