**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 С++ 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.

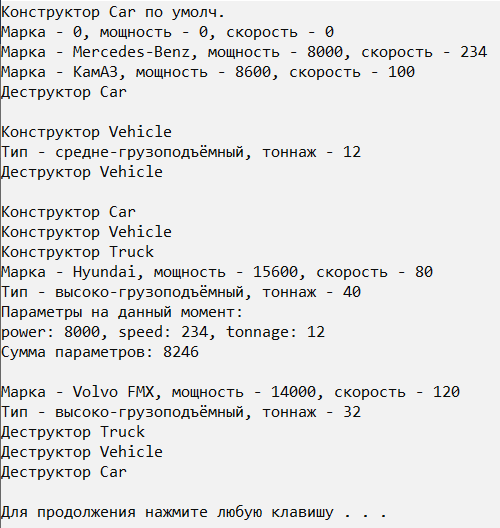


Рисунок 3.1 – Выполнение программы

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.