Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Лабораторная работа №4

По дисциплине ЯП за 4 семестр  
Тема: «Наследование и виртуальные функции»

**Выполнил:**

Студент группы ПО-6(1)  
 2-го курса

Мартынович Даниил

**Проверила:**

Булей Е. В.

Брест 2022

Задание. Вариант 13

Задание:

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

Перечень классов: республика, монархия, государство, королевство.

2. Определить в классе статическую компоненту – указатель на начало связанного списка объектов и статическую функцию для просмотра списка (инициализировать вне определения класса, в глобальной области).

3. Реализовать классы. Определить в классах все необходимые конструкторы и деструктор.

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.

1) Определить иерархию классов:

Государство -> Республика

Государство -> Монархия -> Королевство

2) Определение пользовательских классов:

class State

{

protected:

char\* name;

static State\* head;

State\* next;

public:

State();

void add();

static void look\_up\_list();

virtual~State();

virtual void show() = 0;

};

class Republic : public State

{

protected:

int Size;

public:

Republic();

Republic(const char\* newname,int newSize);

~Republic();

void set(const char\* newname, int newSize);

void show();

};

class Monarch :public State {

protected:

char\* Prezident;

public:

Monarch();

Monarch(const char\* newname, const char\* newPrezident);

void set(const char\* newname, const char\* newPrezident);

void show();

~Monarch();

};

class Kingdom :public Monarch {

protected:

int num\_province;

public:

Kingdom();

Kingdom(const char\* newname, const char\* newPrezident, int num\_province);

void set(const char\* newname, const char\* newPrezident, int num\_province);

void show();

~Kingdom();

};

3) Реализация конструкторов и деструктора:

State::State()

{

name = new char[sizeof("Не указано")];

strcpy(name, "Не указано");

add();

}

State::~State(){}

Republic::Republic():State()

{

Size = 0;

}

Republic::Republic(const char\* newname, int newSize) : State(){

name = new char[sizeof(newname)];

strcpy(name, newname);

this->Size = newSize;

}

Republic::~Republic(){}

Monarch::Monarch():State()

{

Prezident = new char[sizeof("Не указано")];

strcpy(Prezident, "Не указано");

}

Monarch::Monarch(const char\* newname, const char\* newPrezident):State()

{

name = new char[sizeof(newname)];

strcpy(name, newname);

Prezident = new char[sizeof(newPrezident)];

strcpy(Prezident, newPrezident);

}

Monarch::~Monarch(){}

Kingdom::Kingdom():Monarch(){

num\_province = 0;

}

Kingdom::Kingdom(const char\* newname, const char\* newPrezident, int num\_province) : Monarch() {

name = new char[sizeof(newname)];

strcpy(name, newname);

Prezident = new char[sizeof(newPrezident)];

strcpy(Prezident, newPrezident);

this->num\_province = num\_province;

}

Kingdom::~Kingdom(){}

4) Реализация методов для добавления объекта в список:

void State::add() {

State\* p = this;

p->next = head;

head = p;

}

5) Реализация методов для просмотра списка:

void State::look\_up\_list()

{

State\* p = head;

cout << "Просмотр списка" << endl;

while (p)

{

cout << endl;

p->show();

p = p->next;

}

}

6)Код программы:

Файл class STATE.h

#pragma once

#ifndef PROGRAM

#define PROGRAM

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <iomanip>

using namespace std;

class State

{

protected:

char\* name;

static State\* head;

State\* next;

public:

State();

void add();

static void look\_up\_list();

virtual~State();

virtual void show() = 0;

};

#endif // !1

Файл realization STATE.cpp

#include "class STATE.h"

State\* State::head = NULL;

State::State()

{

name = new char[sizeof("Не указано")];

strcpy(name, "Не указано");

add();

}

void State::show()

{

cout << "вызвана виртуальная функция" << endl;

}

void State::add()

{

State\* p = this;

p->next = head;

head = p;

}

void State::look\_up\_list()

{

State\* p = head;

cout << "Просмотр списка" << endl;

while (p)

{

cout << endl;

p->show();

p = p->next;

}

}

State::~State(){}

Файл class REPUBLIC.h

#pragma once

#include "class STATE.h"

class Republic : public State

{

protected:

int Size;

public:

Republic();

Republic(const char\* newname,int newSize);

~Republic();

void set(const char\* newname, int newSize);

void show();

};

Файл realization Republic.cpp

#include "class REPUBLIC.h"

Republic::Republic():State()

{

Size = 0;

}

Republic::Republic(const char\* newname, int newSize) : State(){

name = new char[sizeof(newname)];

strcpy(name, newname);

this->Size = newSize;

}

void Republic::set(const char\* newname, int newSize)

{

name = new char[sizeof(newname)];

strcpy(name, newname);

this->Size = Size;

}

void Republic::show()

{

cout << "Название страны: " << name << endl;

cout << "Размер страны: " << Size << endl;

}

Republic::~Republic(){}

Файл class MONARCH.h

#pragma once

#include "class STATE.h"

class Monarch :public State {

protected:

char\* Prezident;

public:

Monarch();

Monarch(const char\* newname, const char\* newPrezident);

void set(const char\* newname, const char\* newPrezident);

void show();

~Monarch();

};

Файл realization MONARCH.cpp

#include "class MONARCH.h"

Monarch::Monarch():State()

{

Prezident = new char[sizeof("Не указано")];

strcpy(Prezident, "Не указано");

}

Monarch::Monarch(const char\* newname, const char\* newPrezident):State()

{

name = new char[sizeof(newname)];

strcpy(name, newname);

Prezident = new char[sizeof(newPrezident)];

strcpy(Prezident, newPrezident);

}

void Monarch::set(const char\* newname, const char\* newPrezident)

{

name = new char[sizeof(newname)];

strcpy(name, newname);

Prezident = new char[sizeof(newPrezident)];

strcpy(Prezident, newPrezident);

}

void Monarch::show() {

cout << "Название страны: " <<name<< endl;

cout << "Правитель: " <<Prezident<< endl;

}

Monarch::~Monarch(){}

Файл class KINGDOM.h

#pragma once

#include "class MONARCH.h"

class Kingdom :public Monarch {

protected:

int num\_province;

public:

Kingdom();

Kingdom(const char\* newname, const char\* newPrezident, int num\_province);

void set(const char\* newname, const char\* newPrezident, int num\_province);

void show();

~Kingdom();

};

Файл realization KINGDOM.cpp

#include "class KINGDOM.h"

Kingdom::Kingdom():Monarch(){

num\_province = 0;

}

Kingdom::Kingdom(const char\* newname, const char\* newPrezident, int num\_province) : Monarch() {

name = new char[sizeof(newname)];

strcpy(name, newname);

Prezident = new char[sizeof(newPrezident)];

strcpy(Prezident, newPrezident);

this->num\_province = num\_province;

}

void Kingdom::set(const char\* newname, const char\* newPrezident, int num\_province) {

name = new char[sizeof(newname)];

strcpy(name, newname);

Prezident = new char[sizeof(newPrezident)];

strcpy(Prezident, newPrezident);

this->num\_province = num\_province;

}

void Kingdom::show() {

cout << "Название страны: " <<name<< endl;

cout << "Правитель: " <<Prezident<< endl;

cout << "Число провинций: " <<num\_province<< endl;

}

Kingdom::~Kingdom(){}

Файл demonstration.cpp

#include "class STATE.h"

#include "class REPUBLIC.h"

#include "class MONARCH.h"

#include "class KINGDOM.h"

int main()

{

setlocale(LC\_ALL, "RU");

system("color F0");

cout << endl;

Republic b;

b.add();

b.show();

cout << endl;

Republic c("Belgium", 12);

c.add();

c.show();

cout << endl;

Monarch g("New Country", "MS Jin");

g.set("Japan", "Mr Robe");

g.add();

g.show();

cout << endl;

Kingdom d;

d.set("Brazil", "Linkaln", 12);

d.add();

d.show();

cout << endl;

system("pause");

}

Вывод: получил практические навыки создания иерархии классов и использования статических компонентов класса.