Выполнили:

студенты группы 19ВВ2

Гусев Д.О.

Кубасов И.М.

Приняли:

Синев М.П.

Дорошенко И.Н.

Пенза 2021

ОТЧЕТ

по лабораторной работе №6

по курсу «Технологии программирования»

на тему «Виртуальные функции и абстрактные классы»

Министерство образования Российской Федерации

Пензенский государственный университет

Кафедра «Вычислительная техника»

### Цель работы

Изучение механизма виртуализации функций в языке С++. Приобретение навыки разработки и использования виртуальных функций и абстрактных классов.

### Ход работы

### Лабораторное задание

Лабораторное задание выполняется на базе лабораторной работы №5.

Реализовать абстрактный класс **Interface**. В классе объявить чистые виртуальные функции: Input и Output.

Пронаследовать класс **Base** от абстрактного класса **Interface**. В результате структура классов должна быть следующей:

* **Base** : **Interface**
* **Base** : **Derived1**
* **Base** : **Derived2**
* **Derived2** : **Derived3**

В классах **Derived1**, **Derived2** и **Derived3** реализовать виртуальные функции Input и Output.

Требования к программе:

В основной программе объявить несколько указателей типа **Interface\***. По указателям разместить (попытаться разместить) объекты всех доступных классов (**Interface**, **Base**, **Derived1**, **Derived2** и **Derived3**). Для всех размещенных объектов (используя указатель типа **Interface\***) вызвать функции Input и Output.

### Листинг:

**Phone.h**

#pragma once

#include <iostream>

#include <istream>

using namespace std;

class Interface {

public:

virtual void Input() = NULL;

virtual void Output() = NULL;

};

class Phone : public Interface

{

protected:

double m\_size;

char\* m\_model = new char[50];

char\* m\_producer = new char[50];

int m\_price;

int tmp\_size = 500;

int size = 50;

public:

Phone(Phone& Phone) {

strcpy\_s(m\_producer, 50, Phone.m\_producer);

strcpy\_s(m\_model, 50, Phone.m\_model);

m\_size = Phone.m\_size;

m\_price = Phone.m\_price;

}

Phone(Phone\* Phone) {

strcpy\_s(m\_producer, 50, Phone->m\_producer);

strcpy\_s(m\_model, 50, Phone->m\_model);

m\_size = Phone->m\_size;

m\_price = Phone->m\_price;

}

Phone()

{

strcpy(m\_producer, "HUAWEI");

strcpy(m\_model, "EVA-01");

m\_size = 18.9;

m\_price = 19990;

}

~Phone()

{

delete[] m\_producer;

delete[] m\_model;

}

void Input() override

{

char\* tmp = new char[tmp\_size];

cout << "Введите производителя телефона : " << endl;

cin.getline(tmp, tmp\_size);

int len = strlen(tmp);

while (len > size)

{

cout << "Данная строка больше максимального размера (50) введите значения заново " << endl;

cin.getline(tmp, tmp\_size);

len = strlen(tmp);

}

strcpy(m\_producer, tmp);

cout << "Введите модель телефона:" << endl;

cin.getline(tmp, tmp\_size);

len = strlen(tmp);

while (len > size)

{

cout << "Данная строка больше максимального размера (50) введите значения заново " << endl;

cin.getline(tmp, tmp\_size);

len = strlen(tmp);

}

strcpy(m\_model, tmp);

cout << "Введите размер экрана телефона: " << endl;

cin >> m\_size;

cout << "Введите цену телефона: " << endl;

(cin >> m\_price).get();

delete[] tmp;

system("cls");

}

void Output() override {

printf("Производитель: %s\n", m\_producer);

printf("Модель: %s\n", m\_model);

printf("Размер экрана: %f\n", m\_size);

printf("Цена: %d\n", m\_price);

}

};

class derived1 : public Phone {

protected:

char\* m\_country = new char[50];

public:

derived1() {

strcpy\_s(m\_country, 50, "\0");

}

derived1(const char\* country, char\* producer, char\* model, double size, int price) {

strcpy(m\_country, country);

}

derived1(derived1& d) {

strcpy(m\_country, d.m\_country);

strcpy(m\_producer, d.m\_producer);

strcpy(m\_model, d.m\_model);

m\_size = d.m\_size;

m\_price = d.m\_price;

}

void Input() override

{

char\* tmp = new char[tmp\_size];

cout << "Введите страну: " << endl;

cin.getline(tmp, tmp\_size);

int len = strlen(tmp);

while (len > 20)

{

cout << "Данная строка больше максимального размера (50) введите значения заново " << endl;

cin.getline(tmp, tmp\_size);

len = strlen(tmp);

}

strcpy(m\_country, tmp);

Phone::Input();

delete[] tmp;

}

virtual void Output() {

printf("Страна: %s\n", m\_country);

Phone::Output();

}

~derived1() {

delete[] m\_country;

}

};

class derived2 : public Phone {

protected:

int m\_year;

public:

derived2() {

m\_year = 0;

}

derived2(int year, char\* producer, char\* model, double size, int price) {

m\_year = year;

}

derived2(derived2& d) {

m\_year = d.m\_year;

strcpy(m\_producer, d.m\_producer);

strcpy(m\_model, d.m\_model);

m\_size = d.m\_size;

m\_price = d.m\_price;

}

void Input() override

{

cout << "Введите год выпуска телефона: " << endl;

(cin >> m\_year).get();

Phone::Input();

}

void Output() override

{

printf("Год: %d\n", m\_year);

Phone::Output();

}

~derived2() {

}

};

class derived3 : public derived1 {

protected:

char\* m\_city = new char[50];

public:

derived3() {

strcpy\_s(m\_city, 50, "\0");

}

derived3(char\* city, char\* country, char\* producer, char\* model, double size, int price) {

strcpy(m\_city, city);

}

derived3(derived3& d) {

strcpy(m\_city, d.m\_city);

strcpy(m\_country, d.m\_country);

strcpy(m\_producer, d.m\_producer);

strcpy(m\_model, d.m\_model);

m\_size = d.m\_size;

m\_price = d.m\_price;

}

void Input() override

{

char\* tmp = new char[tmp\_size];

cout << "Введите город: " << endl;

cin.getline(tmp, tmp\_size);

int len = strlen(tmp);

while (len > 20)

{

cout << "Данная строка больше максимального размера (50) введите значения заново " << endl;

cin.getline(tmp, tmp\_size);

len = strlen(tmp);

}

strcpy(m\_city, tmp);

derived1::Input();

delete[] tmp;

}

void Output() override {

printf("Город: %s\n", m\_city);

derived1::Output();

}

};

**Source.cpp**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <conio.h>

#include <Windows.h>

#include <locale.h>

#include <iostream>

#include "phone.h"

int main() {

setlocale(LC\_ALL, "Russian");

Interface \* p1, \* p2, \* p3, \* p4;

Phone A;

derived1 B;

derived2 C;

derived3 D;

p1 = &A;

p1->Input();

p1->Output();

p2 = &B;

p2->Input();

p2->Output();

p3 = &C;

p3->Input();

p3->Output();

p4 = &D;

p4->Input();

p4->Output();

return 0;

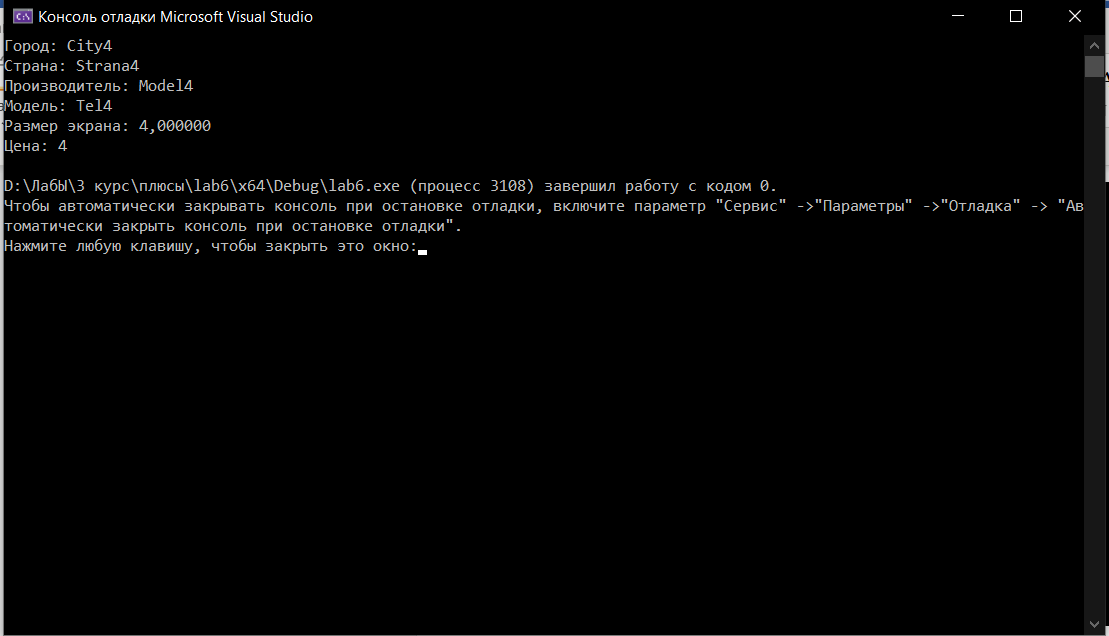
}

**Результат работы**









**Вывод:** Мы изучили механизм виртуализации функций в языке С++. Приобрели навыки разработки и использования виртуальных функций и абстрактных классов.