Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине: “Языки программирования”

Тема: “Наследование и виртуальные функции”Вариант №2

Выполнил: студент 2 курса

группы ПО-7

Дмитрук М.А.

Проверила: Хацкевич М.В.

Брест 2022

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

Постановка задачи: написать программу, в которой создается иерархия классов. Включить полиморфные объекты в связанный список, используя статические 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
5. объекты различных классов и помещаются в список, после чего список просматривается.
6. Сделать соответствующие методы не виртуальными и посмотреть, что будет.
7. Реализовать вариант, когда объект добавляется в список при создании, т.е. в конструкторе (смотри пункт 6 следующего раздела).

Вариант задания: 2) служащий, персона, рабочий, инженер;

Иерархия классов:

class Person  
 const char \*name;  
 float passId;

class Admin: Person  
 int passLevel;

class Worker : Person  
 const char\* company;

class Engineer : Worker  
 int money;

**Код программы:**

**Содержание файла Main.CPP**

#include <iomanip>

#include "Person.h"

int Person::count = 0; Person\*\* Person::items = nullptr;

int main() {

setlocale(LC\_ALL, "Russian"); //русский язык

Worker Worker1("Игорь", 1234.593, "Precision Publishing ltd.");

Worker1.Add(); //добавление первого элемента в список, остальные будут добавлятся при создании

Engineer Eng1("Сергей", 123.1231, "Mitsubishi company", 1200); //создание инженера

Engineer FnC("Василий", 1000273.02, "Toyota", 1500); //создание инженера

Admin Admin1("Иван", 183132.3912, 5); //создание админа

Person::listItems(); //вывод элементов

}

**Содержание файла Person.CPP**

#include <iostream>

#include "Person.h"

using namespace std;

Person::Person(const char\* name, float passId, bool isAdd) :

name(name), passId(passId) {

if (isAdd) Add();

cout << "Создан Person \"" << name << "\"\n";

}

Person::~Person() { cout << "Удалён Person \"" << name << "\"\n"; }

void Person::Add() {

Person\*\* temp = items;

items = new Person \* [count + 1];

for (int i = 0; i < count; i++) items[i] = temp[i];

items[count] = this; count++;

}

void Person::listItems() {

cout << "\n=========================================\n";

for (int i = 0; i < count; i++) items[i]->Print();

cout << "=========================================\n" << endl;

}

Worker::Worker(const char\* name, float passId, const char\* company) : Person(name, passId, false), company(company) { cout << "Создан рабочий \"" << name << "\"\n"; }

Worker::~Worker() { cout << "Удалён рабочий \"" << name << "\"\n"; }

void Worker::Print() { cout << "Рабочий с именем: \"" << name << "\", пасспорт: " << passId << ", компания: " << company << "\n"; }

Engineer::Engineer(const char\* name, float passId, const char\* company, int money) : Worker(name, passId, company), money(money) { cout << "Создан инженер \"" << name << "\"\n"; }

Engineer::~Engineer() { cout << "Удален инженер \"" << name << "\"\n"; }

void Engineer::Print() { cout << "Инженер с именем: \"" << name << "\", пасспортом: " << passId << ", компания: " << company << ", зарплата: " << money << "\n"; }

Admin::Admin(const char\* name, float passId, int passLevel) : Person(name, passId, false), passLevel(passLevel) { cout << "Создан администратор \"" << name << "\"\n"; }

Admin::~Admin() { cout << "Удалён админ \"" << name << "\"\n"; }

void Admin::Print() { cout << "Админ с именем: \"" << name << "\", пасспорт: " << passId << ", уровень: " << passLevel << "\n"; }

**Содержание файла Person.H**

#ifndef PERSON\_H

#define PERSON\_H

class Person {

protected:

const char\* name;

float passId;

public:

static int count;

static Person\*\* items;

Person(const char\* name, float passId, bool isAdd = true);

~Person();

virtual void Print() = 0;

virtual void Add() final;

static void listItems();

};

class Worker : public Person {

protected: const char\* company;

public:

Worker(const char\* name, float passId, const char\* company);

~Worker();

void Print() override;

};

class Engineer : protected Worker {

protected: int money;

public:

Engineer(const char\* name, float passId, const char\* company, int money);

~Engineer();

void Print() override;

};

class Admin : Person {

protected: int passLevel;

public:

Admin(const char\* name, float passId, int passLevel);

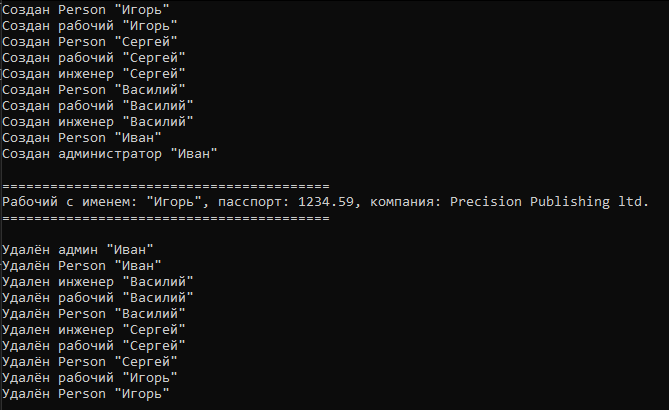
~Admin();

void Print() override;

};

#endif

**Результат работы программы:**

****

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