Grupa 20 septembrie 2009

Proiectare şi programare orientate pe obiecte

Examen scris - Restanță

I. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afişează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{     protected: int x;
      public: A(int i):x(i){ }
              int get_x() { return x; } };
class B: public A
{     public: B(int i):A(i) {}
              operator int() {return x; }
              B operator+(B& b) const {return x+b.x; } };
int main()
{     const B a(22), b(-12);
      cout<<a+b;
      return 0;
}
```

<< nu este def. ~~să~~ afișează obiecte și nici nu există conversie de la A() la un tip standard

II. Descrieți pe scurt în ce constă mecanismul de încapsulare.

III. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afişează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{     int x;
      public: A(int i=25) { x=i; }
      int& f() const { return x; } };
int main()
{     A a(15);
      cout<<a.f();
      return 0;
}
```

~~se poate~~ nu se poate converti ~~int~~ const int& la int&.

IV. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

o metodă statică nu poate lucra cu variabile non-statice ale clasei (pentru că metodele non-statice)

```
#include<iostream.h>
class A
{     int x;
      const int y;
      public: A(int i, int j):x(i), y(j) { }
              static int f(int z, int v) { return x+z+v; } };
int main()
{ A ob(5,-8);
  cout<<ob.f(-9,8);
  return 0;
}
```

V. Spuneți ce este obiectul implicit al unei metode și descrieți pe scurt proprietățile pe care le cunoașteți despre acesta.

VI. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{     int x;
      public: A(int i):x(i){}
              int get_x() const { return x; } };
class B: public A
{     int *y;
      public: B(int i):A(i){ y=new int[i];
                             for(int j=0; j<i; j++) y[j]=1; }
        B(B&);
              int& operator[](int i) { return y[i]; } };
B::B(B& a)
{ y=new int[a.get_x()];
  for(int i=0;i<a.get_x();i++) y[i]=a[i];
}
ostream& operator<<(ostream& o, B a)
{ for(int i=0;i<a.get_x();i++) o<<a[i];
  return o;
}
int main()
{ B b(5);
  cout<<b;
  return 0;
}
```

nu există constructor fără parametri în clasa A()

VII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
#include<typeinfo.h>
class A
{ int i;
  public: A() { i=1; }
          int get_i() { return i; }
};
class B: public A
{ int j;
  public: B() { j=2; }
          int get_j() {return j; }
};
int main()
{ A *p;
  int x=0;
  if (x) p=new A;
  else p=new B;
  if (typeid(p)==typeid(B*)) cout<<((B*)p)->get_j();
  else cout<<"tipuri diferite";
  return 0;
}
```

typeid nu evaluează # tipuri de date ci dif. pointeri

VIII. Descrieți pe scurt moștenirea virtuală și scopul în care este folosită.

.

IX. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{     int x;
      public: A(int i=17) { x=i; }
      int get_x() { return x; } };
class B
{     int x;
      public: B(int i=-16) { x=i; }
      operator A() { return x; }
      int get_x() { return x; } };
int main()
{     B a;
      A b=a;
      cout<<b.get_x();
      return 0;
}
```

-16

X. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{    protected: int x;
     public: A(int i=-16) { x=i; }
             virtual A f(A a) { return x+a.x; }
             void afisare(){ cout<<x; } };
class B: public A
{    public: B(int i=3):A(i) {}
             A f(A a) { return x+a.x+1; } };
int main()
{ A *p1=new B, *p2=new A, *p3=new A(p1->f(*p2));
  p3->afisare();
  return 0;
}
```

XI. Descrieți pe scurt proprietățile unui câmp constant al unei clase.

XII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{    public: int x;
     A(int i=0) { x=i; }
     A operator+(A a) { return A(x+a.x); } };
ostream& operator<<(ostream& o, A a) { o<<a.x; return o; }
template <class T>
class B
{            T y;
     public: B() {}
             B(T i) { y=i; }
             template <class U> B operator+(B<U> ob) { return (ob.y+1); }
             void afisare(){ cout<<y; } };
int main()
{    B<int> b1(-15); B<A> b2(1);
     (b1+b2).afisare();
     return 0;
}
```

XIII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include <iostream.h>
template<class T>
T f(T x, T y)
{ return x+y;
}
int f(int x, int y)
{ return x-y;
}
int main()
{ float a=-15, b=8;
  cout<<f(a,b);
  return 0;
}
```

-7

XIV. Descrieți pe scurt mecanismul de tratare a excepțiilor.

XV. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{     int x;
      public: A(int i=7) { x=i; }
              int get_x() { return x; }
              operator int() { return x; } };
class B: public A
{     public: B(int i=-12):A(i) {}
              B operator+(B a) { return get_x()+a.get_x()+1; } };
int main()
{ B a; int b=-21;
  a+=b;
  cout<<b;
  return 0;
}
```

nu este definit operatorul +=

XVI. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{     public: int x;
              A(int i=0) { x=i; }
              virtual A minus() { return(1-x); } };
class B: public A
{     public: B(int i=0) { x=i; }
              void afisare() { cout<<x; } };
int main()
{     A *p1=new B(18);
*p1=p1->minus();
dynamic_cast<A*>(p1)->afisare();
      return 0;
}
```

p1 nu are metoda afișare (cls. A nu are def. metoda)

XVII. Descrieți pe scurt diferența dintre un pointer și o referință.

XVIII. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează pentru o valoare întreagă citită egală cu 23, în caz negativ spuneți de ce nu este corect.

```
#include<iostream.h>
class A
{     int x;
      public: A(int i=2):x(i){}
              int get_x() const { return x; } };
class B: public A
{     int *y;
      public: B(int i=2):A(i){ y=new int[i];
                               for(int j=0; j<i; j++) y[j]=1; }
              B(B& b){ y=new int[b.get_x()];
                       for(int i=0;i<b.get_x();i++) y[i]=b[i]; }
              int& operator[](int i) const { return y[i]; } };
ostream& operator<<(ostream& o, const B b)
{ for(int i=0;i<b.get_x();i++) o<<b[i];
  return o;
}
int main()
{ const B b(5);
  cout<<b;
  return 0;
}
```

<< trebuie să primească pe b prin referință constantă