МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра автоматизированных систем управления

Отчет по лабораторной работе №6.

По ОБЪЕКТО-ОРИЕНТИРОВАННОМУ ПРОГРАММИРОВАНИЮ

«Сериализация в xml файл»

Выполнила:

студентка гр. ПМ-21-2 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Бутусова В.М.

Проверил:

доц., к.п.н. кафедры АСУ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кургасов В. В.

Липецк 2022

**Цель работы:** научиться использовать механизм сериализации в формат xml для сохранения данных в структурированном виде.

**Задание кафедры:** реализовать на языке C++ ведение справочников для заданной предметной области: создание, удаление, редактирование объектов. Необходимо реализовать возможность сохранения в файл / загрузку из файла данных с использованием механизма сериализации (использовать формат xml). Реализовать иерархию классов минимум из трех уровней с множественным наследованием. Операции сериализации осуществляются с классом, агрегирующим данные.

**Вариант №3:** Предметная область: автомобили.

Код программы:

#include <iostream>

#include <Windows.h>

#include <string>

#include <fstream>

#include <boost\archive\xml\_iarchive.hpp> // библиотека для работы с xml фаилами (ввод)

#include <boost\archive\xml\_oarchive.hpp> // библиотека для работы с xml фаилами (вывод)

using namespace std;

class Machine // главный наследуемый класс - машина

{

public:

string type; // тип авто

string batch; // серия

int number; // номер

string type\_gasoline; // тип бензина

void add\_machine()// функция ввода

{

cout << "Введите серию автомобиля: ";

cin >> batch;

cout << endl << "Введите номер автомобиля: ";

cin >> number;

cout << endl << "Введите тип бензина: ";

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, type\_gasoline);

cout << endl;

}

void show\_machine()// функция вывода

{

cout << "Тип машины: " << type << endl;

cout << "Cерия автомобиля: " << batch << endl;

cout << "Номер автомобиля: " << number << endl;

cout << "Тип бензина: " << type\_gasoline << endl;

}

void change\_machine()// функция изменения

{

cout << "Что вы хотите изменить" << endl;

cout << "1.Cерию автомобиля"<< endl;

cout << "2.Номер автомобиля"<< endl;

cout << "3.Тип бензина" << endl;

}

};

class Transport : public Machine // класс - транспорт, наследуется от класса - машина

{

public:

int count\_wheel; // количество колёс

string season; // сезон

string deterioration\_wheel; // состояние колёс

int p = 0;

pair <string, string> W = make\_pair("Зима", "С шипами");

pair <string, string> S = make\_pair("Лето", "Без шипов");

void add\_transport()// функция ввода

{

add\_machine();

cout << "Введите количество колёс: ";

cin >> count\_wheel;

cout << endl << "Какой сейчас сезон:";

while (p != 1 && p != 2)

{

cout << endl << "1. Зима";

cout << endl << "2. Лето" << endl;

cin >> p;

}

if (p == 1) season = "Зима";

else season = "Лето";

p = 0;

cout << endl << "Введите тип шин: ";

while (p != 1 && p != 2)

{

cout << endl << "1. С шипами";

cout << endl << "2. Без шипов" << endl;

cin >> p;

}

if (p == 1) deterioration\_wheel = "С шипами";

else deterioration\_wheel = "Без шипов";

if (season == W.first && deterioration\_wheel != W.second)

cout << endl << "Советуем заменить шины колёс, на дорогах может заносить";

else if (season == S.first && deterioration\_wheel != S.second)

cout << endl << "Советуем заменить шины колёс, на дорогах уже давно растаял лёд";

cout << endl;

}

void show\_transport()// функция вывода

{

show\_machine();

cout << "Количество колёс: " << count\_wheel << endl;

cout << "Тип колёс: " << deterioration\_wheel << endl;

cout << "Сезон года: " << season << endl;

}

void change\_transport()// функция изменения

{

change\_machine();

cout << "4.Количество колёс"<< endl;

cout << "5.Тип колёс"<< endl;

cout << "6.Сезон года" << endl;

}

};

class Automobile : public Transport // класс - автомобиль, наследуется от класса - транспорт

{

public:

int seating\_capacity; // свободные места

void add\_autho()// функция ввода

{

add\_transport();

cout << "Введите количество пассажирских мест: ";

cin >> seating\_capacity;

cout << endl;

type = "Автомобиль";

}

void show\_autho()// функция вывода

{

show\_transport();

cout << "Количество пассажирских мест: " << seating\_capacity << endl;

}

void change\_auto()// функция изменения

{

change\_transport();

cout << "8.Количество пассажирских мест: " << endl;

}

// функция сериализации

template<class Archive>

void serialize(Archive& ar, const unsigned int version)

{

ar& BOOST\_SERIALIZATION\_NVP(type);

ar& BOOST\_SERIALIZATION\_NVP(batch);

ar& BOOST\_SERIALIZATION\_NVP(number);

ar& BOOST\_SERIALIZATION\_NVP(type\_gasoline);

ar& BOOST\_SERIALIZATION\_NVP(count\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(deterioration\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(season);

ar& BOOST\_SERIALIZATION\_NVP(seating\_capacity);

}

};

class Autospectechnics : public Transport // класс - автоспецтехника, наследуется от класса - транспорт

{

public:

int tonnage; // тоннаж

void add\_spec()// функция ввода

{

add\_transport();

cout << "Введите тоннаж: ";

cin >> tonnage;

cout << endl;

}

void show\_spec()// функция вывода

{

show\_transport();

cout << "Тоннаж: " << tonnage << endl;

}

void change\_spec()// функция изменения

{

change\_transport();

cout << "7.Тоннаж " << endl;

}

};

class Bus : public Automobile // класс - автобус, наследуется от класса - автомобиль

{

public:

string place\_of\_departure; // место отбытия

string arrival\_point; // место прибытия

void add\_bus() // функция ввода

{

add\_autho();

cout << "Место отбытия: ";

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin,place\_of\_departure);

cout << endl << "Место прибытия: ";

cin.clear();

getline(cin,arrival\_point);

cout << endl;

type = "Автобус";

}

template<class Archive>// функция сериализации

void serialize(Archive& ar, const unsigned int version)

{

ar& BOOST\_SERIALIZATION\_NVP(type);

ar& BOOST\_SERIALIZATION\_NVP(batch);

ar& BOOST\_SERIALIZATION\_NVP(number);

ar& BOOST\_SERIALIZATION\_NVP(type\_gasoline);

ar& BOOST\_SERIALIZATION\_NVP(count\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(deterioration\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(season);

ar& BOOST\_SERIALIZATION\_NVP(seating\_capacity);

ar& BOOST\_SERIALIZATION\_NVP(place\_of\_departure);

ar& BOOST\_SERIALIZATION\_NVP(arrival\_point);

}

void show\_bus()// функция вывода

{

show\_autho();

cout << "Место отбытия: " << place\_of\_departure << endl;

cout << "Место прибытия: " << arrival\_point << endl;

}

void change\_bus()// функция изменения

{

change\_auto();

cout << "8.Место отбытия" << endl;

cout << "9.Место прибытия" << endl;

}

};

class Truck : public Autospectechnics // класс - грузовик, наследуется от класса - автоспецтехника

{

public:

string cargo\_type; // тип груза

void add\_truck() // функция ввода

{

add\_spec();

cout << "Введите тип груза: ";

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, cargo\_type);

cout << endl;

type = "Грузовик";

}

template<class Archive>

void serialize(Archive& ar, const unsigned int version)

{

ar& BOOST\_SERIALIZATION\_NVP(type);

ar& BOOST\_SERIALIZATION\_NVP(batch);

ar& BOOST\_SERIALIZATION\_NVP(number);

ar& BOOST\_SERIALIZATION\_NVP(type\_gasoline);

ar& BOOST\_SERIALIZATION\_NVP(count\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(deterioration\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(season);

ar& BOOST\_SERIALIZATION\_NVP(tonnage);

ar& BOOST\_SERIALIZATION\_NVP(cargo\_type);

}

void show\_truck()// функция вывода

{

show\_spec();

cout << "Перевозимый груз: " << cargo\_type << endl;

}

void change\_truck()// функция изменения данных

{

change\_spec();

cout << "8.Перевозимый груз " << endl;

}

};

class Concrete\_Mixer\_truck : public Autospectechnics // класс - бетономешательная машина, наследуется от класса - автоспецтехника

{

public:

string amount\_of\_concrete; // вместимость миксера

void add\_mix() // функция ввода

{

add\_spec();

cout << "Введите вместимость миксера: ";

cin >> amount\_of\_concrete;

cout << endl;

type = "Бетономешалка";

}

template<class Archive> // функция сериализации

void serialize(Archive& ar, const unsigned int version)

{

ar& BOOST\_SERIALIZATION\_NVP(type);

ar& BOOST\_SERIALIZATION\_NVP(batch);

ar& BOOST\_SERIALIZATION\_NVP(number);

ar& BOOST\_SERIALIZATION\_NVP(type\_gasoline);

ar& BOOST\_SERIALIZATION\_NVP(count\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(deterioration\_wheel);

ar& BOOST\_SERIALIZATION\_NVP(season);

ar& BOOST\_SERIALIZATION\_NVP(tonnage);

ar& BOOST\_SERIALIZATION\_NVP(amount\_of\_concrete);

}

void show\_mix() // функция вывода

{

show\_spec();

cout << "Объем миксера: " << amount\_of\_concrete << endl;

}

void change\_mix() // функция изменения данных

{

change\_spec();

cout << "8.Объем миксера" << endl;

}

};

class Serialize // класс для сериализации данных

{

public:

int constant = 0;

friend class boost::serialization::access; // для работы с boost

Concrete\_Mixer\_truck mixer; // объект - бетономешалка

Truck truck; // объект - грузовик

Bus bus; // объект - автобус

Automobile autho; // объект - автомобиль

string Typ; // объект - заголовок

template<class Archive> // шаблонная функция для сериализации данных

void serialize(Archive& ar, const unsigned int version)

{

if (Typ == "Автомобиль" || autho.type == "Автомобиль")

{

ar& BOOST\_SERIALIZATION\_NVP(autho);

constant = 1;

}

else if (Typ == "Автобус" || bus.type == "Автобус")

{

ar& BOOST\_SERIALIZATION\_NVP(bus);

constant = 2;

}

else if (Typ == "Грузовик" || truck.type == "Грузовик")

{

ar& BOOST\_SERIALIZATION\_NVP(truck);

constant = 3;

}

else if (Typ == "Бетономешалка" || mixer.type == "Бетономешалка")

{

ar& BOOST\_SERIALIZATION\_NVP(mixer);

constant = 4;

}

}

void menu\_add() // функция меню для доьбавления новых данных

{

char m = 0;

while (m != '5')

{

cout << "\tКакой тип машины в вашем распоряжении:" << endl;

cout << "1. Автомобиль" << endl;

cout << "2. Автобус" << endl;

cout << "3. Грузовик" << endl;

cout << "4. Бетономешательная машина" << endl;

cout << "5. Вернуться" << endl;

cin >> m;

if (m == '1')

{

autho.add\_autho(); // вызов функции добавления данных об автомобиле

constant = 1;

Typ = autho.type;

break;

}

if (m == '2')

{

bus.add\_bus(); // вызов функции добавления данных об автобусе

constant = 2;

Typ = bus.type;

break;

}

if (m == '3')

{

truck.add\_truck(); // вызов функции добавления данных об грузовике

constant = 3;

Typ = truck.type;

break;

}

if (m == '4')

{

mixer.add\_mix(); // вызов функции добавления данных об бетономешалке

constant = 4;

Typ = mixer.type;

break;

}

}

}

void menu\_change() // функция редактирования данных

{

int x;

if (Typ == "")

{

cout << "Ваш гараж пуст" << endl;

return;

}

else if (Typ == "Автомобиль") // тип данных - автомобиль

{

autho.change\_auto(); // вызов функции изменения данных автомобиля

cin >> x;

cout << "Введите: ";

switch (x)

{

case 1:

cin >> autho.batch;

break;

case 2:

cin >> autho.number;

break;

case 3:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, autho.type\_gasoline);

break;

case 4:

cin >> autho.count\_wheel;

break;

case 5:

cin >> autho.deterioration\_wheel;

break;

case 6:

cin >> autho.season;

break;

case 7:

cin >> autho.seating\_capacity;

break;

}

}

else if (Typ == "Грузовик")

{

truck.change\_truck(); // вызов функции изменения данных грузовика

cin >> x;

cout << "Введите: ";

switch (x)

{

case 1:

cin >> truck.batch;

break;

case 2:

cin >> truck.number;

break;

case 3:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, truck.type\_gasoline);

break;

case 4:

cin >> truck.count\_wheel;

break;

case 5:

cin >> truck.deterioration\_wheel;

break;

case 6:

cin >> truck.season;

break;

case 7:

cin >> truck.tonnage;

break;

case 8:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, truck.cargo\_type);

break;

}

}

else if (Typ == "Автобус")

{

bus.change\_bus();// вызов функции изменения данных автобуса

cin >> x;

cout << "Введите: ";

switch (x)

{

case 1:

cin >> bus.batch;

break;

case 2:

cin >> bus.number;

break;

case 3:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, bus.type\_gasoline);

break;

case 4:

cin >> bus.count\_wheel;

break;

case 5:

cin >> bus.deterioration\_wheel;

break;

case 6:

cin >> bus.season;

break;

case 7:

cin >> bus.seating\_capacity;

break;

case 8:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, bus.place\_of\_departure);

break;

case 9:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, bus.arrival\_point);

break;

}

}

else if (Typ == "Бетономешалка")

{

mixer.change\_mix();// вызов функции изменения данных бетономешалки

cin >> x;

cout << "Введите: ";

switch (x)

{

case 1:

cin >> mixer.batch;

break;

case 2:

cin >> mixer.number;

break;

case 3:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, mixer.type\_gasoline);

break;

case 4:

cin >> mixer.count\_wheel;

break;

case 5:

cin >> mixer.deterioration\_wheel;

break;

case 6:

cin >> mixer.season;

break;

case 7:

cin >> mixer.tonnage;

break;

case 8:

cin.clear();

cin.ignore(INT\_MAX, '\n');

getline(cin, mixer.amount\_of\_concrete);

break;

}

}

}

void show() // функция вывода данных

{

if (constant == 0)

{

cout << "Данных нет" << endl;

}

else if (constant == 1)

{

autho.show\_autho(); // вывод данных автомобиля

}

else if (constant == 2)

{

bus.show\_bus(); // вывод данных автобуса

}

else if (constant == 3)

{

truck.show\_truck(); // вывод данных грузовика

}

else if (constant == 4)

{

mixer.show\_mix(); // вывод данных бетономешательной машины

}

}

};

class serT // класс для сериализации заголовка данных

{

public:

string Type;

template<class Archive>

void serialize(Archive& ar, const unsigned int version)

{

ar& BOOST\_SERIALIZATION\_NVP(Type);

}

};

// шаблонные функции сериализации

template<typename T> //сохранение заголовка(типа) данных

void save\_type(T data) {

ofstream fout("saveT.xml");

boost::archive::xml\_oarchive oa(fout);

oa << BOOST\_SERIALIZATION\_NVP(data);

}

template<typename T> // загрузка заголовка(типа) данных

void load\_type(T& data) {

ifstream fin("saveT.xml");

boost::archive::xml\_iarchive ia(fin);

ia& BOOST\_SERIALIZATION\_NVP(data);

}

template<typename T> //сохранение данных

void save(T data) {

ofstream fout("save.xml");

boost::archive::xml\_oarchive oa(fout);

oa << BOOST\_SERIALIZATION\_NVP(data);

cout << "Успешно сохранено!" << endl;

}

template<typename T> // загрузка данных

void load(T& data) {

ifstream fin("save.xml");

boost::archive::xml\_iarchive ia(fin);

ia& BOOST\_SERIALIZATION\_NVP(data);

cout << "Успешно загружено!" << endl;

}

int main()

{

setlocale(LC\_ALL, "ru");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

Serialize \_xml; // создание объекта данных для сериализации

serT typ; // создание объекта заголовка данных для сериализации

char k = 0;

while (k != '6')

{

cout << "\tГараж" << endl;

cout << "1. Посмотреть данные" << endl;

cout << "2. Загрузить данные" << endl;

cout << "3. Сохранить данные" << endl;

cout << "4. Добавить данные" << endl;

cout << "5. Редактировать данные" << endl;

cout << "6. Выйти из гаража" << endl;

cin >> k;

if (k == '1')

{

\_xml.show(); // вызов функции вывода данных на экран

}

else if (k == '2')

{

load\_type(typ); // вызов шаблонной функции загрузки заголовка данных

\_xml.Typ = typ.Type;

load(\_xml); // вызов шаблонной функции загрузки данных

}

else if (k == '3')

{

typ.Type = \_xml.Typ; // вызов шаблонной функции сохранения заголовка данных

save\_type(typ);

save(\_xml);// вызов шаблонной функции сохранения данных

}

else if (k == '4')

{

\_xml.menu\_add(); // вызов функции меню для добавления новых данных

}

else if (k == '5')

{

\_xml.menu\_change(); // вызов функции меню для редактирования данных

}

}

}

**Пример работы программы.**

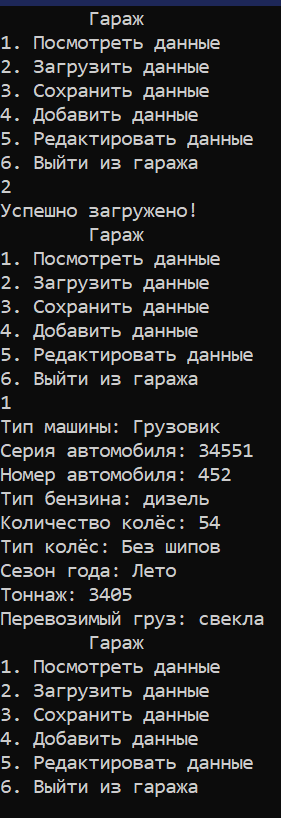


Рисунок 1 – демонстрация загрузки файла.

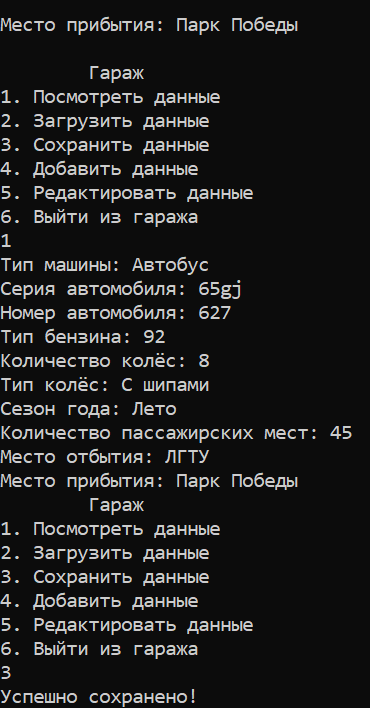
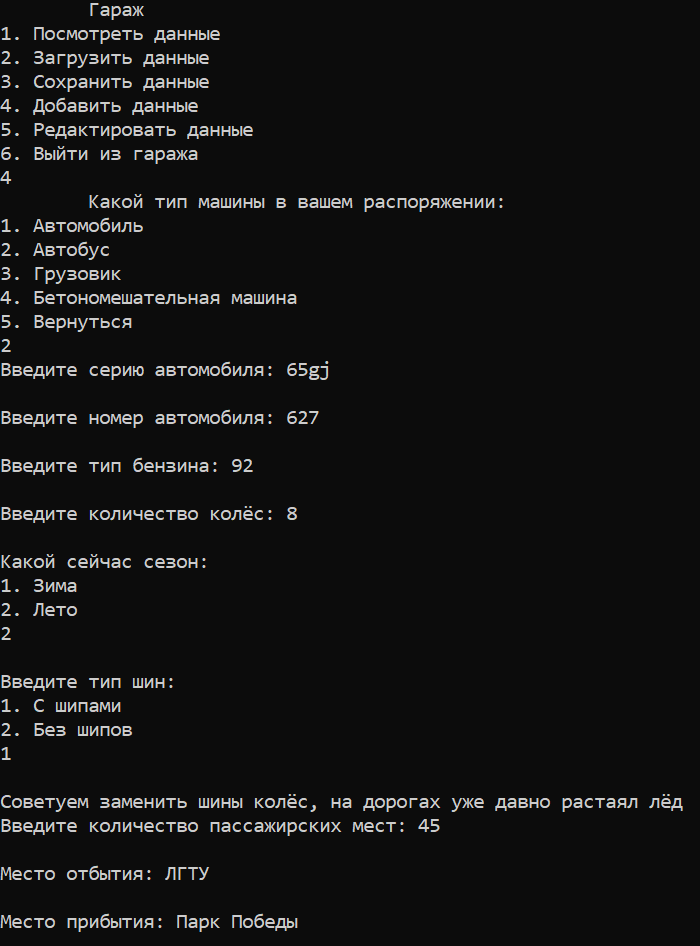


Рисунок 2-3 – демонстрация сохранения файла.

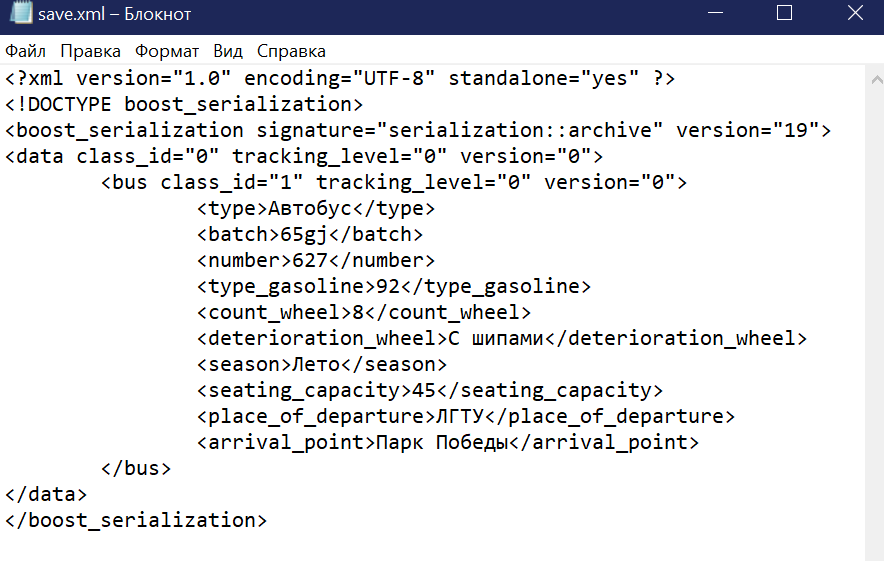


Рисунок 4 – xml файл.

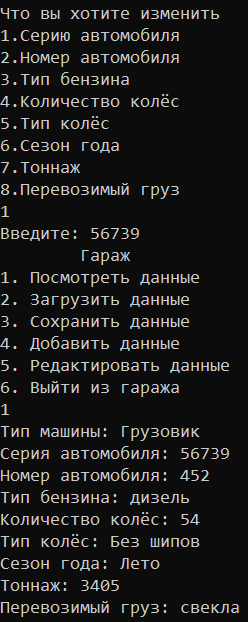


Рисунок 5 – демонстрация редактирования данных.

**Вывод:** в данной лабораторной работе я научилась использовать механизм сериализации в формат xml для сохранения данных в структурированном виде.

**Контрольные вопросы.**

**1. Что такое иерархия классов?**

Иерархия классов - это структура многочисленных связанных классов, определяющая, какие классы наследуют функции от других классов.

**2. Что представляет собой xml файл?**

Это текстовый документ, содержащий описание объектов, процессов, различных данных при помощи расширяемого языка разметки XML. Этот язык имеет иерархическую структуру, удобную для описания различных типов данных и документов. Такое описание удобно читать человеку и анализировать компьютеру.

**3. Что такое сериализация? Для каких целей она может быть использована?**

Сериализация – процесс перевода структуры данных в последовательность байтов.

Сериализация используется для передачи объектов по сети и для сохранения их в файлы. Например, нужно создать распределённое приложение, разные части которого должны обмениваться данными со сложной структурой.

**4. Для чего нужно множественное наследование?**

Множественное наследование позволяет классу перенимать функциональность у множества других классов.

**5. Какой порядок вызовов конструкторов и деструкторов при множественном наследовании?**

Конструкторы вызываются по очереди от самого дальнего предка до самого последнего наследника. Деструкторы класса наоборот, вызываются от самого раннего наследника до самого дальнего предка.