Grupa 153 13 iunie 2008
Proiectare şi programare orientate pe obiecte

Examen scris

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 B
{     int x;
      public: B(int i=2):x(i){}
              int get_x() const { return x; } };
class D: public B
{     int *y;
      public: D(int i=2):B(i){ y=new int[i];
                               for(int j=0; j<i; j++) y[j]=1; }
              D(D& a){ y=new int[a.get_x()];
                       for(int i=0;i<a.get_x();i++) y[i]=a[i]; }
              int& operator[](int i) const { return y[i]; } };
ostream& operator<<(ostream& o, const D& a)
{ for(int i=0;i<a.get_x();i++) o<<a[i];
  return o;
}
int main()
{ D ob(5);
  cout<<ob;
  return 0;
}
```

REZ: 11111

II. Descrieți trei metode de proiectare diferite prin care elementele unei clase se pot regăsi în dublu exemplar, sub diverse forme, în definiția altei clase.

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 B
{     protected: int x;
      public: B(int i=10) { x=i; }
              int get_x() { return x; } };
class D: public B
{     public: D(int i):B(i) {}
              D operator+(const D& a) {return x+a.x; } };
int main()
{ D ob1(7), ob2(-12);
  cout<<(ob1+ob2).get_x();
  return 0;
}
```

REZ: -5

IV. 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 B
{    protected: int x;
     public: B(int i=16) { x=i; }
             B f(B ob) { return x+ob.x; }
             void afisare(){ cout<<x; } };
class D: public B
{     public: B f(B ob) { return x+1; } };

int main()
{ B *p1=new D, *p2=new B, *p3=new B(p1->f(*p2));
  p3->afisare();
  return 0;
}
```

REZ: 32

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

VI. 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 cls
{     int *v,nr;
      public: cls(int i) { nr=i; v=new int[i];
                           for (int j=1; j<nr; j++) v[j]=0; }
              int size() { return nr; }
              int operator[](int i) { return *(v+i); } };
int main()
{ cls x(10);
  x[4]=-15;
  for (int i=0; i<x.size(); i++) cout<<x[i];
  return 0;
}
```

REZ: [] nu este left-value

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>
class cls
{     int x;
      public: cls(int i=3) { x=i; }
              const int& f(){ return x; } };
int main()
{ const cls a(-3);
  int b=a.f();
  cout<<b;
  return 0;
}
```

REZ: obiectul const a un poate apela o metoda neconst

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 B
{ static int x;
  int i;
  public: B() { x++; i=1; }
          ~B() { x--; }
          static int get_x() { return x; }
          int get_i() { return i; }
};
int B::x;
class D: public B
{ public: D() { x++; }
          ~D() { x--; }
};
int f(B *q)
{ return (q->get_i())+1;
}
int main()
{ B *p=new B;
  cout<<f(p);
  delete p;
  p=new D;
  cout<<f(p);
  delete p;
  cout<<D::get_x();
  return 0;
```

REZ: 221

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 B
{     int x;
      public: B(int i=1) { x=i; }
              int get_x() { return x; }
              operator int() { return x; } };
class D: public B
{     public: D(int i=-1):B(i) {}
              D operator+(D a) { return get_x()+a.get_x()+1; } };
int main()
{ D a, b;
  b=27+a;
  cout<<b;
  return 0;
}
```

REZ: 26

XI. Enumerați 3 metode de implementare a polimorfismului de compilare.

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 cls
{     static int x;
public: int f(cls a) { return a.x++; }
        static int g(cls a) { return a.f(a)/2; } };
int cls::x=7;
int main()
{ cls ob;
  cout<<cls::g(ob);
  return 0;
}
```

REZ: 3

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>
class cls
{     int *v,nr;
      public: cls(int i=0) { nr=i; v=new int[i];
                             for (int j=0; j<size(); j++) v[j]=3*j; }
        ~cls() { delete[] v; }
              int size() { return nr; }
              int& operator[](int i) { return v[i]; }
              cls operator+(cls); };
cls cls::operator+(cls y)
{   cls x(size());
    for (int i=0; i<size(); i++) x[i]=v[i]+y[i];
    return x; }
int main()
{ cls x(10), y=x, z;
  x[3]=y[6]=-15;
  z=x+y;
  for (int i=0; i<x.size(); i++) cout<<z[i];
  return 0;
}
```

REZ: nedeterminat

XIV. Descrieți pe scurt comportamentul operatorului `dynamic_cast`.

XV. Spuneți dacă programul de mai jos este corect. În caz afirmativ, spuneți ce afișează pentru o valoare întreagă citită egală cu `15`, în caz negativ spuneți de ce nu este corect.

```
#include <iostream.h>
int f(int y)
{ if (y<0) throw y;
  return y/2;
}
int main()
{ int x;
  try
  {
  cout<<"Da-mi un numar par: ";
  cin>>x;
  if (x%2) x=f(x);
  else throw x;
  cout<<"Numarul "<<x<<" e bun!"<<endl;
  }
  catch (int i)
  {  cout<<"Numarul "<<i<<" nu e bun!"<<endl;
  }
  return 0;
}
```

REZ: Numarul 7 e bun!

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 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];
  int i=0;
  for(;i<10;i++) p[i].set_x(i);
  for(i=0;i<10;i++) cout<<p[i].get_x(i);
  return 0;
}
```

REZ: un are constructor fara parametri

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, în caz negativ spuneți de ce nu este corect.

```
#include <iostream.h>
template<class T>
int f(T x, T y)
{ return x+y;
}
int f(int x, float y)
{ return x-y;
}
int main()
{ int a=5, b=8;
  cout<<f(a,b);
  return 0;
}
```

REZ: 13