**//виртуальные деструкторы L11**

const double PI = 3.14159265; //Глобальное определение PI

#include <iostream>

using namespace std;

class CContainer //Общий базовый класс для специфических контейнеров

{

public:

**//Деструктор**

**~CContainer()**

**{**

**cout<<"destructor CContainer work "<< endl;**

**}**

//Функция для вычисления объема - без содержимого

//Это определение чистой ('pure') виртуальной функции, на что указывает '=0'

virtual double *Volume*() const = 0;

//Функция для отображения объема

virtual void *ShowVolume*() const

{ cout << endl

<<"Volume equals" << *Volume*() << endl;

}

};

class CBox : public CContainer //Производный класс

{

public:

//Деструктор

**~CBox()**

**{**

**cout<<"destructor CBox work "<< endl;**

**}**

//Функция для отображения объема объекта

virtual void *ShowVolume*() const

{ cout << endl

<<"Volume CBox equals" << *Volume*()<< endl;

}

//Функция для вычисления объекта CBox

virtual double *Volume*() const

{return m\_Length\*m\_Width\*m\_Height;}

//Конструктор

CBox (double lv = 1.0,double wv = 1.0, double hv = 1.0)

:m\_Length(lv), m\_Width (wv), m\_Height(hv) {}

protected:

double m\_Length;

double m\_Width;

double m\_Height;

};

class CCan : public CContainer

{

public:

**~CCan()**

**{**

**cout<<"destructor CCan work "<< endl;**

**}**

//Функция вычисления объема банки

virtual double *Volume*() const

{return 0.25\*PI\*m\_Diameter\* m\_Diameter\*m\_Height;}

//Конструктор

CCan (double hv = 4.0, double dv = 2.0)

:m\_Height(hv), m\_Diameter(dv) {}

protected:

double m\_Height;

double m\_Diameter;

};

class CGlassBox : public CBox //Производный класс

{

public:

~CGlassBox()

{

cout<<"destructor CGlassBox work "<< endl;

}

//функция производного класса для вычисления объема CGlassBox резервирующая 15% на упаковку

virtual double *Volume*()const

{ return 0.85\*m\_Length\* m\_Width\* m\_Height; }

//Конструктор

CGlassBox (double lv, double wv, double hv):CBox(lv, wv, hv) { }

};

int main(void)

{

//указатель на абстрактный базовый класс,

//инициализированный адресом объекта CBox

CContainer\* pC1 = new CBox(2.0, 3.0, 4.0);

CCan myCan(6.5, 3.0); //Определение объекта CCan

CGlassBox myGlassBox(2.0, 3.0, 4.0); //Определение объекта CGlassBox

pC1->*ShowVolume*(); //вывод объема CBox

cout <<"\*\*\*\*\*\*\*\*\*\*\*delete CBox "<< endl;

delete pC1;

pC1 = new CGlassBox(4.0, 5.0, 6.0);

pC1->*ShowVolume*();

cout <<"\*\*\*\*\*\*\*\*\*\*\*delete CGlassBox "<< endl;

delete pC1;

//инициализация адресом объекта CCan

pC1 = &myCan; //присвоить указателю адрес myCan

pC1->*ShowVolume*(); //вывод объема CCan

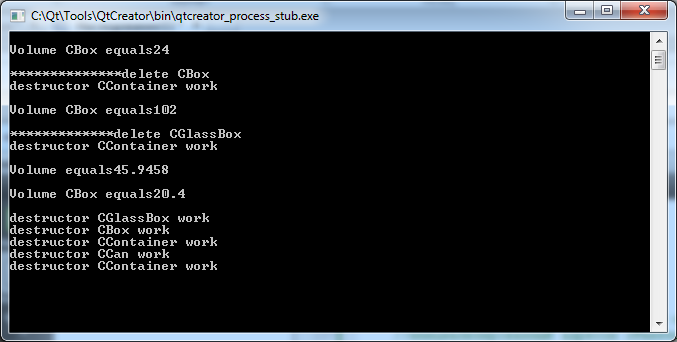
pC1 = &myGlassBox; //присвоить указателю адрес myGlassBox

pC1->*ShowVolume*(); //вывод объема CGlassBox

cout << endl;

return 0;

}

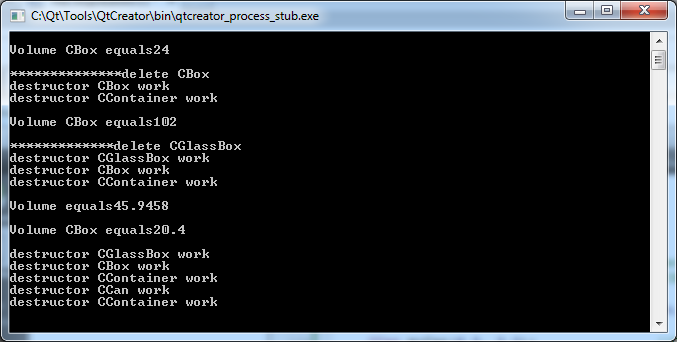


**virtual~*CContainer*()**

**{**

**cout<<"destructor CContainer work "<< endl;**

**}**



**//виртуальные деструкторы L11**

const double PI = 3.14159265; //Глобальное определение PI

#include <iostream>

using namespace std;

class CContainer //Общий базовый класс для специфических контейнеров

{

public:

**//Деструктор**

**virtual~*CContainer*()**

**{**

**cout<<"destructor CContainer work "<< endl;**

**}**

//Функция для вычисления объема - без содержимого

//Это определение чистой ('pure') виртуальной функции, на что указывает '=0'

virtual double *Volume*() const = 0;

//Функция для отображения объема

virtual void *ShowVolume*() const

{ cout << endl

<<"Volume equals" << *Volume*() << endl;

}

};

class CBox : public CContainer //Производный класс

{

public:

//Деструктор

**~*CBox*()**

**{**

**cout<<"destructor CBox work "<< endl;**

**}**

//Функция для отображения объема объекта

virtual void *ShowVolume*() const

{ cout << endl

<<"Volume CBox equals" << *Volume*()<< endl;

}

//Функция для вычисления объекта CBox

virtual double *Volume*() const

{return m\_Length\*m\_Width\*m\_Height;}

//Конструктор

CBox (double lv = 1.0,double wv = 1.0, double hv = 1.0)

:m\_Length(lv), m\_Width (wv), m\_Height(hv) {}

protected:

double m\_Length;

double m\_Width;

double m\_Height;

};

class CCan : public CContainer

{

public:

**~*CCan*()**

**{**

**cout<<"destructor CCan work "<< endl;**

**}**

//Функция вычисления объема банки

virtual double *Volume*() const

{return 0.25\*PI\*m\_Diameter\* m\_Diameter\*m\_Height;}

//Конструктор

CCan (double hv = 4.0, double dv = 2.0)

:m\_Height(hv), m\_Diameter(dv) {}

protected:

double m\_Height;

double m\_Diameter;

};

class CGlassBox : public CBox //Производный класс

{

public:

**~*CGlassBox*()**

**{**

**cout<<"destructor CGlassBox work "<< endl;**

**}**

//функция производного класса для вычисления объема CGlassBox резервирующая 15% на упаковку

virtual double *Volume*()const

{ return 0.85\*m\_Length\* m\_Width\* m\_Height; }

//Конструктор

CGlassBox (double lv, double wv, double hv):CBox(lv, wv, hv) { }

};

int main(void)

{

//указатель на абстрактный базовый класс,

//инициализированный адресом объекта CBox

CContainer\* pC1 = new CBox(2.0, 3.0, 4.0);

CCan myCan(6.5, 3.0); //Определение объекта CCan

CGlassBox myGlassBox(2.0, 3.0, 4.0); //Определение объекта CGlassBox

pC1->*ShowVolume*(); //вывод объема CBox

cout <<"delete CBox "<< endl;

delete pC1;

pC1 = new CGlassBox(4.0, 5.0, 6.0);

pC1->*ShowVolume*();

cout <<"delete CGlassBox "<< endl;

delete pC1;

//инициализация адресом объекта CCan

pC1 = &myCan; //присвоить указателю адрес myCan

pC1->*ShowVolume*(); //вывод объема CCan

pC1 = &myGlassBox; //присвоить указателю адрес myGlassBox

pC1->*ShowVolume*(); //вывод объема CGlassBox

cout << endl;

return 0;

}

**//Деструктор**

**virtual~*CContainer*()**

**{**

**cout<<"destructor CContainer work "<< endl;**

**}**

