Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине: “Языки программирования”

Тема: “Наследование и виртуальные функции”**Вариант №3**

**Выполнил**: студент 2 курса группы ПО-7 Панкратов Роман Сергеевич

**Проверила:** Бойко Д.О.

Брест 2021

**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:**

Перечень классов: рабочий, кадры, инженер, администрация

**Код программы:**

**main.cpp**

#include <iostream>  
#include "classes.h"  
  
std::list <Worker\*> Worker::objects{};  
  
int main() {  
 auto t1 = new Cadrs("test1", "no1", 1);  
 auto t2 = new Engineer("test2", "no2", 2);  
 auto t3 = new Administration("test3", "no3", 3);  
  
 Worker::print();  
  
 delete t1;  
 delete t2;  
 delete t3;  
  
 return 0;  
}

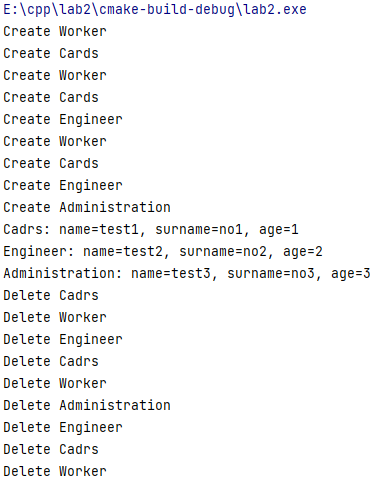
**classes.h**

#include "string"  
#include "list"  
  
class Worker {  
private:  
 static std::list <Worker\*> objects;  
  
protected:  
 std::string \_name;  
 std::string \_surname;  
 int \_age;  
  
public:  
 Worker(std::string name, std::string surname, int age);  
 virtual ~Worker();  
  
 void add();  
 static void print();  
  
 virtual void show() = 0;  
};  
  
class Cadrs: public Worker {  
public:  
 Cadrs(std::string name, std::string surname, int age);  
 ~Cadrs() override;  
  
 void show() override;  
};  
  
class Engineer: public Cadrs {  
public:  
 Engineer(std::string name, std::string surname, int age);  
 ~Engineer() override;  
  
 void show() override;  
};  
  
class Administration: Engineer {  
public:  
 Administration(std::string name, std::string surname, int age);  
 ~Administration() override;  
  
 void show() override;  
};

**classes.cpp**

#include "classes.h"  
#include "iostream"  
  
void Worker::add() {  
 objects.push\_back(this);  
}  
  
void Worker::print() {  
 for (Worker\* object : objects)  
 object->show();  
}  
  
Worker::Worker(std::string name, std::string surname, int age): \_name(name), \_surname(surname), \_age(age) {  
 add();  
 std::cout << "Create Worker" << std::endl;  
}  
  
Worker::~Worker() {  
 std::cout << "Delete Worker" << std::endl;  
}  
  
Cadrs::Cadrs(std::string name, std::string surname, int age): Worker(name, surname, age) {  
 std::cout << "Create Cards" << std::endl;  
}  
  
Cadrs::~Cadrs() {  
 std::cout << "Delete Cadrs" << std::endl;  
}  
  
void Cadrs::show() {  
 std::cout << "Cadrs: " << "name=" << \_name << ", surname=" << \_surname << ", age=" << \_age << std::endl;  
}  
  
Engineer::Engineer(std::string name, std::string surname, int age): Cadrs(name, surname, age) {  
 std::cout << "Create Engineer" << std::endl;  
}  
  
Engineer::~Engineer() {  
 std::cout << "Delete Engineer" << std::endl;  
}  
  
void Engineer::show() {  
 std::cout << "Engineer: " << "name=" << \_name << ", surname=" << \_surname << ", age=" << \_age << std::endl;  
}  
  
Administration::Administration(std::string name, std::string surname, int age): Engineer(name, surname, age) {  
 std::cout << "Create Administration" << std::endl;  
}  
  
Administration::~Administration() {  
 std::cout << "Delete Administration" << std::endl;  
}  
  
void Administration::show() {  
 std::cout << "Administration: " << "name=" << \_name << ", surname=" << \_surname << ", age=" << \_age << std::endl;  
}

**Результат программы:**

****

**Вывод:**

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