Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №3

За 4 семестр

По дисциплине «ООТПиСП»

Тема: «Наследование и виртуальные функции»

Выполнил: студент 2 курса

Группы ПО-4(2)

Коташевич С.Н.

Проверил: Миндер А.В.

Брест 2021

Лабораторная работа №3

Наследование и виртуальные функции

**Цель:** получить практические навыки создания иерархии классов и использования статических компонентов класса.

Вариант 13

**Перечень классов:**

**13) республика, монархия, королевство, государство.**

**Основное содержание работы:**

Написать программу, в которой создаётся иерархия классов. Включить полиморфные объекты в связный список, используя статистические компоненты класса. Показать использование виртуальных функций.

**Порядок выполнения работы:**

1. Определить иерархию классов (в соответствии с вариантом).

2. Определить в классе статическую компоненту - указатель на начало связанного списка объектов и статическую функцию для просмотра списка.

3. Реализовать классы.

4. Написать демонстрационную программу, в которой создаются объекты различных классов и помещаются в список, после чего список просматривается.

5. Сделать соответствующие методы не виртуальными и посмотреть, что будет.

6. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе (смотри пункт 6 следующего раздела).

**Методические указания:**

1. Для определения иерархии классов связать отношением наследования классы, приведенные в приложении (для заданного варианта). Из перечисленных классов выбрать один, который будет стоять во главе иерархии. Это абстрактный класс.

2. Определить в классах все необходимые конструкторы и деструктор.

3. Компонентные данные класса специфицировать как protected.

4. Пример определения статических компонентов:

static person\* begin; // указатель на начало списка

static void print(void); // просмотр списка

5. Статическую компоненту-данное инициализировать вне определения класса, в глобальной области.

6. Для добавления объекта в список предусмотреть метод класса, т.е. объект сам добавляет себя в список. Например, a.Add() − объект a добавляет себя в список.

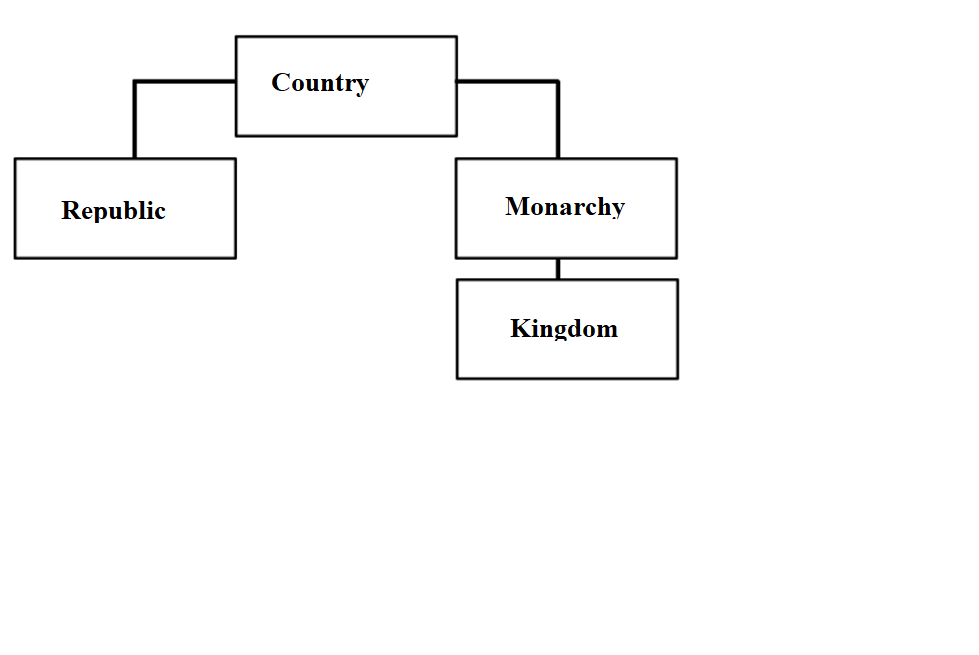
Включение объекта в список можно выполнять при создании объекта, т.е. поместить операторы включения в конструктор. В случае иерархии классов, включение объекта в список должен выполнять только конструктор базового класса. Вы должны продемонстрировать оба этих способа.

7. Список просматривать путем вызова виртуального метода Show каждого объекта.

8. Статический метод просмотра списка вызывать не через объект, а через класс.

9. Определение классов, их реализацию, демонстрационную программу поместить в отдельные файлы.

**Иерархия классов:**



**Листинг демонстрационной программы:**

**Модуль Country.hpp**

#pragma once

#include <iostream>

using namespace std;

class Country {

protected:

string \_name;

string \_capital;

Country\* next;

public:

Country();

static Country\* begin;

void Add();

virtual void Show() = 0;

static void print();

virtual ~Country();

};

**Модуль Country.cpp**

#include "Country.hpp"

Country\* Country::begin = NULL;

Country::Country() : \_name("None"), \_capital("None")

{

next = NULL;

Add();

}

void Country::Add() {

Country\* temp = this;

temp->next = begin;

begin = temp;

}

void Country::Show() {

cout << \_name << endl

<< \_capital << endl;

}

void Country::print() {

Country\* temp = begin;

while (temp != NULL)

{

temp->Show();

temp = temp->next;

}

}

Country::~Country(){}

**Модуль Monarchy.hpp**

#pragma once

#include "Country.hpp"

class Monarchy : public Country {

protected:

int \_population;

public:

Monarchy();

Monarchy(string name, string capital, int population);

void Show();

~Monarchy();

};

**Модуль Monarchy.cpp**

#include "Monarchy.hpp"

Monarchy::Monarchy() : \_population(0)

{}

Monarchy::Monarchy(string name, string capital, int population) : \_population(population)

{

\_name = name;

\_capital = capital;

}

void Monarchy::Show() {

Country::Show();

cout << \_population << endl;

}

Monarchy::~Monarchy()

{}

**Модуль Kingdom.hpp**

#pragma once

#include "Monarchy.hpp"

class Kingdom : public Monarchy {

protected:

string \_king;

public:

Kingdom();

Kingdom(string name, string capital, int population, string king);

void Show();

~Kingdom();

};

**Модуль Kingdom.cpp**

#include "Kingdom.hpp"

Kingdom::Kingdom() : \_king("None")

{}

Kingdom::Kingdom(string name, string capital, int population, string king) : Monarchy(name, capital, population), \_king(king)

{}

void Kingdom::Show() {

Monarchy::Show();

cout << \_king << endl;

}

Kingdom::~Kingdom()

{}

**Модуль Republic.hpp**

#pragma once

#include "Country.hpp"

class Republic : public Country {

protected:

double \_area;

public:

Republic();

Republic(string name, string capital, double area);

void Show();

~Republic();

};

**Модуль Republic.cpp**

#include "Republic.hpp"

Republic::Republic() : \_area(0.0)

{}

Republic::Republic(string name, string capital, double area) : \_area(area)

{

\_name = name;

\_capital = capital;

}

void Republic::Show() {

Country::Show();

cout << \_area << endl;

}

Republic::~Republic()

{}

**Модуль mian.cpp**

#include <iostream>

#include "Country.hpp"

#include "Monarchy.hpp"

#include "Kingdom.hpp"

#include "Republic.hpp"

int main()

{

system("color F0");

Republic r11("Belarus", "Minsk", 12312.12);

//r11.Add();

Republic r12("Russia", "Moskow", 356556.34534);

//r12.Add();

r11.Show();

r12.Show();

cout << endl;

Monarchy\* m1 = new Monarchy("France", "Paris", 123456);

//m1->Add();

m1->Show();

cout << endl;

Kingdom k1("Belgium", "Brussels", 23244224, "Philip");

k1.Show();

cout << endl;

Country::print();

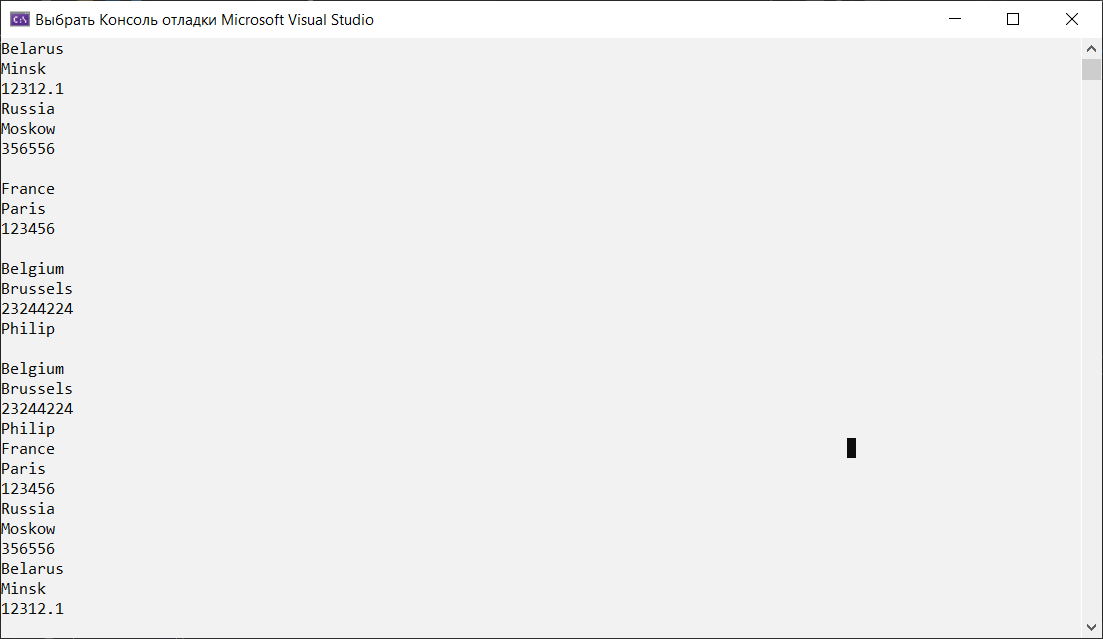
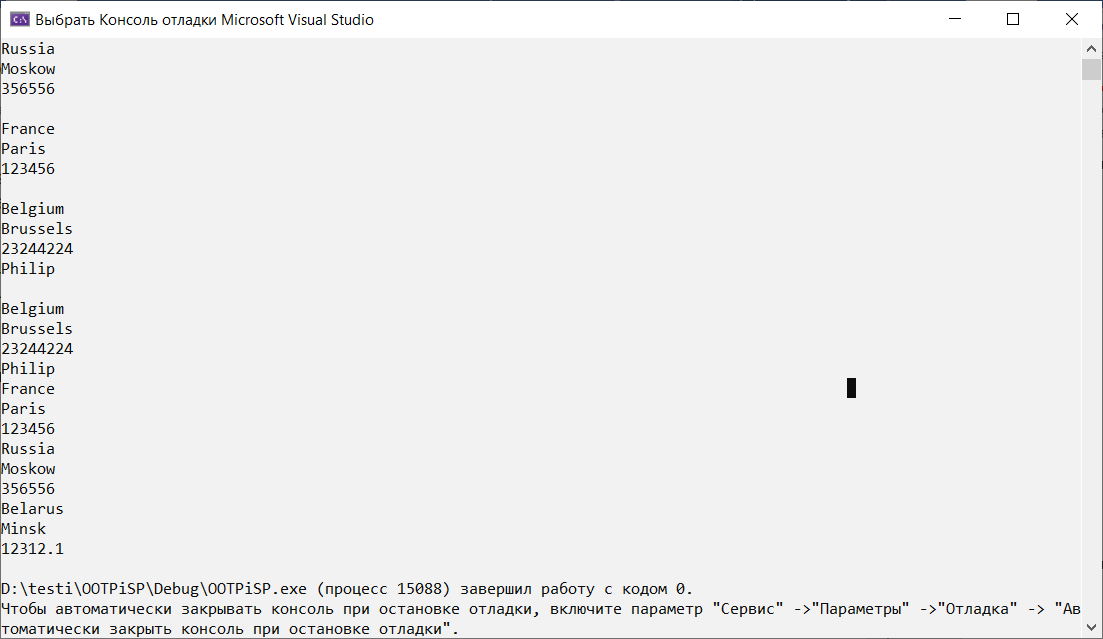
delete m1;

return 0;

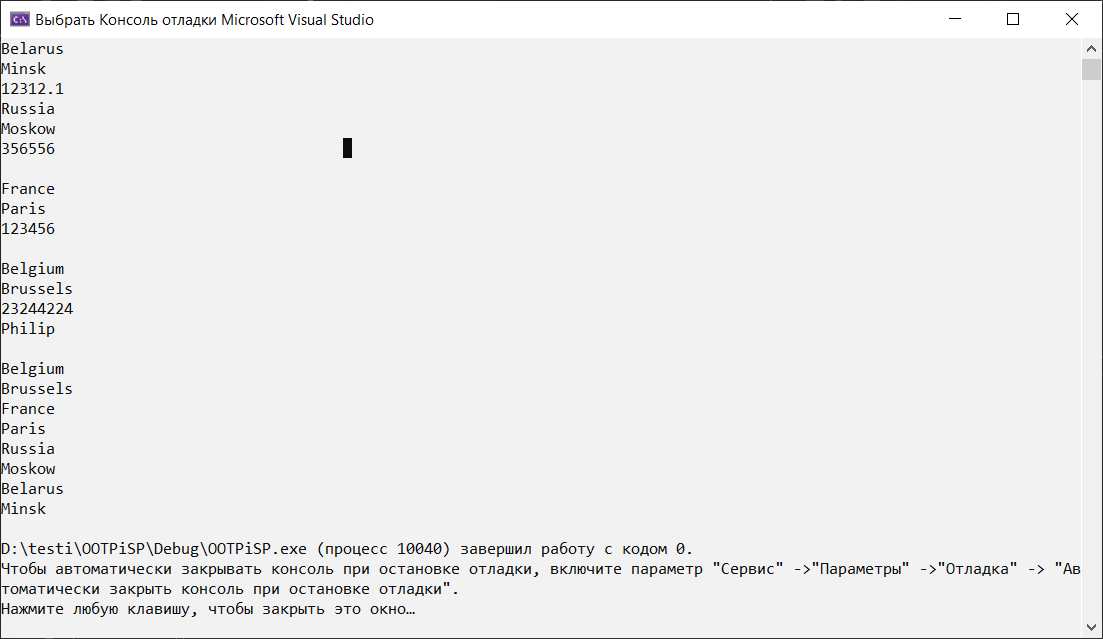
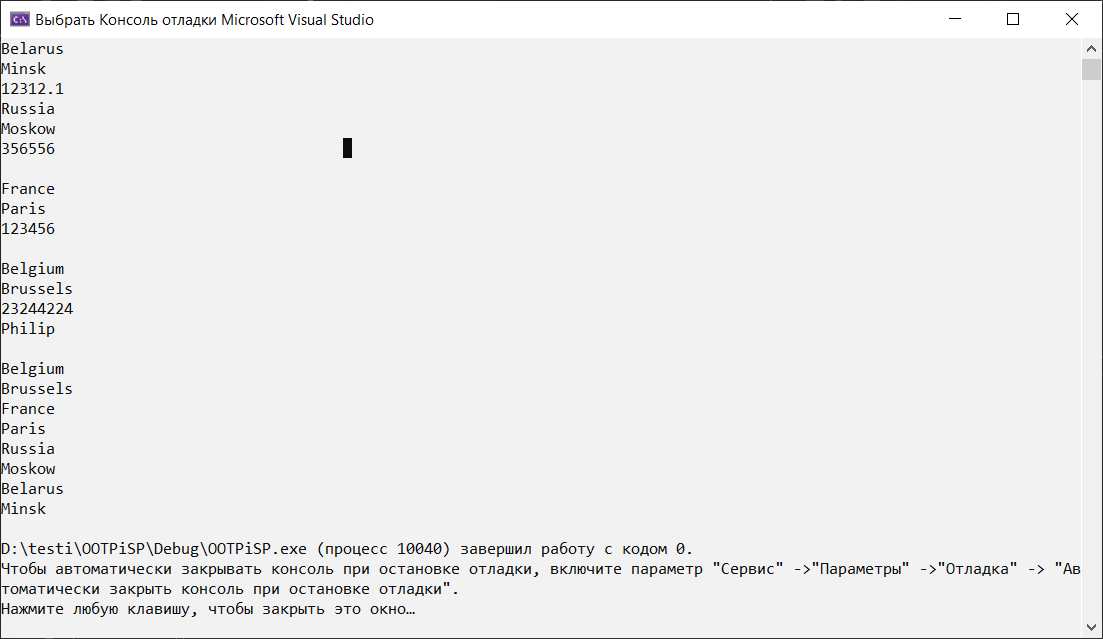
}

**Результат работы программы:**

**С виртуальными функциями:**

**Без виртуальных функций:**

**Вывод**: в ходе лабораторной работы получил практические навыки создания иерархии классов и использования статических компонентов класса.