Министерство образования Республики Беларусь

Учреждения образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №5

По дисциплине: «Языки программирования»

Тема: «Наследование и виртуальные функции»

Выполнила:

Студентка 1 курса

Группа ПО-7

Фурсевич Д.С.

Проверил:

Бойко Д. О.

2021

Лабораторная работа №5

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

**Ход работы**

**Вариант 11**

**Задание:**

1. Определить иерархию классов (автомобиль, поезд, транспортное средство, экспресс).

2. Определить в классе статическую компоненту – указатель на начало связанного списка объектов и статическую функцию для просмотра списка (инициализировать вне определения класса, в глобальной области).

3. Реализовать классы. Определить в классах все необходимые конструкторы и деструктор.

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.

**1)** Иерархия класса в виде графа:

**2)** Определение пользовательских классов:

class transport\_means { // класс транспортное средство

protected:

char\* type; //тип

int weight; // вес

public:

transport\_means();

void add();

virtual void show() = 0; // чистая виртуальная функция

static transport\_means\* head; //указатель на начало связного списка

transport\_means\* next;

static void show\_list(); //для просмотр списка

~transport\_means();

};

class train : public transport\_means { //класс поезд

protected:

char\* kind; //

public:

train();

train(char\* type, int weight, char\* kind);

void train\_set(char\* type, int weight, char\* kind);

void show() override;

~train();

};

class car : public transport\_means { // класс автомобиль

protected:

char\* mark;

public:

car();

car(char\* type, int weight, char\* mark);

void car\_set(char\* type, int weight, char\* mark);

void show() override;

~car();

};

class express : public train {// класс экспресс

protected:

char\* otkuda;

char\* kuda;

public:

express();

express(char\* type, int weight, char\* kind, char\* otkuda, char\* kuda);

void expres\_set(char\* type, int weight, char\* kind, char\* otkuda, char\* kuda);

void show() override;

~express();

};

**3)** Реализация конструкторов и деструктора:

transport\_means::transport\_means()

{

add();

}

transport\_means ::~transport\_means() {}

train::train() : transport\_means() {}

train::train(char\* type, int weight, char\* kind) {

this->type = type;

this->weight = weight;

this->kind = kind;

}

train::~train() {}

car::car() : transport\_means() {}

car::car(char\* type, int weight, char\* mark) {

this->type = type;

this->weight = weight;

this->mark = mark;

}

car::~car() {}

express::express() :train() {}

express::express(char\* type, int weight, char\* kind, char\* otkuda, char\* kuda) {

this->type = type;

this->weight = weight;

this->kind = kind;

this->otkuda = otkuda;

this->kuda = kuda;

}

express::~express() {}

**4)** Реализация методов для добавления объекта в список:

void transport\_means::add()

{

transport\_means\* new\_tr = this;

new\_tr->next = head;

head = new\_tr;

}

**5)** Реализация методов для просмотра списка:

void transport\_means::show\_list() { //для просмотр списка

transport\_means\* new\_tr = head;

cout << "Список: " << endl;

while (new\_tr) {

new\_tr->show();

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_"<<endl;

new\_tr = new\_tr->next;

}

}

**6)** Листинг демонстрационной программы:

Модуль **transport\_means.h:**

#pragma once

#include <iostream>

#include <Windows.h>

using namespace std;

class transport\_means { // класс транспортное средство

protected:

char\* type; //тип

int weight; // вес

public:

transport\_means();

void add();

virtual void show() = 0; // чистая виртуальная функция

static transport\_means\* head; //указатель на начало связного списка

transport\_means\* next;

static void show\_list(); //для просмотр списка

~transport\_means();

};

Модуль **transport\_means.cpp:**

#include "transport\_means.h"

transport\_means\* transport\_means::head = 0;

transport\_means::transport\_means()

{

add();

}

void transport\_means::add()

{

transport\_means\* new\_tr = this;

new\_tr->next = head;

head = new\_tr;

}

void transport\_means::show() {}

void transport\_means::show\_list() { //для просмотр списка

transport\_means\* new\_tr = head;

cout << "Список: " << endl;

while (new\_tr) {

new\_tr->show();

cout << "\_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_"<<endl;

new\_tr = new\_tr->next;

}

}

transport\_means ::~transport\_means() {}

Модуль **car.h:**

#pragma once

#include "transport\_means.h"

class car : public transport\_means { // класс автомобиль

protected:

char\* mark;

public:

car();

car(char\* type, int weight, char\* mark);

void car\_set(char\* type, int weight, char\* mark);

void show() override;

~car();

};

Модуль **car.cpp:**

#include "car.h"

car::car() : transport\_means() {}

car::car(char\* type, int weight, char\* mark) {

this->type = type;

this->weight = weight;

this->mark = mark;

}

void car::car\_set(char\* type, int weight, char\* mark) {

this->type = type;

this->weight = weight;

this->mark = mark;

}

void car::show() {

cout << "Тип: " << type << endl;

cout << "Вес: " << weight << endl;

cout << "Марка: " << mark << endl;

}

car::~car() {}

Модуль **train.h:**

#pragma once

#include "transport\_means.h"

class train : public transport\_means { //класс поезд

protected:

char\* kind; //

public:

train();

train(char\* type, int weight, char\* kind);

void train\_set(char\* type, int weight, char\* kind);

void show() override;

~train();

};

Модуль **train.cpp:**

#include "train.h"

train::train() : transport\_means() {}

train::train(char\* type, int weight, char\* kind) {

this->type = type;

this->weight = weight;

this->kind = kind;

}

void train::train\_set(char\* type, int weight, char\* kind) {

this->type = type;

this->weight = weight;

this->kind = kind;

}

void train::show() {

cout << "Тип: " << type << endl;

cout << "Вес: " << weight << endl;

cout << "Вид: " << kind << endl;

}

train::~train() {}

Модуль **express.h:**

#pragma once

#include "train.h"

class express : public train {// класс экспресс

protected:

char\* otkuda;

char\* kuda;

public:

express();

express(char\* type, int weight, char\* kind, char\* otkuda, char\* kuda);

void expres\_set(char\* type, int weight, char\* kind, char\* otkuda, char\* kuda);

void show() override;

~express();

};

Модуль **express.cpp:**

#include "express .h"

express::express() :train() {}

express::express(char\* type, int weight, char\* kind, char\* otkuda, char\* kuda) {

this->type = type;

this->weight = weight;

this->kind = kind;

this->otkuda = otkuda;

this->kuda = kuda;

}

void express::expres\_set(char\* type, int weight, char\* kind, char\* otkuda, char\* kuda) {

this->type = type;

this->weight = weight;

this->kind = kind;

this->otkuda = otkuda;

this->kuda = kuda;

}

void express::show() {

cout << "Тип: " << type << endl;

cout << "Вес: " << weight << endl;

cout << "Вид: " << kind << endl;

cout << "Откуда отправляется: " << otkuda << endl;

cout << "Куда прибывает: " << kuda << endl;

}

express::~express() {}

Модуль **main.cpp:**

#include "transport\_means.h"

#include "car.h"

#include "train.h"

#include "express .h"

int main()

{

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

char\* type = new char[100];

cin.getline(type, 100);

char\* mark = new char[100];

cin.getline(mark, 100);

car element\_1(type, 35, mark);

car n = element\_1;

n.add(); //элемент сам себя добавляет в список

element\_1.show();

cout << endl;

type = new char[100];

cin.getline(type, 100);

char\* kind = new char[100];

cin.getline(kind, 100);

train \*element\_2 = new train(); //включение объекта в список при создании объекта

element\_2->train\_set(type, 40, kind);

element\_2->show();

cout << endl;

type = new char[100];

cin.getline(type, 100);

kind = new char[100];

cin.getline(kind, 100);

char\* otkuda = new char[100];

cin.getline(otkuda, 100);

char\* kuda = new char[100];

cin.getline(kuda, 100);

express element\_3 = express();

element\_3.expres\_set(type, 40, kind, otkuda, kuda);

element\_3.show();

cout << endl;

transport\_means::show\_list();

delete[] type;

delete[] kind;

delete[] mark;

delete[] otkuda;

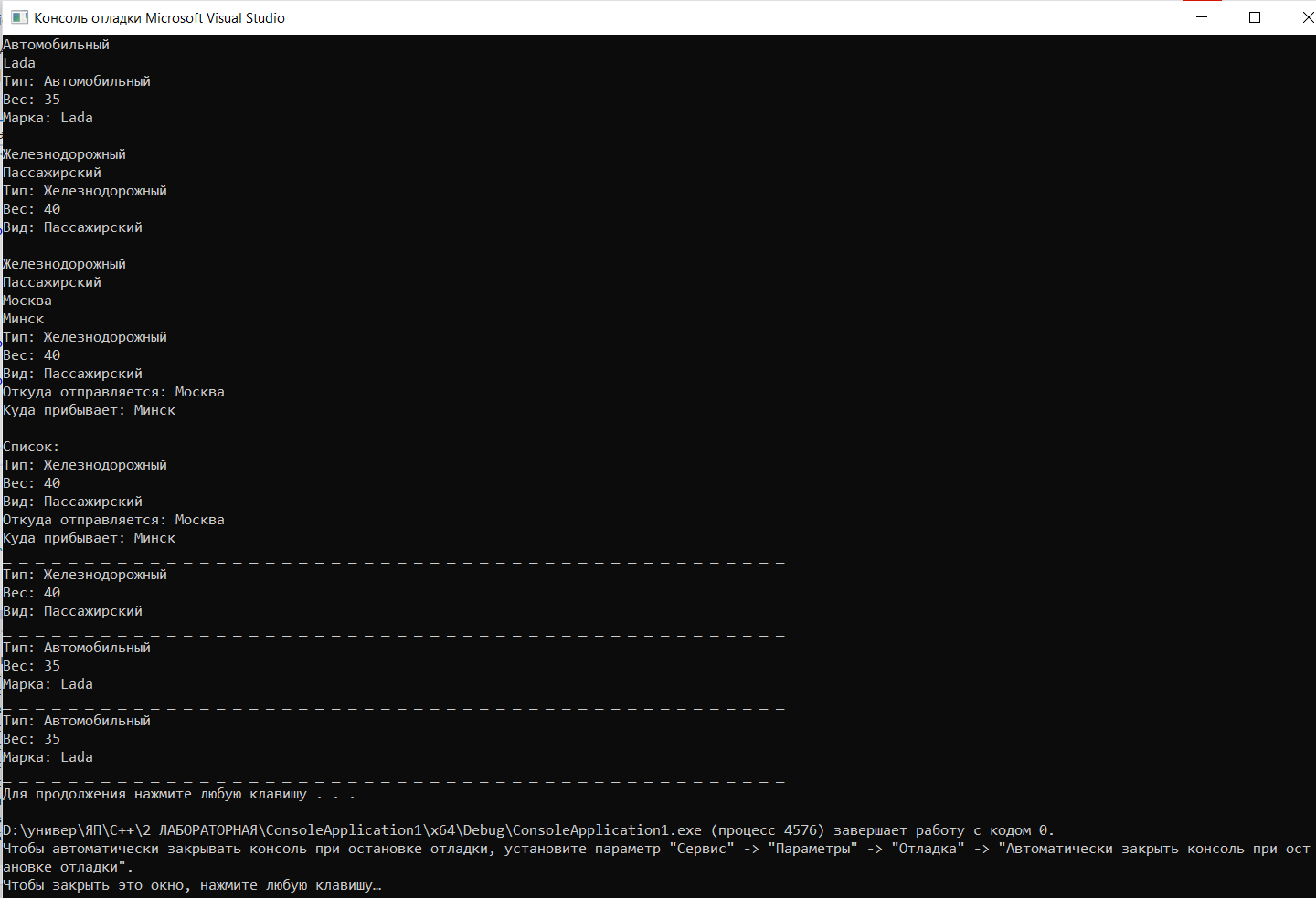
delete[] kuda;

system("pause");

return 0;

}

**Результат выполнения:**



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