Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №4

По дисциплине: «ООТПиСП»

Тема: «Наследование и виртуальные функции»

Вариант №3

Выполнил:

Студент 2-го курса

Группы ПО-7

Комиссаров А.Е.

Проверила:

Хацкевич М.В.

Брест 2022

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

**Порядок выполнения работы**

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

2. Определить в классе статическую компоненту - указатель на нача-

ло 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.

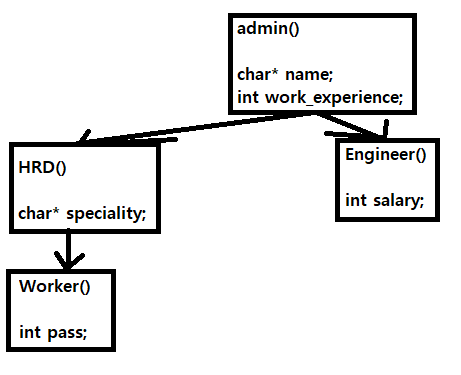
5. Сделать соответствующие методы не виртуальными и посмотреть,

что будет.

6. Реализовать вариант, когда объект добавляется в список при соз-

дании, т.е. в конструкторе

**Иерархия классов**

****

**Код программы:**

**main.cpp**

#include"class Admin.h"

#include"class Worker.h"

#include"class HRD.h"

#include"class Engineer.h"

#include <iostream>

int main() {

setlocale(LC\_ALL, "rus");

char\* name = new char[100];

cout << "Enter the name:" << endl;

cin.getline(name, 100);

Engineer object1(name, 5, 900);

Engineer p = object1;

object1.show();

cout << endl;

name = new char[100];

char\* speciality = new char[100];

cout << "Enter the name:" << endl;

cin.getline(name, 100);

cout << "Enter speciality:" << endl;

cin.getline(speciality, 100);

HRD\* object2 = new HRD(); //добавление объекта в список при его создании

object2->set(name, 3, speciality);

object2->show();

cout << endl;

name = new char[100];

speciality = new char[100];

cout << "Enter the name:" << endl;

cin.getline(name, 100);

cout << "Enter speciality:" << endl;

cin.getline(speciality, 100);

Worker object3 = Worker();

object3.set(name, 2, speciality, 3);

object3.show();

cout << endl;

Admin::look\_up\_list();

delete object2;

delete[] name;

delete[] speciality;

return 0;

}

**admin.h:**

#pragma once

#include <iostream>

using namespace std;

class Admin {

protected:

char\* name;

int work\_experience;

public:

Admin();

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

Admin\* next;

void add();

virtual void show();

static void look\_up\_list(); //статическая функция для просмотра списка

virtual~Admin();

};

**admin.cpp:**

#include "class Admin.h"

Admin\* Admin::head = NULL;

Admin::Admin() {

add();

}

void Admin::show() {

cout << "Вызов виртуальной функции" << endl;

}

void Admin::add() {

Admin\* p = this;

p->next = head;

head = p;

}

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

Admin\* p = head;

cout << "Просмотр списка" << endl;

while (p) {

cout << "==============================================================" << endl;

p->show();

p = p->next;

}

}

Admin::~Admin() {}

**worker.h:**

#include "class Admin.h"

Admin\* Admin::head = NULL;

Admin::Admin() {

add();

}

void Admin::show() {

cout << "Вызов виртуальной функции" << endl;

}

void Admin::add() {

Admin\* p = this;

p->next = head;

head = p;

}

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

Admin\* p = head;

cout << "Просмотр списка" << endl;

while (p) {

cout << "==============================================================" << endl;

p->show();

p = p->next;

}

}

Admin::~Admin() {}

**worker.cpp:**

#include"class Worker.h"

Worker::Worker() : HRD() {}

Worker::Worker(char\* name, int work\_experience, char\* speciality, int pass) : HRD(name, work\_experience, speciality) {

this->name = name;

this->work\_experience = work\_experience;

this->speciality = speciality;

this->pass = pass;

}

void Worker::set(char\* name, int work\_experience, char\* speciality, int pass) {

this->name = name;

this->work\_experience = work\_experience;

this->speciality = speciality;

this->pass = pass;

}

void Worker::show() {

cout << "Имя: " << name << endl;

cout << "Опыт работы(в годах): " << work\_experience << endl;

cout << "Специальность: " << speciality << endl;

cout << "Количество пропусков без причины: " << pass << endl;

}

Worker::~Worker() {}

**HRD.h:**

#pragma once

#include"class Admin.h"

class HRD :public Admin {

protected:

char\* speciality;

public:

HRD();

HRD(char\* name, int work\_experience, char\* speciality);

void set(char\* name, int work\_experience, char\* speciality);

void show();

~HRD();

};

**HRD.cpp:**

#include"class HRD.h"

HRD::HRD() : Admin() {}

HRD::HRD(char\* name, int work\_experience, char\* speciality) : Admin() {

this->name = name;

this->work\_experience = work\_experience;

this->speciality = speciality;

}

void HRD::set(char\* name, int work\_experience, char\* speciality) {

this->name = name;

this->work\_experience = work\_experience;

this->speciality = speciality;

}

void HRD::show() {

cout << "Имя: " << name << endl;

cout << "Опыт работы(в годах): " << work\_experience << endl;

cout << "Специальность: " << speciality << endl;

}

HRD::~HRD() {}

**Engineer.h:**

#pragma once

#include"class Admin.h"

class Engineer :public Admin {

protected:

int salary;

public:

Engineer();

Engineer(char\* name, int work\_experience, int salary);

void set(char\* name, int work\_experience, int salary);

void show();

~Engineer();

};

**Engineer.cpp:**

#include"class Engineer.h"

Engineer::Engineer() : Admin() {}

Engineer::Engineer(char\* name, int work\_experience, int salary) : Admin() {

this->name = name;

this->work\_experience = work\_experience;

this->salary = salary;

}

void Engineer::set(char\* name, int work\_experience, int salary) {

this->name = name;

this->work\_experience = work\_experience;

this->salary = salary;

}

void Engineer::show() {

cout << "Имя: " << name << endl;

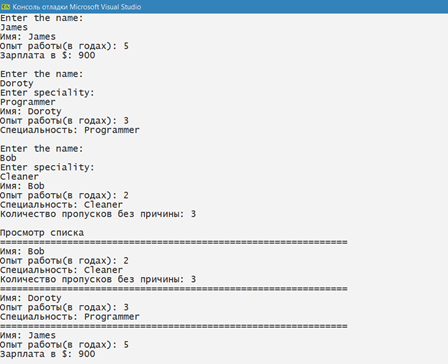
cout << "Опыт работы(в годах): " << work\_experience << endl;

cout << "Зарплата в $: " << salary << endl;

}

Engineer::~Engineer() {}

**Результат работы программы:**

****

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