Seminar 2

Exerciții tip examen:

1. Descrieţi pe scurt diferenţa dintre funcţiile care returnează valoare şi cele care returnează referinţă. (Examen 2005)
   Răspuns:
   O funcție care returnează o valoarea returnează o copie a unui obiect de tipul de retur al funcției. O funcție care returnează o referință oferă acces direct la obiectul original din memorie. Orice modificare efectuată asupra valorii returnate afectează și obiectul original. Acest tip de funcție este util pentru a evita copierea inutilă a obiectelor mari și pentru a permite modificarea datelor direct.
   Atenție:
   Trebuie să fim atenți când facem returnări prin referință: trebuie să ne asigurăm că obiectul la care se face referire există și după ce funcția returnează referința. În caz contrar, referința returnată va deveni suspendată (indicând un obiect care a fost distrus), iar utilizarea acestei referințe va duce la un comportament nedefinit.(mai multe detalii aici)
2. Suneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, altfel, spuneţi de ce nu este corect. (Examen 2005)

```
#include <iostream>
class cls1
{ int x;
public: cls1(){ x=13; }
   int g(){ static int i; i++; return (i+x); } };
class cls2
{ int x;
public: cls2(){ x=27; }
   cls1& f(){ static cls1 ob; return ob; } };
int main()
{   cls2 ob;
   std::cout<<ob.f().g();
   return 0;
}
```

Răspuns:
Afișează 14
Întrebări suplimentare:
Ce se întamplă dacă mai afisăm o dată ob.f().g()?
Ce se întâmplă dacă obiectul din funcția f nu e static?

3. Cum trebuie definită variabila n din clasa de mai jos, dacă dorim ca ea să contorizeze numărul tuturor obiectelor care aparţin clasei cls la un moment dat. (Examen 2005)

```
class cls {
   int n;
   public:
       cls() { n ++; }
       ~cls(){ n --; }
};
```

Răspuns:
static int n

4. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, altfel, spuneţi de ce nu este corect. (Examen 2005)

```
class cls {
   int x;
   public:
       cls(int y) {x=y; }
       int operator*(cls a, cls b) {
           return (a.x*b.x);
       }
};

int main() {
   cls m(100),n(15);
   std::cout<<m*n;
   return 0;
}
```

Răspuns:
Nu compilează, operatorul * va aștepta 3 parametri (obiectul care apelează operatorul plus alte 2 obiecte transmise ca parametrii).
Ca să rezolvăm eroarea, facem suprascrierea operatorului ca funcție friend și păstrăm cei 2 parametrii.

5. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ spuneţi de ce nu este corect. (Examen 2005)

```
class cls {
   int vi;
   public:
       cls(int v=37) { vi=v; }
```

```
   friend int& f(cls);
};

int& f(cls c) { return c.vi; }

int main() {
   const cls d(15);
   f(d)=8;
   std::cout<<f(d);
   return 0;
}
```

Răspuns:
Nu compilează deoarece f întoarce o referință la un obiect temporar (se crează o copie a lui d, transmisă în funcția f sub numele de c, care va fi distrus la ieșirea din funcție, deci nu va exista în main). Pentru a rezolva problema, trebuie să transmitem parametru în funcția f ca referință (atenție, se modifică și la declararea funcției friend), dar acum apare altă problemă: se încalcă promisiunea de const pentru obiectul d. Ștergem const de la d.

6. Spuneţi pe scurt prin ce se caracterizează o metodă statică a unei clase.
   Răspuns:
   Metodele statice sunt metode care pot fi apelate fără a instanția un obiect de tipul clasei, fiind specifice clasei și nu obiectului. O metodă statică poate accesa doar date sau metode statice ale unei clase.
7. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, altfel, spuneţi de ce nu este corect.

```
#include <iostream>

class cls {
   int x;
public:
   cls(int i = 25) { x = i; }
   int f();
};

int cls::f() { return x; }

int main() {
   const cls d(15);
```

```
    std::cout << d.f();
    return 0;
}
```

Răspuns:
Nu compilează, este încălcată promisiunea de const a obiectului d. Avem 2 metode să rezolvăm această eroare: 1. îl facem pe d neconstant sau 2. facem metoda constantă (int cls::f() const {return x;}). Atenție că pt 2 trebuie să schimbăm și antetul din clasă (int f() const;)

8. Spuneţi de câte ori se apelează destructorul clasei cls în programul de mai jos. Justificaţi.

```
class cls {
public:
    int x;
    cls(int i = 0) { x = i; }
};

cls f(cls c) {
    c.x++;
    return c;
}

int main() {
    cls r(10);
    cls s = f(r);
    return 0;
}
```

Răspuns:
De 3 ori. Dacă întrebarea ar fi pusă pentru constructor, în funcție de ce versiune de c++ rulează pe calculator, răspunsul poate varia. În mod normal, dacă distrugem 3 obiecte, ar trebui și să construim 3 obiecte, dar din cauza unor optimizări la nivel de compilator, care evită copierea obiectelor dacă ele nu sunt folosite ulterior, constructorul va fi apelat o singură dată, pentru obiectul r. Această optimizare poartă numele de return value optimization sau copy-epsilon. Mai multe detalii puteți să găsiți aici.

9. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneți o (singură) modificare prin care programul devine corect. (examen 2016)

```
#include <iostream>
using namespace std;
class A
{   protected: static int x;
   private: int y;
   public: A(int i) { x=i; y=-i+3; }
           int put_x(A a) { return x+a.y; }
};
int A::x=5;
int main()
{
   A a(10);
   cout<<a.put_x(70);
   return 0;
}
```

Răspuns:
Compilează, afișează 3.

10. Descrieţi pe scurt lista de iniţializare a constructorilor și utilizarea ei (sintaxă, utilizare, moștenire, compunere). (examen 2016)

Răspuns:
Sintaxa 0.1p
Utilizare: se transmit valori catre clasa de baza (clasele mostenite),
se transmit valori catre constructori din obiecte compuse, initializare campuri statice, etc 0.3p
fara prostii 0.1p

11. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneți o (singură) modificare prin care programul devine corect. (examen 2016)

```
#include <iostream>
using namespace std;
class cls
{   int x;
```

```
public: cls(int i) { x=i; }
   int set_x(int i) { int y=x; x=i; return y; }
   int get_x(){ return x; } };
int main()
{  cls *p=new cls[10];
   for(int i=3;i<9;i++) p[i].set_x(i);
   for(int i=0;i<4;i++) cout<<p[i].get_x();
   return 0;
}
```

Răspuns:

Nu compilează, nu are constructor fără parametru. Punem în constructor valoare default (cls(int i =0)).

12. Spuneţi dacă programul de mai jos este corect. În caz afirmativ, spuneţi ce afişează, în caz negativ propuneți o (singură) modificare prin care programul devine corect.Răspuns:

```
#include <iostream>
using namespace std;
struct X
{
   int i;
   public:
       X(int ii ) { i = ii; };
       int f1() const { return i; }
       X f2() const {   int i=this->f1(); return X(74-i); }
};
const X f3() {  return X(8); }

int f4(const X& x) { return x.f1(); }

int main()
{
   X ob(19);
   cout<<f4(ob.f2());
   return 0;
}
```

Răspuns:

Afișează 55.

13. Spuneți dacă programul de mai jos este corect:

```
class A
{
   int x;
public:
   A(int i = 25) { x = i; }
   int& f() const { return x; }
};
int main()
{
   A ob(5);
   cout << ob.f();
   return 0;
}
```

Răspuns:
Nu compilează deoarece o funcție constantă nu are voie sa returneze o adresa. Dacă asta ar fi posibil, ar însemna că orice modificare asupra rezultatului lui f se va vedea asupra pointerului this.Ca programul sa compileze modificăm in int f() const sau în int& f().

14. Spuneți dacă programul de mai jos este corect:

```
class problema {
   int i;

public:
   problema(int j = 5) { i = j; }
   void schimba() { i++; }
   void afiseaza() { cout << "starea curenta " << i << "\n"; }
};
problema mister1() { return problema(6); }
void mister2(problema& o)
{
   o.afiseaza();
   o.schimba();
   o.afiseaza();
}
int main()
{
   mister2(mister1());
```

```
    return 0;
}
```

Răspuns:
Nu compilează, deoarece mister2 primește ca parametru o referință, dar obiectul returnat de mister1 este constant. Așadar, se încalcă promisiunea lui const. Ca să rezolvăm problema, scoatem referința de la parametrul lui mister2.