Министерство образования Республики Беларусь

УО «Брестский государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №2**

По дисциплине: “Языки программирования”

Тема: “Наследование и виртуальные функции”**Вариант №11**

**Выполнил**: студент 2 курса группы ПО-7

**Лобан К.Ю.**

**Проверила:** Дряпко А. В.

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:

11) автомобиль, поезд, транспортное средство, экспресс

**Иерархия классов:**

**class** Vehicle  
 int speed;

**class** Train : Vehicle  
const char\* direction;

**class** Auto : Vehicle  
const char\* marka;

**class** Express: Train  
int maxSpeed;

**Код программы:**

**Vehicle.h**

#ifndef VEHICLE\_H

#define VEHICLE\_H

class Vehicle

{

protected:

int speed;

public:

static Vehicle\*\* items;

static int count;

Vehicle();

Vehicle(int speed, bool isAdd );

Vehicle(Vehicle& vehicle);

~Vehicle();

static void ShowItems();

virtual void AddItem() final;

virtual void Show() = 0;

};

class Auto : public Vehicle

{

protected:

const char\* marka;

public:

Auto();

Auto(int speed, const char\* marka, bool isAdd);

Auto(Auto& \_auto);

~Auto();

void Show() override;

};

class Train : public Vehicle

{

protected:

const char\* direction;

public:

Train();

Train(int speed, const char\* direction, bool isAdd);

Train(Train& train);

~Train();

void Show() override;

};

class Express : public Train

{

int maxSpeed;

public:

Express();

Express(int speed, const char\* direction, int maxSpeed, bool isAdd);

Express(Express& express);

~Express();

void Show() override;

};

#endif //VEHICLE\_H

**Auto.cpp**

#include "Vehicle.h"

#include <iostream>

using namespace std;

Auto::Auto()

{

speed = 40;

marka = "Audi";

cout << "Created Vehicle (Auto())" << endl;

}

Auto::Auto(int speed, const char\* marka, bool isAdd = false):Vehicle(speed, isAdd) {

this->marka = marka;

cout << "Created Auto (Auto(int, const char\*, bool))" << endl;

}

Auto::Auto(Auto& \_auto) {

speed = \_auto.speed;

marka = \_auto.marka;

cout << "Created Auto (Auto(&))" << endl;

}

Auto::~Auto() {

cout << "Deleted Auto" << endl;

}

void Auto::Show() {

cout << "Auto. Marka:" << marka << ".Speed:" << speed << endl;

}

**Express.cpp**

#include "Vehicle.h"

#include <iostream>

using namespace std;

Express::Express():Train()

{

maxSpeed = 190;

cout << "Created Express (Express())" << endl;

}

Express::Express(int speed, const char\* direction, int maxSpeed, bool isAdd = false): Train(speed, direction, isAdd) {

this->maxSpeed = maxSpeed;

cout << "Created Express (Express(int, const char\*,int, bool))" << endl;

}

Express::Express(Express& express) {

speed = express.speed;

maxSpeed = express.maxSpeed;

direction = express.direction;

cout << "Created Express (Express(Express&))" << endl;

}

Express::~Express() {

cout << "Deleted Express" << endl;

}

void Express::Show() {

cout << "Express. Direction:" << direction << ".Speed:" << speed <<".Max speed" << maxSpeed << endl;

}

**Train.cpp**

#include "Vehicle.h"

#include <iostream>

using namespace std;

Train::Train():Vehicle()

{

direction = "Minsk";

cout << "Created Train (Train())" << endl;

}

Train::Train(int speed, const char\* direction, bool isAdd = false):Vehicle(speed, isAdd) {

this->direction = direction;

cout << "Created Train (Train(int, const char\*, bool))" << endl;

}

Train::Train(Train& train) {

speed = train.speed;

direction = train.direction;

cout << "Created Train (Train(Train&))" << endl;

}

Train::~Train() {

cout << "Deleted Train" << endl;

}

void Train::Show() {

cout << "Train. Direction:" << direction << ".Speed:" << speed << endl;

}

**Vehicle.cpp**

Vehicle::Vehicle(Vehicle& vehicle) {

speed = vehicle.speed;

cout << "Created Vehicle (Vehicle(Vehicle&))" << endl;

}

Vehicle::~Vehicle() {

cout << "Deleted Vehicle" << endl;

}

void Vehicle::AddItem()

{

++Vehicle::count;

Vehicle\*\* tempItems = items;

items = new Vehicle \* [count];

for (int i = 0; i < count - 1;i++) {

items[i] = tempItems[i];

}

items[count - 1] = this;

delete tempItems;

}

void Vehicle::ShowItems() {

cout << "Items:" << endl;

for (int i = 0; i < count; i++) {

items[i]->Show();

}

}

**Source.cpp**

#include <iostream>

#include "Vehicle.h"

using namespace std;

Vehicle\*\* Vehicle::items = nullptr;

int Vehicle::count = 0;

int main() {

Vehicle::ShowItems();

cout << "Create auto" << endl;

Auto audi(50, "audi", true);

audi.Show();

Vehicle::ShowItems();

cout << "Create train" << endl;

Train train(130, "Brest", false);

Vehicle::ShowItems();

cout << "Add train" << endl;

train.AddItem();

Vehicle::ShowItems();

cout << "Create express" << endl;

Express trainExpress;

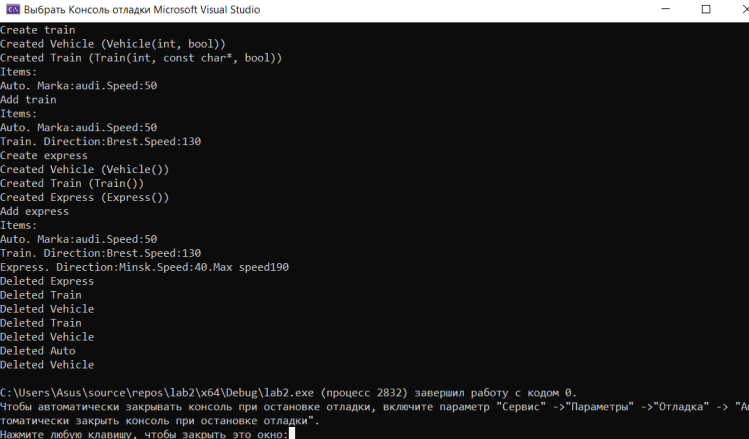
cout << "Add express" << endl;

trainExpress.AddItem();

Vehicle::ShowItems();

}

**Результат программы:**



**Вывод:**

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