Mai 2009 Grupa 151

1.

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

2. Crearea dinamica de obiecte

3.

```
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;}
```

4.
```
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 + (const B& b) {return x + b.x + 1;}   };
int main()
{       B a(2), b(-12);
        cout << (a + b) + a;
        return 0;  }
```

5. Proprietatile campurilor statice

6.
```
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;  }
```

7.
```
class A
{       int x;
        const int y;
        public:
        A(int i, int j) : x(i), y(j){}
```

```
        int f(int, int) const;   };

int A :: f(int i, int j) const{return x ++;}
int main()
{       A ob(5, -8);
        cout << ob.f(-9, 8);
        return 0;  }
```

8. Proprietatile destructorului

```
9. 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];
}
int main()
{       B b(5);
```

```
        cout << b;
        return 0;  }
```

10.
```
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).name() == "B")
                cout << ((B*)p) -> get_j();
        else
                cout << "tipuri diferite";
        return 0;
}
```

11. Descrieti pe scurt diferenta dintre parametrii transmisi prin pointeri si cei prin referinta.

12.
```
class A
{       int x;
        public:
                A(int i = 17){x = i;}   };
class B
{       int x;
        public:
                B(int i = -16){x = i;}
                operator A () {return x;}
                int get_x() {return x;}    };
int main()
{       A a;
```

```
        B b = a;
        cout << b.get_x();        return 0;  }
```

13.
```
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;
}
```

14. Ce repezinta o functie virtuala si in ce conditii o functie virtuala defineste o clasa abstracta.

15.
```
class A
{       public:
                int x;
                A(int i = 0) {x = i;}
                A operator + (const 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; }
                B operator + (B ob) {return B(ob.y + 1);}
```

```
        void afisare(){cout << y;}  };

int main()
{       B<int> b1(-15); B<A> b2(1);
        (b1 + b2).afisare();
        return 0;
}
```

16.
```
template <class T>
T f(T x, T y)
{       return x + y;  }

int f(int x, int y)
{       return x - y;  }

int main()
{       int* a = new int(5), *b = new int(8);
        cout << f(a, b);
        return 0;  }
```

17. RTTI

18.
```
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();}    };
int main()
{       B a; int b = -21;
        b += a;
        cout << b;
        return 0;  }
```