Exemplu:

========== Virtual1.cpp ==========

```
#include <iostream.h>

class A{
            int i;
      public:
            A(int i = 0) {
              this->i = i;
            }

            void print() {
              cout << "Baza " << i << endl;
            }
};

class B:public A{
            int i;
      public:
            B(int i = 0): A(i + 1) {
              this->i = i;
            }

            void print() {
              cout << "Derivata " << i << endl;
            }
};

void main(void){
  A a;
  B b;

  a.print();
  b.print();

  A *pa;
  B *pb;

  pa = new A(2);
  pb = new B(20);

  pa->print();
  pb->print();

  pa = pb;
  pa->print();
}
```

========== Virtual2.cpp ==========

```
#include <iostream.h>

class A{
            int i;
      public:
            A(int i = 0){
              this->i = i;
            }

            virtual void print(){
              cout << "Baza " << i << endl;
            }
```

```
};

class B:public A{
            int i;
      public:
            B(int i = 0): A(i + 1) {
              this->i = i;
            }

            void print() {
              cout << "Derivata " << i << endl;
            }
};

class C: public B{
      public:
            void print(){
              cout << "C class" << endl;
            }
};

void main(void){
  A a;
  B b;

  cout << sizeof(A) << endl;
  cout << sizeof(B) << endl;
  a.print();
  b.print();

  A *pa;
  B *pb;

  pa = new A(2);
  pb = new B(20);

  pa->print();
  pb->print();

  pa = pb;
  pa->print();

  C *pc = new C;
  pc->print();

  pa = pc;
  pa->print();
}
```

========== Virtual3.cpp ==========

```
#include <iostream.h>

class A{
            int i;
            char *p;
      public:
            A(int i = 0){
              this->i = i;
            }
```

```
            ~A(){
               cout << "Destructor baza" << endl;
            }
};

class B:public A{
            int i;
      public:
            B(int i = 0): A(i + 1){
              this->i = i;
            }

            ~B(){
              cout << "Destructor derivat" << endl;
            }
};

void main(void){
  A *pa;
  B *pb;
  pa = new A(1);
  pb = new B(10);

  delete pa;
  //pa->~A();
  delete pb;
  pa = pb;
}
```

========== Point.h ==========

```
// class Location contine pozitia de coordonate X si Y pe ecran
// class Point spune daca un punct este vizibil sau nu

enum Boolean {false, true};

class Location {
      protected:         // permite claselor derivate sa acceseze date private
            int X;
            int Y;

      public:
            Location(int initX, int initY);
            int getX();
            int getY();
};

class Point : public Location {      // derivata din clasa Location
// derivarea publica inseamna ca X si Y sunt protected in Point

      protected:
            Boolean visible;

      public:
            Point(int initX, int initY);      // constructor
            virtual void show();
            virtual void hide();
            Boolean isVisible();
            void moveTo(int newX, int newY);
};
```

========== Point.cpp ==========

```
#include "vpoint.h"
```

```
#include <graphics.h>

Location::Location(int initX, int initY) {
   X = initX;
   Y = initY;
};

int Location::getX(void) {
   return X;
};

int Location::getY(void) {
   return Y;
};

// functiile membru ale clasei Point: se presupune
// ca programul principal a initializat modul grafic

Point::Point(int initX, int initY) : Location(initX,initY) {
   visible = false;
};

void Point::show(void) {
   visible = true;
   putpixel(X, Y, getcolor());       // utilizeaza culoarea implicita
};

void Point::hide(void) {
   visible = false;
   putpixel(X, Y, getbkcolor());    //pentru stergere se utilizeaza culoarea de fond
};

Boolean Point::isVisible(void) {
   return visible;
};

void Point::moveTo(int newX, int newY) {
   hide();          // punctul curent devine invizibil
   X = newX;        // modifica X si Y
   Y = newY;
   show();          // arata punctul la noile coordonate
};
```

========== Circle.cpp ==========

```
#include <graphics.h>
#include "vpoint.h"
#include <conio.h>

class Circle : public Point {
            int radius;           // membru privat

      public:
            Circle(int initX, int initY, int initRadius);
            void show(void);
            void hide(void);
            void expand(int expandBy);
            void contract(int contractBy);
};

Circle::Circle(int initX, int initY, int initRadius) : Point(initX, initY) {
   radius = initRadius;
}
```

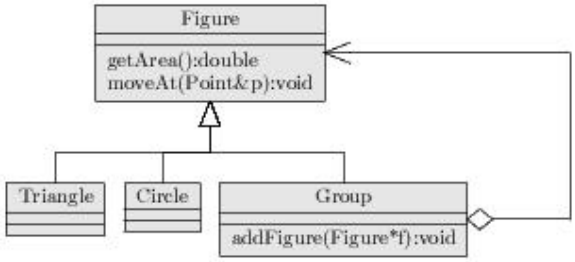
```
void Circle::show() {
   visible = true;
   circle(X, Y, radius);      // deseneaza cercul
}

void Circle::hide() {
   if (!visible)
      return;                 // nu este nevoie sa ascundem cercul

   unsigned int tempColor;    // salveaza culoarea curenta
   tempColor = getcolor();
   setcolor(getbkcolor());    // seteaza culoarea de desenare la culoarea
fondului
   visible = false;
   circle(X, Y, radius);      // sterge cercul
   setcolor(tempColor);       // seteaza culoarea inapoi la culoarea initiala
}

void Circle::expand(int expandBy) {
   Boolean vis = visible;  // este cercul vizibil?
   if (vis)
     hide();        // daca da, ascunde-l
   radius += expandBy;     // mareste raza
   if (radius < 0)
      radius = 0;
   if (vis)
     show();        // deseneaza noul cerc daca inainte fusese vizibil
}

inline void Circle::contract(int contractBy) {
   expand(-contractBy);
}

void main() {
   // initializeaza grafica
   int graphdriver = DETECT, graphmode;
   initgraph(&graphdriver, &graphmode, "");

   Circle circle(50, 100, 25);
   circle.show();
   getch();
   circle.moveTo(100, 125);

   getch();
   circle.expand(25);
   getch();
   circle.contract(35);
   getch();
   closegraph();
}
```

Tema

1. Realizați un container eterogen de figuri. Containerul poate gestiona în acelaşi timp cercuri, triunghuri, pătrate. Fără a modifica în nici un fel codul său, acesta va putea gestiona şi alte tipuri de figuri ce pot fi adăugate ulterior.

2. Să considerăm ierarhia de figuri geometrice din figură. Compozitul defineste un grup de figuri, ce este tratat la rândul său ca o figură. Aşadar, o figură este fie o figură atomică (punct, cerc, pătrat etc.), fie un grup de figuri (vezi diagrama de mai jos). Conform cu această definiție, un grup de figuri poate conține la rândul său alte grupuri de figuri. Aria unui grup de figuri va fi privită ca

suma ariilor figurilor din grup, deplasarea unui grup de figuri va consta în deplasarea figurilor conținute în acel grup etc.



Scrieți un program care să îndeplinească funcțiunile unui editor grafic rudimentar: crearea tipurilor de figuri de mai sus, selectare/deselectare, stergerea lor, gruparea lor, deplasarea, zoom in/out.

3. Definiți o clasă denumită Poligon în care să existe ca date membru numărul de vârfuri (int n;) şi un tablou de elemente de tip Punct reprezentând vârfurile unui poligon (Punct varf[20];). Printre metode includeți cele pentru citire, afişare, verificarea convexității şi calculul ariei poligonului.

4. Să se realizeze un program C++ care să gestioneze comenzile de la o fabrică de componente electronice. O comandă este compusă din:

- un cod numeric format din 11 cifre
- descrierea clientului care a realizat comanda
- lista componentelor.

O componentă electronică se identifică prin: un cod numeric format din maxim 6 cifre, denumire, cantitate, costul pe unitatea de produs.

Informațiile corespunzătoare comenzilor vor fi pastrate pe disc într-un fişier XML. Aplicația va putea să creeze, modifice şi să şteargă o comandă.

Exemple de parsere scrise în C++ ce pot fi utilizate:

- Arabica http://www.jezuk.co.uk/cgi-bin/view/SAX
- Xerces-C++ http://xml.apache.org/xerces-c/
- TinyXml http://www.grinninglizard.com/tinyxml/
- simple STL XML parser http://www.codeproject.com/cpp/stlxmlparser.asp

Bibliografie

1. *Programming in C++, Rules and Recommendations*, Mats Henricson and Erik Nyquist(Swedish telecom)
http://www.doc.ic.ac.uk/lab/cplus/c++.rules/

2. *C++ Coding Standard*, Todd Hoff
http://www.possibility.com/Cpp/CppCodingStandard.html

3. *The C++ Programming Language*, B. Stroustrup, Addison-Wesley.
(Cap. 12 şi parțial Cap. 15)