**// Абстрактные классы. Чистые виртуальные функции. Непрямые базовые классы.**

#include <iostream>

#include<math.h>

**const double pi=3.1416;**

using namespace std;

**class CCont**

{

public:

//Функция для вычисления объема - без содержимого. Это определение чистой ('pure') виртуальной

// функции, на что указывает '=0'

**virtual double Volume()=0;**

void Show()

{ cout<<Volume()<<endl; }

};

**class CBottle: public CCont**

{

protected:

double m\_rc;

double m\_hc;

public:

CBottle(double rc=1.0,double hc=1.0) : m\_rc(rc), m\_hc(hc)

{ cout<<"constr CBottle "<<endl; }

virtual double Volume()

{ return pi\*pow(m\_rc,2)\*m\_hc; }

};

**class CBox: public CCont**

{

protected:

double m\_l;

double m\_w;

double m\_h;

public:

CBox(double l=1.0, double w=1.0, double h=1.0):

m\_l(l), m\_w(w), m\_h(h)

{ cout<<"constr CBox "<<endl; }

virtual double Volume()

{ return m\_l\*m\_w\*m\_h; }

};

**class CCandy: public CBox**

{

public:

CCandy(double l, double w, double h): CBox(l,w,h)

{ cout<<"constr CCandy "<<endl; }

virtual double Volume()

{ return 0.85\*m\_l\*m\_w\*m\_h; }

};

int main()

{

// Указатель на абстрактный базовый класс, инициализированный адресом объекта CBox

// Указатель на базовый класс может содержать адрес объекта любого производного класса, даже //когда несколько различных классов унаследованы от одного и того же базового класса, и потому вы можете //иметь автоматический выбор правильной виртуальной функции для всего диапазона производных классов.

CCont\* pbox=new CBox (3.0, 4.0, 5.0);

//CCont\* pbox;

// CBox box1(6.0, 7.0, 8.0 );

//pbox=&box1;

cout << " V box1 : "<<pbox->Volume() << endl;

cout << " V box1 : ";

pbox->Show();

CCandy box2(2.0,1.0,2.0);

pbox=&box2;

cout << " V candy box : ";

pbox->Show();

CBottle bot(2.0, 4.0);

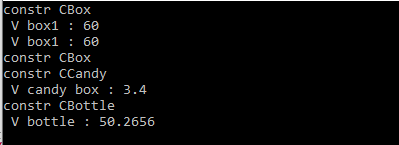
pbox=&bot;

cout << " V bottle : ";

pbox->Show();

return 0;

}



#include <iostream>

#include<math.h>

const double pi=3.1416;

using namespace std;

**class CCont**

{

public:

**virtual double Volume()=0;**

**virtual void Show()**

{ cout<<” V:”<<Volume()<<endl; }

};

**class CBottle: public CCont**

{

protected:

double m\_rc;

double m\_hc;

public:

CBottle(double rc=1.0,double hc=1.0) :m\_rc(rc),m\_hc(hc)

{ cout<<"constr CBottle "<<endl; }

**virtual double Volume()**

{ return pi\*pow(m\_rc,2)\*m\_hc; }

**virtual void Show()**

{ cout<<" V bottle :"<<Volume()<<endl; }

};

**class CBox: public CCont**

{

protected:

double m\_l;

double m\_w;

double m\_h;

public:

CBox(double l=1.0, double w=1.0, double h=1.0):

m\_l(l), m\_w(w), m\_h(h)

{ cout<<"constr CBox "<<endl; }

**virtual double Volume()**

{ return m\_l\*m\_w\*m\_h; }

};

**class CCandy: public CBox**

{

public:

CCandy(double l, double w, double h): CBox(l,w,h)

{ cout<<"constr CCandy "<<endl; }

**virtual double Volume()**

{ return 0.85\*m\_l\*m\_w\*m\_h; }

};

int main()

{

CCont\* pbox1=new CBox (3.0, 4.0, 5.0);

CBox\* pbox2=new CCandy(2.0,1.0,2.0);

CCont\* pbot=new CBottle(2.0, 4.0);

cout <<endl;

pbox1->Show();

cout <<endl;

pbox2->Show();

cout <<endl;

pbot->Show();

return 0;

}

