Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

за 1 семестр

По дисциплине: «Языки программирования»

Тема: «Наследование и виртуальные функции»

Выполнила:

Савицкая А.Ю., ПО-7

Проверил:

Бойко Д.О.

Брест 2021

**Цель работы:**

Получить практические навыки создания иерархии классов и использования статических компонентов класса.

**Задание:**

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).

1) Иерархия класса в виде графа:



2) Определение пользовательских классов:

class Education {

protected:

char\* person;

int room;

virtual void show() = 0;

public:

Education();

static Education\* head; //указатель на начало связанного списка объектов

Education\* next;

void add();

//статическая функция для просмотра списка

static void look\_up\_list();

virtual~Education();

};

class Exam :public Education {

protected:

int kol;

public:

Exam();

Exam(char\* person, int room, int kol);

void set(char\* person, int room, int kol);

void show();

~Exam();

};

class Final\_exam :public Exam {

protected:

int time;

public:

Final\_exam();

Final\_exam(char\* person, int room, int kol, int time);

void set(char\* person, int room, int kol, int time);

void show();

~Final\_exam();

};

class Test : public Education {

protected:

int hull;

public:

Test();

Test(char\* person, int room, int hull);

void set(char\* person, int room, int hull);

void show();

~Test();

};

3) Реализация конструкторов и деструкторов:

Education::Education() {

add();

}

Education::~Education() {}

Exam::Exam() : Education() {}

Exam::Exam(char\* person, int room, int kol) : Education() {

this->person = person;

this->room = room;

this->kol = kol;

}

Exam::~Exam() {}

Final\_exam::Final\_exam() : Exam() {}

Final\_exam::Final\_exam(char\* person, int room, int kol, int time) : Exam(person, room, kol) {

this->person = person;

this->room = room;

this->kol = kol;

this->time = time;

}

Final\_exam::~Final\_exam() {}

Test::Test() : Education() {}

Test::Test(char\* person, int room, int hull) : Education() {

this->person = person;

this->room = room;

this->hull = hull;

}

Test::~Test() {

}

4) Реализация методов для добавления объекта в список:

void Education::add() {

Education\* p = this;

p->next = head;

head = p;

}

5) Реализация методов для просмотра списка:

void Education::look\_up\_list() {

Education\* p = head;

cout << "Список: " << "\n";

while (p) {

cout << "\n";

p->show();

p = p->next;

}

}

6) Текст программы:

1. education.h

#pragma once

#include <iostream>

using namespace std;

class Education {

protected:

char\* person;

int room;

virtual void show() = 0;

public:

Education();

static Education\* head; //указатель на начало связанного списка объектов

Education\* next;

void add();

//статическая функция для просмотра списка

static void look\_up\_list();

virtual~Education();

};

2. YAP2.cpp

#include "education.h"

Education\* Education::head = NULL;

Education::Education() {

add();

}

void Education::show() {

cout << "на месте виртуальной функции вывожу это" << endl;

}

void Education::add() {

Education\* p = this;

p->next = head;

head = p;

}

void Education::look\_up\_list() {

Education\* p = head;

cout << "Список: " << "\n";

while (p) {

cout << "\n";

p->show();

p = p->next;

}

}

Education::~Education() {}

3.test.h

#pragma once

#include "education.h"

class Test : public Education {

protected:

int hull;

public:

Test();

Test(char\* person, int room, int hull);

void set(char\* person, int room, int hull);

void show();

~Test();

};

4.test\_realization.cpp

#include "test.h"

Test::Test() : Education() {}

Test::Test(char\* person, int room, int hull) : Education() {

this->person = person;

this->room = room;

this->hull = hull;

}

void Test::set(char\* person, int room, int hull) {

this->person = person;

this->room = room;

this->hull = hull;

}

void Test::show() {

cout << "Преподаватель: " << person << endl;

cout << "Аудитория: " << room << endl;

cout << "Корпус: " << hull << endl;

}

Test::~Test() {

}

5.exam.h

#pragma once

#include "education.h"

class Exam :public Education {

protected:

int kol;

public:

Exam();

Exam(char\* person, int room, int kol);

void set(char\* person, int room, int kol);

void show();

~Exam();

};

6.exam\_realization.cpp

#include "exam.h"

Exam::Exam() : Education() {}

Exam::Exam(char\* person, int room, int kol) : Education() {

this->person = person;

this->room = room;

this->kol = kol;

}

void Exam::set(char\* person, int room, int kol) {

this->person = person;

this->room = room;

this->kol = kol;

}

void Exam::show() {

cout << "Преподаватель: " << person << endl;

cout << "Аудитория: " << room << endl;

cout << "Количество должников: " << kol << endl;

}

Exam::~Exam() {}

7.final\_exam.h

#pragma once

#include "exam.h"

class Final\_exam :public Exam {

protected:

int time;

public:

Final\_exam();

Final\_exam(char\* person, int room, int kol, int time);

void set(char\* person, int room, int kol, int time);

void show();

~Final\_exam();

};

8.final\_exam\_realization.cpp

#include "final\_exam.h"

Final\_exam::Final\_exam() : Exam() {}

Final\_exam::Final\_exam(char\* person, int room, int kol, int time) : Exam(person, room, kol) {

this->person = person;

this->room = room;

this->kol = kol;

this->time = time;

}

void Final\_exam::set(char\* person, int room, int kol, int time) {

this->person = person;

this->room = room;

this->kol = kol;

this->time = time;

}

void Final\_exam::show() {

cout << "Преподаватель: " << person << endl;

cout << "Аудитория: " << room << endl;

cout << "Количество должников: " << kol << endl;

cout << "Длительность(в академических часах): " << time << endl;

}

Final\_exam::~Final\_exam() {}

9. demonastration.cpp

#include "education.h"

#include "exam.h"

#include "final\_exam.h"

#include "test.h"

#include <iostream>

using namespace std;

int main() {

setlocale(0, "");

system("color F0");

char\* person = new char[100];

cin.getline(person, 100);

Test object1(person, 422, 1);

Test p = object1;

p.add(); //объект сам добавляет себя в список

object1.show();

cout << endl;

person = new char[100];

cin.getline(person, 100);

int kol;

cin >> kol;

Exam\* object2 = new Exam(); //включение объекта в список при создании объекта

object2->set(person, 312, kol);

object2->show();

cout << endl;

person = new char[100];

cin.ignore();

cin.getline(person, 100);

int room;

cin >> room;

Final\_exam object3 = Final\_exam();

object3.set(person, room, 20, 4);

object3.show();

cout << endl;

Education::look\_up\_list();

delete object2;

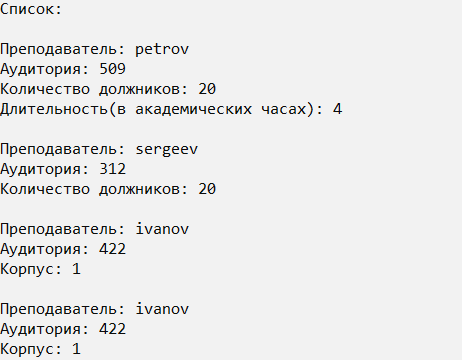
delete[] person;

system("pause");

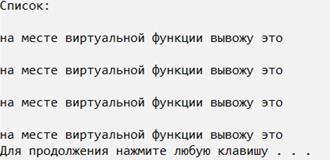
return 0;

}

Результаты тестирования программы:



- без виртуальной функции:



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