3 ЛАБОРАТОРНАЯ РАБОТА №3

«ИССЛЕДОВАНИЕ МЕХАНИЗМА МНОЖЕСТВЕННОГО НАСЛЕДОВАНИЯ»

3.1 Цель работы

Приобретение практических навыков при написании объектно-ориентированных программ с использованием механизма множественного наследования.

3.2 Вариант задания – 8

Требуется описать интерфейс ввода-вывода. Описать иерархию классов. Базовые классы: Машина (марка, мощность, скорость), Транспорт (тоннаж, тип). Класс-наследник: Грузовик. Для каждого класса описать конструкторы и деструктор, функции ввода и вывода значений полей. В каждом классе должны присутствовать минимум одно уникальное поле и один уникальный метод. Проиллюстрировать корректную работу механизма множественного наследования — для этого создать объекты базовых классов и заполнить их поля данными, вывести на печать. Создать объект класса-наследника, его поля заполнить значениями соответствующих полей базовых классов. Вывести на печать данные полученного объекта.

3.3 Ход работы

3.3.1 В программе были созданы 2 базовых класса “Car” и “Vehicle”. Далее создан класс “Truck”, наследующий поля и методы базовых классов. В функции main проведены действия с объектами этих трёх классов.

3.3.2 Написана программа на C++ согласно вышеописанного алгоритма.

```
#include <iostream>
#include <windows.h>
#include <string>

#define UI unsigned int

using namespace std;

class Car {
    string brand;
```

```
    UI power;
    UI speed;
public:
    Car() { power = speed = 0; brand = "0"; cout << "Конструктор Car по умолч." <<
endl; }
    Car(string _brand, UI _power, UI _speed) { brand = _brand; power = _power; speed
= _speed; cout << "Конструктор Car" << endl; }
    ~Car() { cout << "Деструктор Car" << endl; }

    void set_car(string _brand, UI _power, UI _speed) { brand = _brand; power =
_power; speed = _speed; }
    void show_car() { cout << "Марка - " << brand << ", мощность - " << power << ",
скорость - " << speed << endl; }
};

class Vehicle {
    UI tonnage;
    string type;
public:
    Vehicle() { tonnage = 0; type = "0"; cout << "Конструктор Vehicle по умолч." <<
endl; }
    Vehicle(UI _tonnage, string _type) { tonnage = _tonnage; type = _type; cout <<
"Конструктор Vehicle" << endl; }
    ~Vehicle() { cout << "Деструктор Vehicle" << endl; }

    void set_vehicle(UI _tonnage, string _type) { tonnage = _tonnage; type = _type; }
    void show_vehicle() { cout << "Тип - " << type << ", тоннаж - " << tonnage <<
endl; }
};

class Truck : public Car, public Vehicle {
public:
    Truck() : Car(), Vehicle() { cout << "Конструктор Truck по умолч." << endl; }
    Truck(string _brand, UI _power, UI _speed, UI _tonnage, string _type) :
Car(_brand, _power, _speed), Vehicle(_tonnage, _type) { cout << "Конструктор Truck"
<< endl; }
    ~Truck() { cout << "Деструктор Truck" << endl; }

    void set_truck(string _brand, UI _power, UI _speed, UI _tonnage, string _type) {
set_car(_brand, _power, _speed); set_vehicle(_tonnage, _type); }
    void show_truck() { show_car(); show_vehicle(); }
    UI sum_parameters(UI _power, UI _speed, UI _tonnage);
};

UI Truck::sum_parameters(UI _power, UI _speed, UI _tonnage){
   return _power+_speed+_tonnage;
```

```
}

int main() {
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    Car *car_obj = new Car;
    car_obj->show_car();
    string _brand = "Mercedes-Benz";
    UI _power = 8000;
    UI _speed = 234;
    car_obj->set_car(_brand, _power, _speed);
    car_obj->show_car();
    car_obj->set_car("КамАЗ", 8600, 100);
    car_obj->show_car();
    delete car_obj;
    cout << endl;

    UI _tonnage = 12;
    string _type = "средне-грузоподъёмный";
    Vehicle *vehicle_obj = new Vehicle(_tonnage, _type);
    vehicle_obj->show_vehicle();
    delete vehicle_obj;
    cout << endl;

    Truck *truck_obj = new Truck("Hyundai", 15600, 80, 40, "высоко-грузоподъёмный");
    truck_obj->show_truck();
    cout << "Параметры на данный момент: " << endl;
    cout << "power: " << _power << ", speed: " << _speed << ", tonnage: " << _tonnage
<< endl;
    cout << "Сумма параметров: " << truck_obj->sum_parameters(_power, _speed,
_tonnage) << endl << endl;
    _brand = "Volvo FMX";
    _power = 14000;
    _speed = 120;
    _tonnage = 32;
    _type = "высоко-грузоподъёмный";
    truck_obj->set_truck(_brand, _power, _speed, _tonnage, _type);
    truck_obj->show_truck();
    delete truck_obj;
    cout << endl;

   system("pause");
    return 0;
}
```

3.3.3 Выполнена отладка программы.

Результаты тестирования отображены на рисунке 3.1. На изображении видно как успешно выполняется программа.

```
Конструктор Car по умолч.
Марка - 0, мощность - 0, скорость - 0
Марка - Mercedes-Benz, мощность - 8000, скорость - 234
Марка - КамАЗ, мощность - 8600, скорость - 100
Деструктор Car

Конструктор Vehicle
Тип - средне-грузоподъёмный, тоннаж - 12
Деструктор Vehicle

Конструктор Car
Конструктор Vehicle
Конструктор Truck
Марка - Hyundai, мощность - 15600, скорость - 80
Тип - высоко-грузоподъёмный, тоннаж - 40
Параметры на данный момент:
power: 8000, speed: 234, tonnage: 12
Сумма параметров: 8246

Марка - Volvo FMX, мощность - 14000, скорость - 120
Тип - высоко-грузоподъёмный, тоннаж - 32
Деструктор Truck
Деструктор Vehicle
Деструктор Car

Для продолжения нажмите любую клавишу . . .
```

Рисунок 3.1 – Выполнение программы

Результаты тестирования полностью соответствуют ожиданиям.

Выводы

В ходе выполнения данной лабораторной работы были получены навыки разработки программ, использующих множественное наследование. Были закреплены навыки разработки и отладки программ, использующих классы и объекты. Полученные во время разработки навыки помогут разрабатывать более сложные программы с использованием классов и объектов, более эффективные по времени выполнения алгоритмы.