Министерство образования Республики Беларусь

Учреждение образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №2

За 3 семестр

По дисциплине «Языки программирования»

Тема: «Наследование и виртуальные функции»

Выполнил: студент 2 курса

Группы ПО-4(2)

Коташевич С.Н.

Проверил: Хацкевич М.В.

Брест 2020

**Лабораторная работа №2**

**Наследование и виртуальные функции**

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

**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.

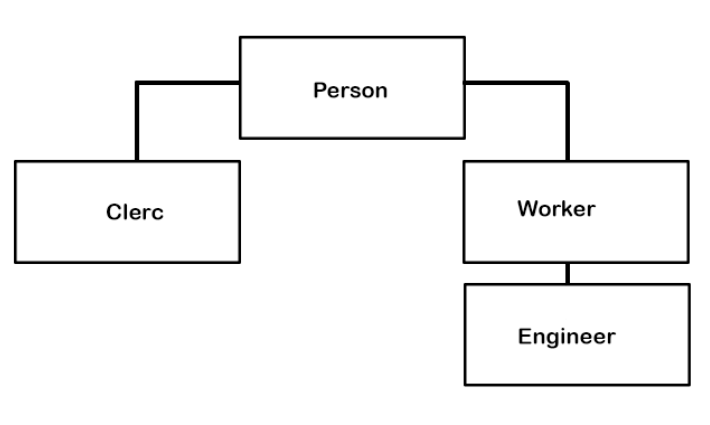
8. Статический метод просмотра списка вызывать не через объект, а через класс.

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

**Вариант 2:**

2) служащий, персона, рабочий, инженер.

**Иерархия классов:**



**Определение пользовательский классов:**

class person

{

protected:

char\* name;

char\* surname;

public:

person(); //конструктор по умолчанию

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

person\* next; //указатель на следующий экземпляр списка

virtual void show() = 0; //функция (виртуальная) просмотра содержимого экземпляров

void add\_list(); //добавление экземпляра в список

static void look\_up(); //просмотр списка

~person(); //деструктор

};

class worker :public person

{

protected:

int experience;

public:

worker(); //конструктор по умолчанию

worker(const char\* name, const char\* surname, int experience); //конструктор с параметрами

void set(const char\* name, const char\* surname, int experience); //присваивание экземпляру значений

void show(); //просмотр содержимого экземпляров

virtual ~worker(); //деструктор

};

class engineer :public worker

{

protected:

int age;

public:

engineer(); //конструктор по умолчанию

engineer(const char\* name, const char\* surname, int experience, int age); //конструктор с параметрами

void set(const char\* name, const char\* surname, int experience, int age); //присваивание экземпляру значений

void show(); //просмотр содержимого экземпляров

~engineer(); //деструктор

};

class clerc :public person

{

protected:

int time;

public:

clerc(); //конструктор по умолчанию

clerc(const char\* name,const char\* surname, int time); //конструктор с параметрами

void set(const char\* name, const char\* surname, int time); //присваивание экземпляру значений

void show(); //просмотр содержимого экземпляров

~clerc(); //деструктор

};

**Реализация конструкторов с параметрами и деструктора:**

worker::worker(const char\* name, const char\* surname, int experience)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->experience = experience;

}

engineer::engineer(const char\* name, const char\* surname, int experience, int age)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->experience = experience;

this->age = age;

}

clerc::clerc(const char\* name, const char\* surname, int time)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->time = time;

}

person::~person()

{

delete[] name;

delete[] surname;

}

**Реализация методов для добавления объектов в список:**

void person::add\_list()

{

person\* p = this;

p->next = head;

head = p;

}

**Реализация методов для просмотра списка:**

void person::look\_up()

{

person\* p = head;

cout << "Loking for the list" << endl;

while (p) {

p->show();

p = p->next;

}

}

**Листинг демонстрационной программы:**

**Модуль person.h**

#ifndef PERSON\_H

#define PERSON\_H

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

using namespace std;

class person

{

protected:

char\* name;

char\* surname;

public:

person(); //конструктор по умолчанию

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

person\* next; //указатель на следующий экземпляр списка

virtual void show() = 0; //функция (виртуальная) просмотра содержимого экземпляров

void add\_list(); //добавление экземпляра в список

static void look\_up(); //просмотр списка

virtual ~person(); //деструктор

};

#endif // !PERSON\_H

#endif // !PERSON\_H

**Модуль person.cpp**

#include "person.h"

person\* person::head = NULL;

person::person()

{

name = NULL;

surname = NULL;

next = NULL;

//add\_list();

}

void person::show()

{

cout << "Where is virtual function?" << endl;

}

void person::add\_list()

{

person\* p = this;

p->next = head;

head = p;

}

void person::look\_up()

{

person\* p = head;

cout << "Loking for the list" << endl;

while (p) {

p->show();

p = p->next;

}

}

person::~person()

{

delete[] name;

delete[] surname;

}

**Модуль worker.h**

#ifndef WORKER\_H

#define WORKER\_H

#include "person.h"

class worker :public person

{

protected:

int experience;

public:

worker(); //конструктор по умолчанию

worker(const char\* name, const char\* surname, int experience); //конструктор с параметрами

void set(const char\* name, const char\* surname, int experience); //присваивание экземпляру значений

void show(); //просмотр содержимого экземпляров

~worker(); //деструктор

};

#endif // !WORKER\_H

**Модуль worker.cpp**

#include "worker.h"

worker::worker() : person() { experience = 0; }

worker::worker(const char\* name, const char\* surname, int experience)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->experience = experience;

}

void worker::set(const char\* name, const char\* surname, int experience)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->experience = experience;

}

void worker::show()

{

cout << endl << name << endl

<< surname << endl

<< experience << endl;

}

worker::~worker() {}

**Модуль engineer.h**

#ifndef ENGINEER\_H

#define ENGINEER\_H

#include "worker.h"

class engineer :public worker

{

protected:

int age;

public:

engineer(); //конструктор по умолчанию

engineer(const char\* name, const char\* surname, int experience, int age); //конструктор с параметрами

void set(const char\* name, const char\* surname, int experience, int age); //присваивание экземпляру значений

void show(); //просмотр содержимого экземпляров

~engineer(); //деструктор

};

#endif // !ENGINEER\_H

**Модуль engineer.cpp**

#include "engineer.h"

engineer::engineer() : worker() { age = 0; }

engineer::engineer(const char\* name, const char\* surname, int experience, int age)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->experience = experience;

this->age = age;

}

void engineer::set(const char\* name, const char\* surname, int experience, int age)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->experience = experience;

this->age = age;

}

void engineer::show()

{

cout << endl << name << endl

<< surname << endl

<< experience << endl

<< age << endl;

}

engineer::~engineer() {}

**Модуль clerc.h**

#ifndef CLERC\_H

#define CLERC\_H

#include "person.h"

class clerc :public person

{

protected:

int time;

public:

clerc(); //конструктор по умолчанию

clerc(const char\* name,const char\* surname, int time); //конструктор с параметрами

void set(const char\* name, const char\* surname, int time); //присваивание экземпляру значений

void show(); //просмотр содержимого экземпляров

~clerc(); //деструктор

};

#endif // !CLERC\_H

**Модуль clerc.cpp**

#include "clerk.h"

clerc::clerc() : person() { time = 0; }

clerc::clerc(const char\* name, const char\* surname, int time)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->time = time;

}

void clerc::set(const char\* name, const char\* surname, int time)

{

this->name = new char[20];

this->surname = new char[20];

strcpy(this->name, name);

strcpy(this->surname, surname);

this->time = time;

}

void clerc::show()

{

cout << endl << name << endl

<< surname << endl

<< time << endl;

}

clerc::~clerc() {}

**Модуль mian.cpp**

#include "person.h"

#include "worker.h"

#include "engineer.h"

#include "clerk.h"

using namespace std;

int main()

{

system("color F0");

clerc cl1("Ivan", "Skinder", 2);

cl1.add\_list();

clerc cl2("Lobster", "Dopaminovich", 2);

cl2.add\_list();

cl1.show();

cl2.show();

cout << endl;

worker\* wr1 = new worker();

wr1->add\_list();

wr1->set("Rodya", "Ogorod", 2);

wr1->show();

cout << endl;

engineer en1 = engineer();

en1.add\_list();

en1.set("Vanya", "Lobster", 4, 12);

en1.show();

cout << endl;

person::look\_up();

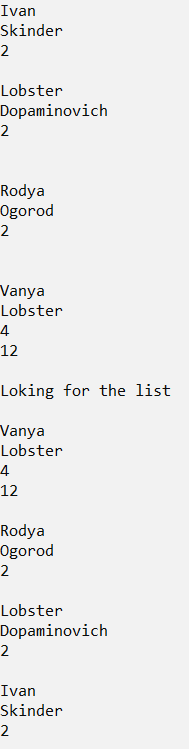
delete wr1;

return 0;

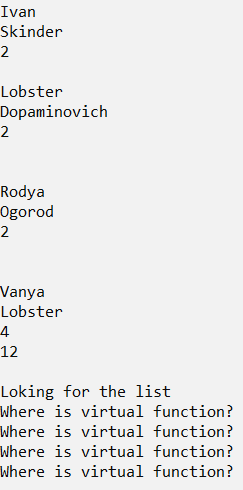
}

**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.