МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №14

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

доц., канд. техн. наук		К.А. Курицын
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ (ПРОЕКТУ)

РАЗРАБОТКА ПРОГРАММНОГО ПРОДУКТА «СИСТЕМА ЗАПРОСОВ К БАЗЕ ДАННЫХ»

по дисциплине: ТЕХНОЛОГИЯ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 1841		М.С.Собко
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2020

Содержание

2.Постановка задачи:

Вариант 4.4:

Создать класс «Легковая машина». При создании легковой машины клиент может получить: ВАЗ, Kia, Nissan, Toyota. Машины обладают характеристиками: цвет, тип двигателя, объем двигателя, габариты, год выпуска, число дверей, модель, марка шин, объем багажника. ВАЗ отличается наличием возможности установки багажника на крышу. Kia отличается возможностью установки подогрева зеркал заднего вида. Nissan отличается установкой подогрева сидений. Toyota отличается установкой АКПП. Использовать паттерн «Строитель». Результат создания машины и их характеристики записываются в отдельные выходные файлы «vaz», «kia», «nissan», «toyota». Пользователь может загрузить все данные о ранее созданных машинах из файлов.

3. Введение

3.1 Цели работы:

- Реализовать программу на языке C++, который решает поставленную задачу;
- Отладка и тестирование полученного программного продукта;
- Реализовать программу используя паттерн “Строитель”
- Работа с Github

3.2 Задачи:

- применить на практике знания языка C++, стандартных библиотек;
- разработать и применить методы, обеспечивающие надежность программного продукта на этапе исполнения, возможность расширения функционала, удобство взаимодействие стороннего пользователя;
- методы, примененные при разработке и реализации проекта направлены на обеспечение надежного функционирования программы и максимальное взаимодействие пользователя с программой.
- В основном содержании работы изложена информация о проектировании и подходам к решению поставленной задачи и конкретизацию полученных и примененных на практике методов, обосновании и описании выбранных подходов, и полученных результатах проведенной работы.

3.3Структура работы:

- Формализация задачи – изменение формулировки поставленной задачи в формальный вид, содержащий конкретные понятия и четко выделенные задачи. Необходима для того, чтобы выдвигать дальнейшие гипотезы относительно реализации решения.
- Проектирование – выдвижение гипотез о решении задачи, принятие наиболее подходящей и описание видов исходных данных, структурирование данных и алгоритмы их обработки, выбор технологий, подходящих для реализации, а также описание ожидаемого поведения.
- Реализация проекта – написание текста программных модулей и сценария работы для дальнейшей отладки и выпуска.
- Тестирование и отладка – взаимосвязанные и итеративные процессы имитации пользовательских сценариев взаимодействия с готовым программным продуктом, выявление допущенных ошибок и внесение исправлений, необходимых для соответствия поставленным целям и задачам.
- Выпуск – финальная версия продукта представляется как результат работы.

3.4 Функциональные требования:

- добавление новой модели автомобиля ВАЗ, Nissan, Toyota, Kia;
- независимые характеристики автомобиля.
- задать особености для каждой марки автомобиля
- удаление автомобиля из базы данных;
- создание новой базы данных
- Загрузка базы данных из файлов
- Защита от не верного выбора
- Просмотр базы данных

3.5 Требования к стороннему ПО:

Требования к оборудованию:
Процессор с тактовой частотой не ниже 1,8 ГГц. Рекомендуется использовать как минимум двухъядерный процессор.
2 ГБ ОЗУ; рекомендуется 8 ГБ ОЗУ (минимум 2,5 ГБ при выполнении на виртуальной машине)
Место на жестком диске: до 210 ГБ (минимум 800 МБ) свободного места в зависимости от установленных компонентов; обычно для установки требуется от 20 до 50 ГБ свободного места.
Скорость жесткого диска: для повышения производительности установите Windows и Visual Studio на твердотельный накопитель (SSD)
Видеоадаптер с минимальным разрешением 720p (1280 на 720 пикселей); для оптимальной работы Visual Studio рекомендуется разрешение WXGA (1366 на 768 пикселей) или более высокое.

4 Основная часть:

4.1. Формализация:

Паттерн «Строитель» -это порождающий паттерн проектирования, который позволяет создавать сложные объекты пошагово. Строитель даёт возможность использовать один и тот же код строительства для получения разных представлений объектов.

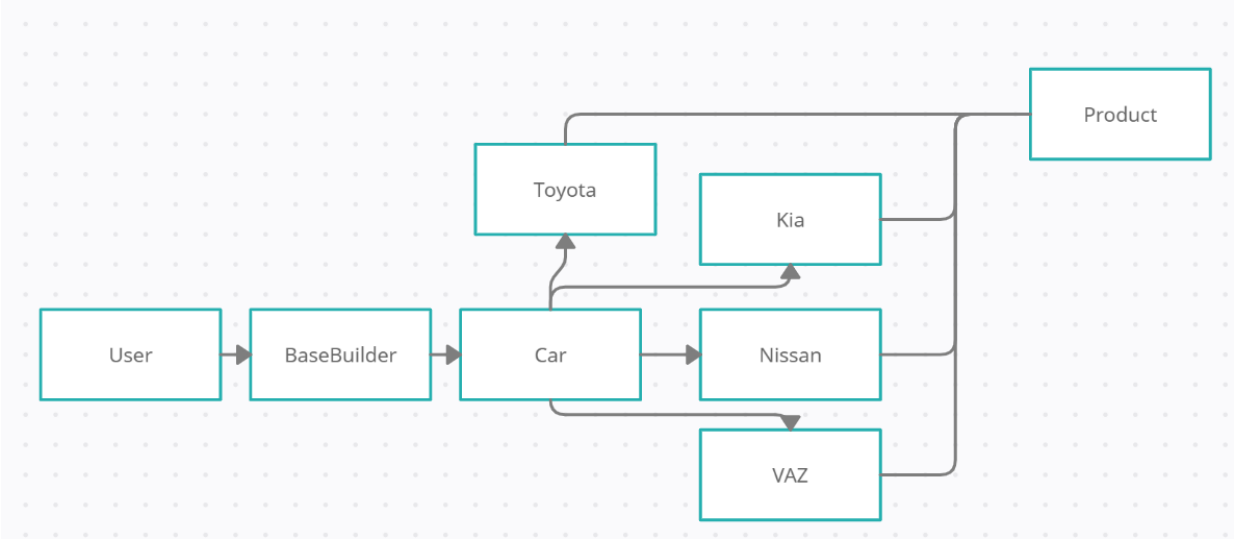

Рисунок 1 – Схема конструирования.

Base builder -Интерфейс строителя объявляет шаги конструирования продуктов, общие для всех видов строителей.
Toyota, Nissan, kia, ВАЗ-Конкретные строители реализуют строительные шаги, каждый по-своему. Конкретные строители могут производить разнородные объекты, не имеющие общего интерфейса.
Продукт — создаваемый объект. Продукты, сделанные разными строителями, не обязаны иметь общий интерфейс.

- определить исключительные ситуации:
  - невозможно открыть/закрыть файл - при возникающих проблемах в работе с файлом;
  -База была пуста– возникает в случае, если пользователь хочет загрузить информацию из пустого файла.
  -Нельзя изменить марку автомобилями и особенности.

4.2 Проектирование:

1) CarsBuilder.cpp: определяет точку входа для консольного приложения.
2) Функция Edit(): позволяет изменять уже существующие автомобили (очищает файлы и записывает данные).
3) Функция Delete():позволяет удалять уже существующие автомобили (очищает файлы и перезаписывать данные).
4) Функция Print - вывод всех машин
5) Функция ReadInFile – Загрузка данных из 4х файлов или очистка файлов.
6) Функция Set…() - установка всех параметров.
7) Функция GetAndSave()- запись данных в файлы

Все сохраненные данные обрабатываются, добавляются, удаляются, записываются в разные файлы.
modelavto- модель
color -цвет
engineType - тип двигателя
engineVolume - объём двигателя
dimention - габариты
year - прозводства
doorCount - количество дверей
model - марка
mark - марка шин
VolumeTrunk -объём багажника

TrunkRoof - возвожность установить багажник на крыше(только для ВАЗ-а)
heatingMirrors- возможность подогрева зеркал (KIA)
heatingSeats - возможность подогрева сидений(Nissan)
АКРР- установка АКРР

3.3. Результаты работы программы:

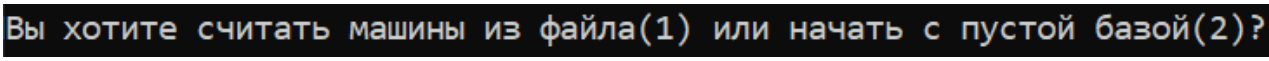

Рисунок 2 –Меню выбора базы данных.

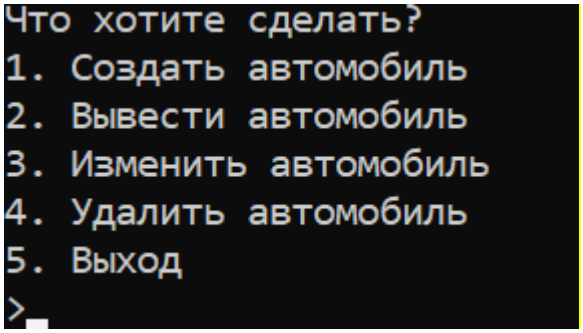

Рисунок 3 – Основное меню

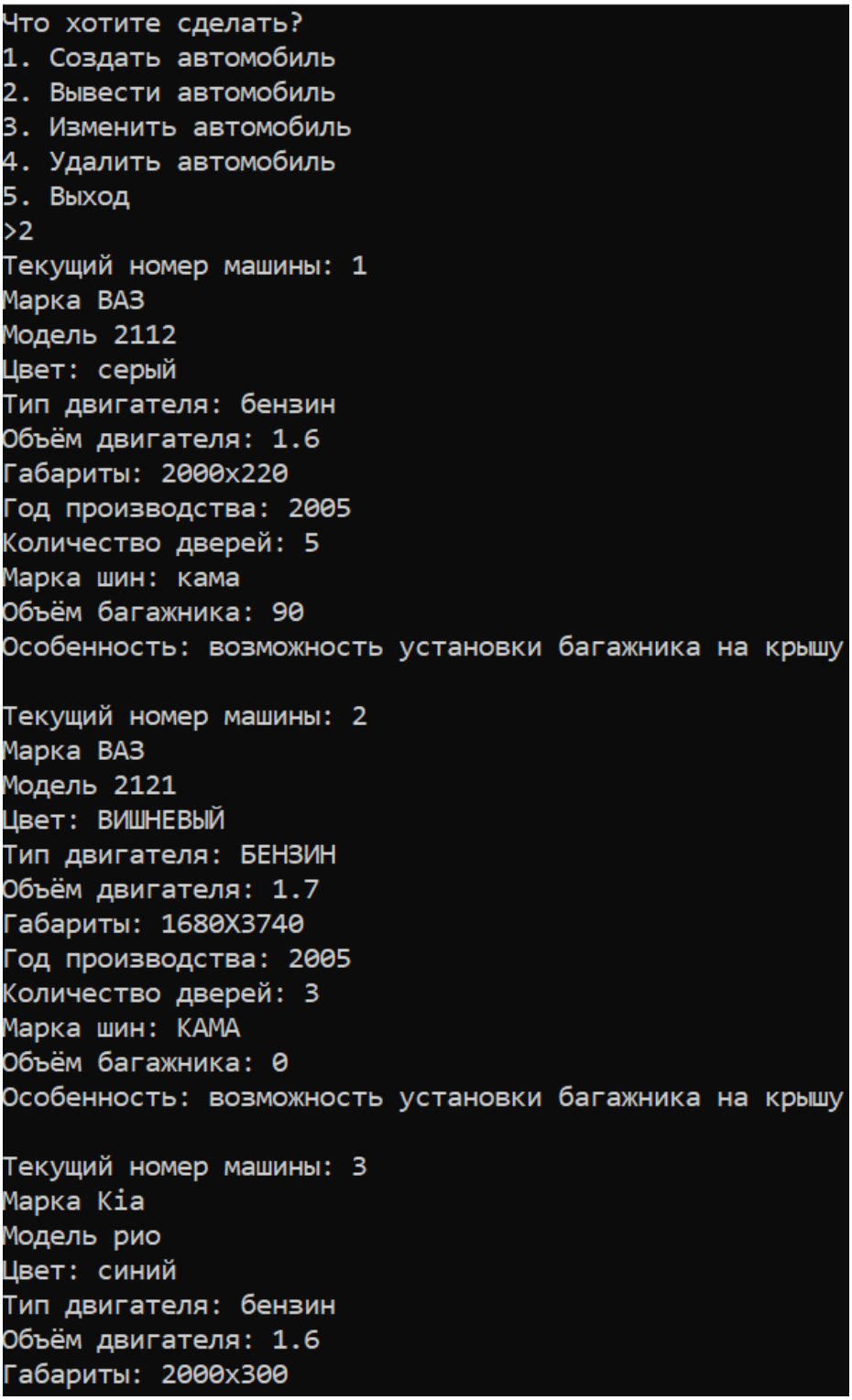

Рисунок 4 – Загрузка данных с файла и вывод на экран

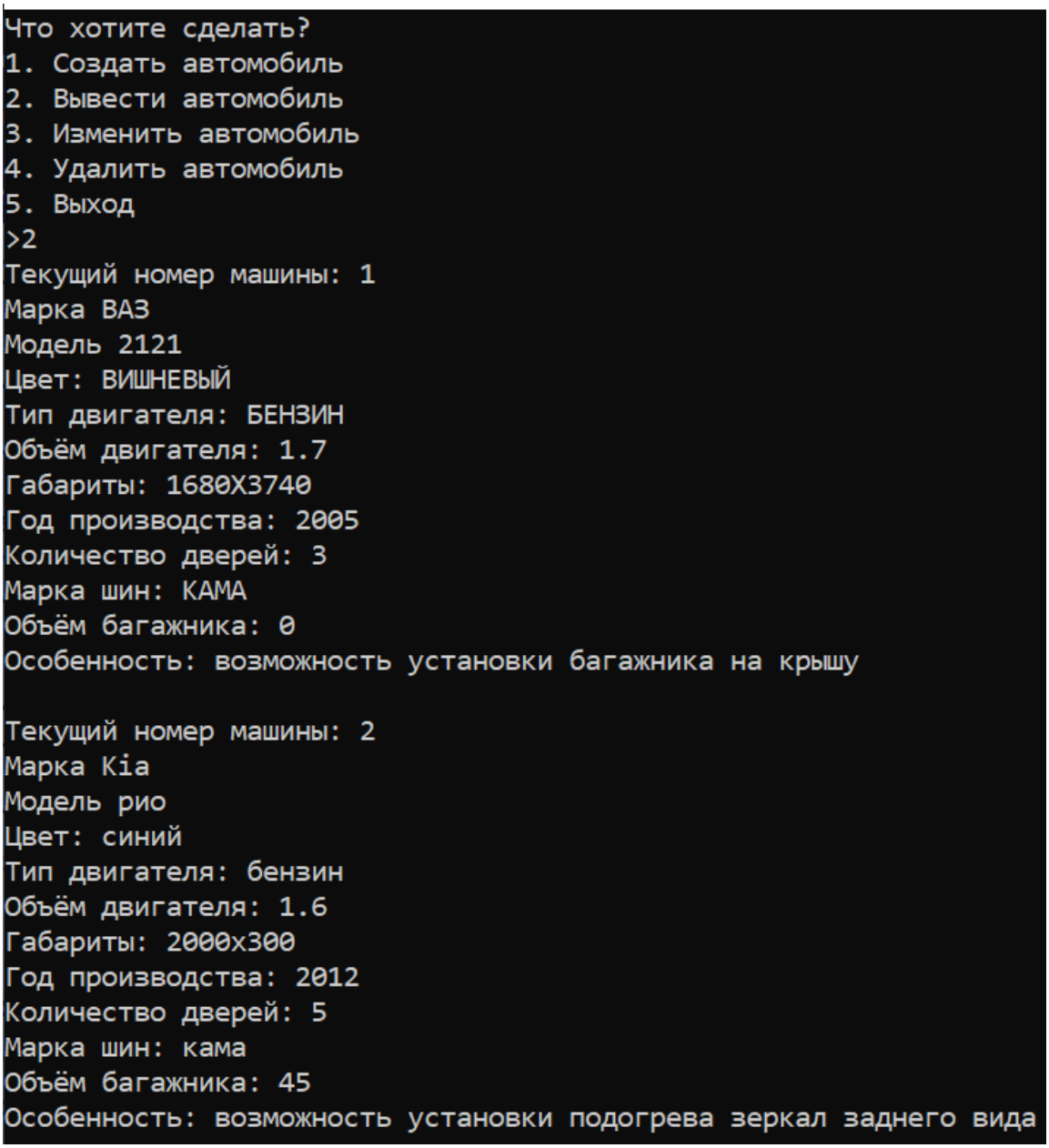

Рисунок 5 – Даннные после удаления 1ого автомобиля

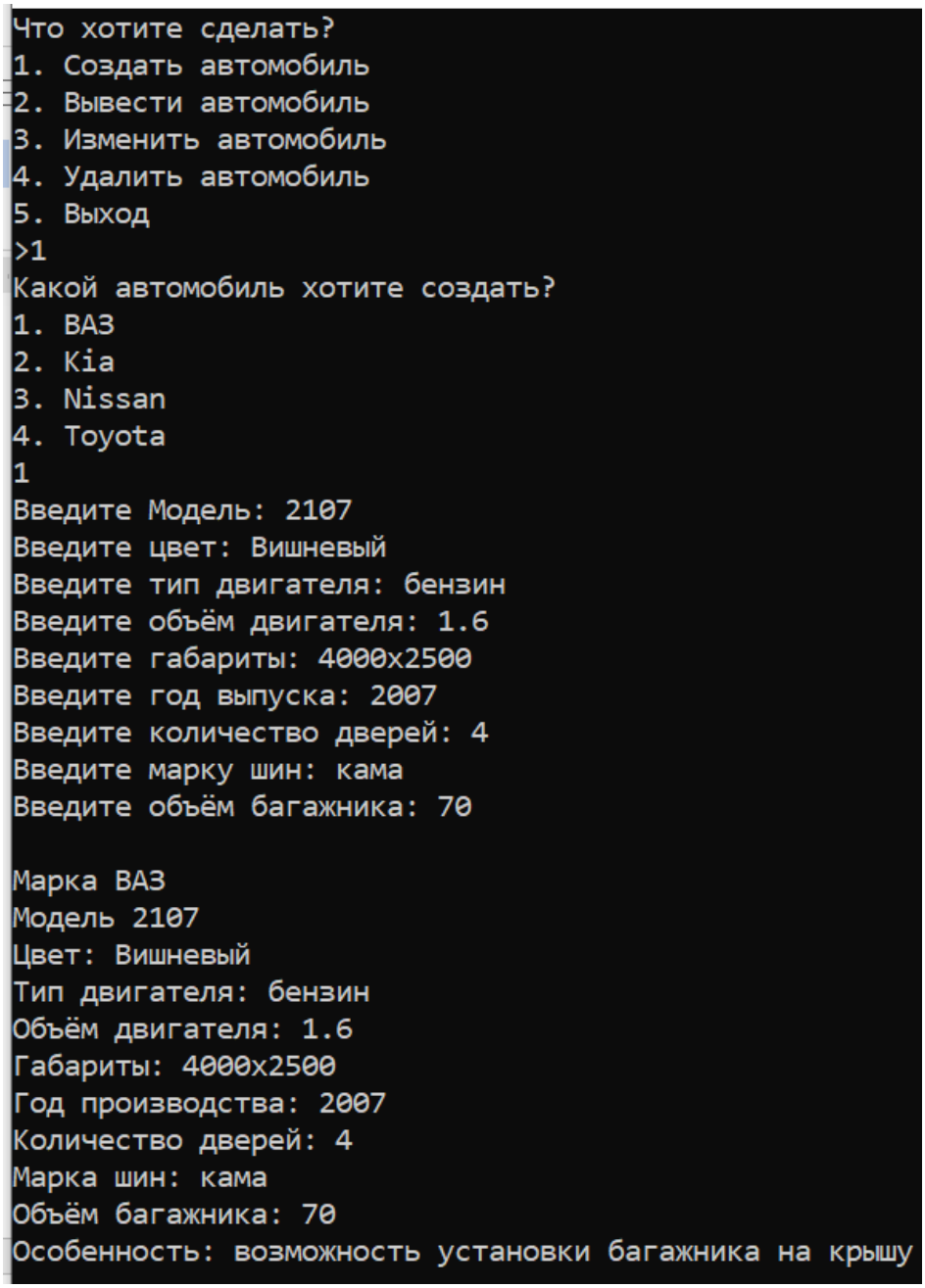

Рисунок 6 – Создание нового автомобиля

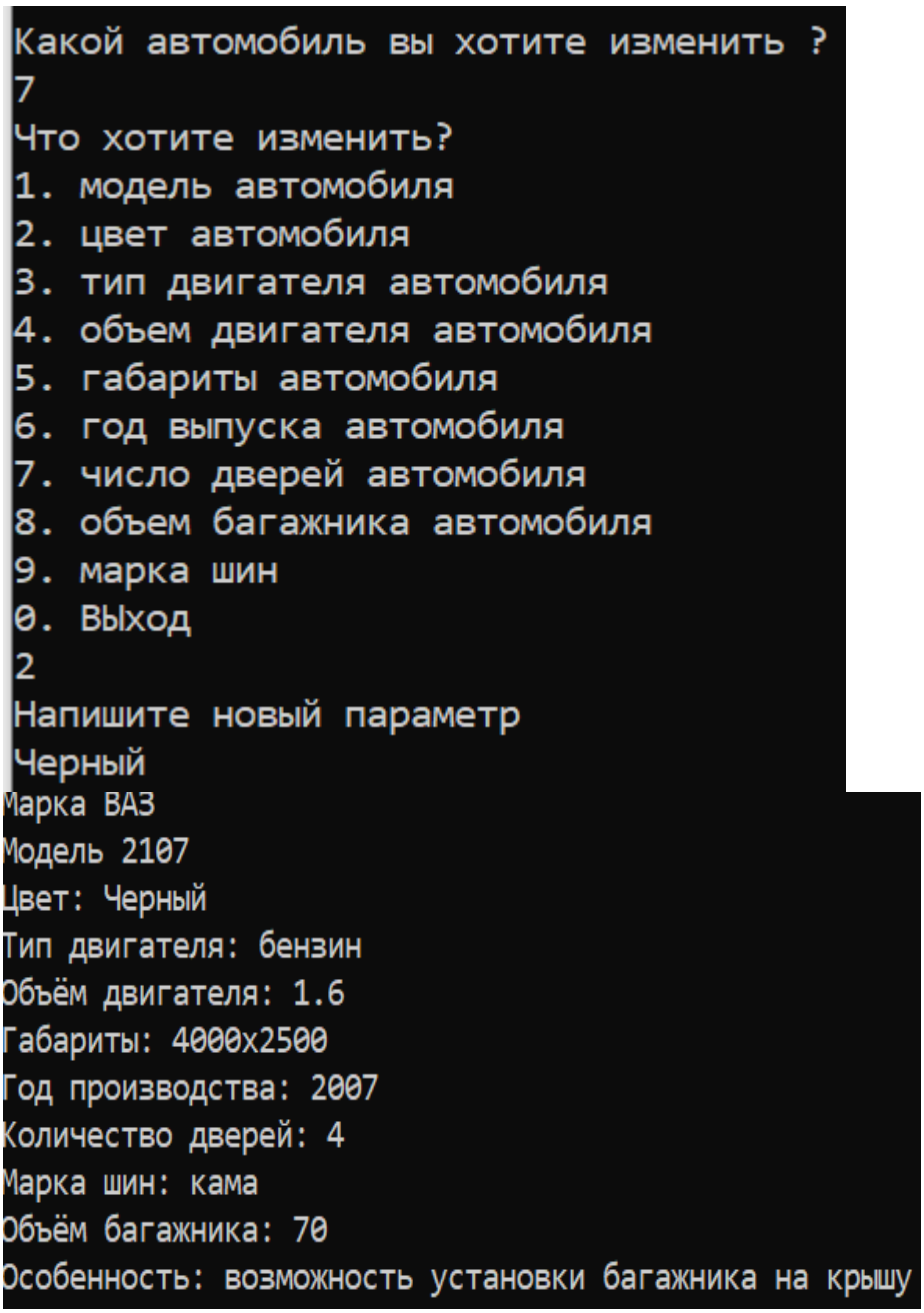

Рисунок 7-изменение одного из параметров

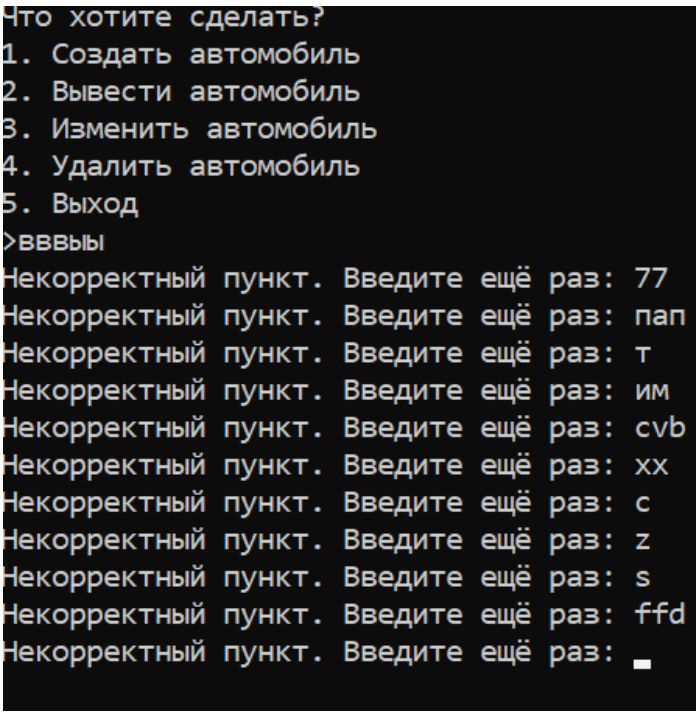

Рисунок 8-защита от ошибок при выбора

5. Заключение:

В проделанной работе был реализован паттерн «Строитель», позволяющий собрать объекты поэлементно. Конечный продукт собирается поэтапно, не зависимо от предыдущих свойств, так же учли особенности каждой марки автомобилей. Настроена защита от не правильного выбора команд, при вводе букв или не корректных команд повторяется запрос на выбор команды. Так же программа поддерживает полностью русский язык.

Были использованы такие инструменты языка программирования C++:

- Указатели и ссылки.
- Виртуальные функции
- Шаблонный класс vector из библиотеки STL, который обеспечивает удобное хранение данных;
- Обработки исключительных ситуации, которые увеличивают надежность программы;

6. Список литературы:

- Электронный ресурс https://refactoring.guru/ru/design-patterns/builder
- Герберт Шилдт. «C++. Базовый курс».
- Т.В. Зудилова, С.В. Одиночкина, И.С. Осетрова, Н.А. Осипов, 2012. Работа пользователя в Microsoft Word 2010 - СПб: НИУ ИТМО, 2012. – 100 с.

7. Листинг программы:

7.1 GitHub

Исходный код программы опубликован на веб-ресурсе GitHub:
https://github.com/michailSobko/kurs

CarsBuilder.cpp

```
#include "stdafx.h";
#include <vector>
#include <locale.h>
#include <cstdlib>
#include "BaseBuilder.h";

#include "Kia.h";
#include "Nissan.h";
#include "Toyota.h";
#include "VAZ.h";

using namespace std;


void CreateVAZ(vector<BaseBuilder*> &cars) {
	VAZ vaz;
	vaz.Read();

	cars.push_back(new VAZ(vaz));

}

void Edit(vector<BaseBuilder*>& cars) {
	int i,number;
	string color, dimention, mark, engineType, modelavto;
	int year, doorCount;
	double engineVolume, VolumeTrunk;
	cout << "Какой автомобиль вы хотите изменить ?" << endl;
	cin >> i;
	i--;
	cout << "Что хотите изменить?" << endl;
	cout << "1. модель автомобиля" << endl;
	cout << "2. цвет автомобиля" << endl;
	cout << "3. тип двигателя автомобиля" << endl;
	cout << "4. объем двигателя автомобиля" << endl;
	cout << "5. габариты автомобиля" << endl;
	cout << "6. год выпуска автомобиля" << endl;
	cout << "7. число дверей автомобиля" << endl;
	cout << "8. объем багажника автомобиля" << endl;
	cout << "9. марка шин " << endl;
	cout << "0. ВЫход" << endl;
	cin >> number;
```

```
while (number < 0 || number > 9) {
        cin.clear();
        cin.ignore(32767, '\n');
        cout << "Некорректный пункт. Введите ещё раз: ";
        cin >> number; // считываем пункт меню заново
}
cout << "Напишите новый параметр" << endl;
switch (number) {
case 1:
        cin >> modelavto;
        cars[i]->SetModelavto(modelavto);
        break;
case 2:
        cin >> color;
        cars[i]->SetColor(color);
        break;
case 3:
        cin >> engineType;
        cars[i]->SetEngineType(engineType);
        break;
case 4:
        cin >> engineVolume;
        cars[i]->SetEngineVolume(engineVolume);
        break;
case 5:
        cin >> dimention;
        cars[i]->SetDimention(dimention);
        break;
case 6:
        cin >> year;
        cars[i]->SetYear(year);
        break;
case 7:
        cin >> doorCount;
        cars[i]->SetDoorCount(doorCount);
        break;
case 8:
        cin >> VolumeTrunk;
        cars[i]->SetVolumeTrunk(VolumeTrunk);
        break;
case 9:
        cin >> mark;
        cars[i]->SetMark(mark);
        break;


}
ofstream out("kia.txt");
out.close();
out.open("BA3.txt");
out.close();
out.open("toyota.txt");
```

```
	out.close();
	out.open("nissan.txt");
	out.close();
	for (int j = 0; j < cars.size(); j++)
		cars[j]->GetAndSave();

}

void Delete(vector<BaseBuilder*>& cars) {

	cout << "Введите № автомобиля, который нужно удалить";
	int i;
	cin >> i;

	while (i < 1 || i > cars.size()) {
		cin.clear();
		cin.ignore(32767, '\n');
		cout << "Некорректный пункт. Введите ещё раз: ";
		cin >> i; // считываем пункт меню заново
	}
	i--;
	cars.erase(cars.begin() + i);
	ofstream out("kia.txt");
	out.close();
	out.open("BA3.txt");
	out.close();
	out.open("toyota.txt");
	out.close();
	out.open("nissan.txt");
	out.close();
	for (int j = 0; j < cars.size(); j++)
		cars[j]->GetAndSave();
}



void CreateKia(vector<BaseBuilder*> &cars) {
	Kia kia;
	kia.Read();

	cars.push_back(new Kia(kia));
}

void CreateNissan(vector<BaseBuilder*> &cars) {
	Nissan nissan;
	nissan.Read();

	cars.push_back(new Nissan(nissan));
}

void CreateToyota(vector<BaseBuilder*> &cars) {
	Toyota toyota;
```

```
	toyota.Read();

	cars.push_back(new Toyota(toyota));
}

void Create(vector<BaseBuilder*> &cars) {
	cout << "Какой автомобиль хотите создать? " << endl;
	cout << "1. ВАЗ" << endl;
	cout << "2. Kia" << endl;
	cout << "3. Nissan" << endl;
	cout << "4. Toyota" << endl;
	int item;

	cin >> item;

	while (item < 1 || item > 4) {
		cin.clear();
		cin.ignore(32767, '\n');
		cout << "Некорректный пункт. Введите ещё раз: ";
		cin >> item;
	}

	switch (item) {
	case 1:
		CreateVAZ(cars);
		break;

	case 2:
		CreateKia(cars);
		break;

	case 3:
		CreateNissan(cars);
		break;

	case 4:
		CreateToyota(cars);
		break;
	}

	cars[cars.size() - 1]->GetAndSave();
}

void Print(vector<BaseBuilder*> cars) {
	for (size_t i = 0; i < cars.size(); i++) {
		cout << "Текущий номер машины: " << (i + 1);
		cars[i]->Print();
	}
}

void ReadInFile(vector<BaseBuilder*> &cars) {
	int item;
```

```
	cout << "Вы хотите считать машины из файла(1) или начать с пустой базой(2)?" <<
endl;
	cin >> item;

	while (item < 1 || item > 2) {
		cin.clear();
		cin.ignore(32767, '\n');
		cout << "Некорректный пункт. Введите ещё раз: ";
		cin >> item;
	}

	if (item == 2) {
		// очистка всех файлов
		ofstream f("BA3.txt");
		f.close();

		f.open("kia.txt");
		f.close();

		f.open("toyota.txt");
		f.close();

		f.open("nissan.txt");
		f.close();

		return;
	}
	считывание из всех файлов всех машин
	ifstream f1("BA3.txt");

	if (f1) {
		while (!f1.eof()) {
			VAZ vaz;
			if (vaz.GetInFile(f1))
				cars.push_back(new VAZ(vaz));
		}

		f1.close();
	}

	ifstream f2("kia.txt");

	if (f2) {
		while (!f2.eof()) {
			Kia kia;
			if (kia.GetInFile(f2))
				cars.push_back(new Kia(kia));
		}

		f2.close();
	}
```

```
	ifstream f3("toyota.txt");

	if (f3) {
		while (!f3.eof()) {
			Toyota toyota;
			if (toyota.GetInFile(f3))
				cars.push_back(new Toyota(toyota));
		}

		f3.close();
	}

	ifstream f4("nissan.txt");

	if (f4) {
		while (!f4.eof()) {
			Nissan nissan;
			if (nissan.GetInFile(f4))
				cars.push_back(new Nissan(nissan));
		}

		f4.close();
	}

	if (cars.size() > 0) {
		cout << "Считанные автомобили: " << endl << endl;
		Print(cars);
	}
	else
		cout << "База была пуста";

	system("pause");
}

int main() {
	setlocale(LC_ALL, "Russian");
	system("chcp 1251");
	vector<BaseBuilder*> cars;

	ReadInFile(cars);

	int item;

	do {
		system("cls");
		cout << "Что хотите сделать?" << endl;
		cout << "1. Создать автомобиль" << endl;
		cout << "2. Вывести автомобиль" << endl;
		cout << "3. Изменить автомобиль" << endl;
		cout << "4. Удалить автомобиль" << endl;
		cout << "5. Выход" << endl;
```

```
		cout << ">";
		cin >> item;


		while (item < 1 || item > 5) {
			cin.clear();
			cin.ignore(32767, '\n');
			cout << "Некорректный пункт. Введите ещё раз: ";
			cin >> item;
		}


		switch (item) {
		case 1:
			Create(cars);
			break;

		case 2:
			Print(cars);
			break;
		case 3:
			Print(cars);
			Edit(cars);
			break;
		case 4:
			Print(cars);
			Delete(cars);
			break;
		}

		if (item != 5) {
			system("pause");
		}
	} while (item != 5);

	return 0;
}
```

BaseBuilder.h

```
#pragma once

#include "Car.h"
#include<fstream>
#include<iostream>
#include<string>
#include <vector>

using namespace std;

class BaseBuilder {
```

```
protected:
	Car car;
public:
	BaseBuilder();


	virtual void SetModel() {};
	virtual void SetTrunkRoof() {};
	virtual void SetHeatingMirrors() {};
	virtual void SetHeatingSeats() {};
	virtual void SetAKPP() {};


	void SetColor(string color);
	void SetEngineType(string engineType);
	void SetEngineVolume(double engineVolume);
	void SetDimention(string dimention);
	void SetYear(int yesr);
	void SetDoorCount(int doorCount);
	void SetMark(string mark);
	void SetModelavto(string modelavto);
	void SetVolumeTrunk(double volumeTrunk);



	virtual void PrintDifference() = 0; // функция вывода особенности

	// функции вывода параметров
	void PrintModelavto();
	void PrintModel();
	void PrintColor();
	void PrintEngineType();
	void PrintEngineVolume();
	void PrintDimention();
	void PrintYear();
	void PrintDoorCount();
	void PrintMark();
	void PrintVolumeTrunk();

	virtual Car GetAndSave() = 0; // получение и сохранение машины
	virtual bool GetInFile(ifstream &in) = 0; // считывание из файла
	virtual void Print() = 0; // вывод
	virtual void Read() = 0; // считывание

	~BaseBuilder();
};
```

BaseBuilder.cpp

```
#include "stdafx.h"
#include "BaseBuilder.h"
//
```

```
BaseBuilder::BaseBuilder()
{
}

BaseBuilder::~BaseBuilder()
{
}

void BaseBuilder::SetModelavto(string modelavto)
{
	car.modelavto = modelavto;
}

void BaseBuilder::SetColor(string color) {
	car.color = color;
}

void BaseBuilder::SetEngineType(string engineType) {
	car.engineType = engineType;
}

void BaseBuilder::SetEngineVolume(double engineVolume) {
	car.engineVolume = engineVolume;
}

void BaseBuilder::SetDimention(string dimention) {
	car.dimention = dimention;
}

void BaseBuilder::SetYear(int year) {
	car.year = year;
}

void BaseBuilder::SetDoorCount(int doorCount) {
	car.doorCount = doorCount;
}

void BaseBuilder::SetMark(string mark) {
	car.mark = mark;
}

void BaseBuilder::SetVolumeTrunk(double volumeTrunk) {
	car.VolumeTrunk = volumeTrunk;
}

void BaseBuilder::PrintModelavto() {
	cout << "Модель " << car.modelavto << endl;
}

void BaseBuilder::PrintModel() {
	cout << "Марка " << car.model << endl;
```

```
}

void BaseBuilder::PrintColor() {
	cout << "Цвет: " << car.color << endl;
}

void BaseBuilder::PrintEngineType() {
	cout << "Тип двигателя: " << car.engineType << endl;
}

void BaseBuilder::PrintEngineVolume() {
	cout << "Объём двигателя: " << car.engineVolume << endl;
}

void BaseBuilder::PrintDimention() {
	cout << "Габариты: " << car.dimention << endl;
}

void BaseBuilder::PrintYear() {
	cout << "Год производства: " << car.year << endl;
}

void BaseBuilder::PrintDoorCount() {
	cout << "Количество дверей: " << car.doorCount << endl;
}

void BaseBuilder::PrintMark() {
	cout << "Марка шин: " << car.mark << endl;
}

void BaseBuilder::PrintVolumeTrunk() {
	cout << "Объём багажника: " << car.VolumeTrunk << endl;
}
```

Car.h

```
#pragma once
#include<fstream>
#include<iostream>
#include<string>

using namespace std;

// машина
class Car {
public:
	string modelavto = ""; // модель
	string color = ""; // цвет
	string engineType = ""; // тип двигателя
	double engineVolume = 0; // объём двигателя
```

```
	string dimention = ""; // габариты
	int year = 0; // год прозводства
	int doorCount = 0; // количество дверей
	string model = ""; // марка
	string mark = ""; // марка шин
	double VolumeTrunk = 0; // объём багажника

	bool TrunkRoof = false; // возвожность установить багажник на крыше(только для ВАЗ-а)
	bool heatingMirrors = false; // возможность подогрева зеркал (KIA)
	bool heatingSeats = false; // возможность подогрева сидений(Nissan)
	bool AKPP = false; // установка АКРР

	Car();
	~Car();
};
```

Car.cpp

```
#include "stdafx.h"
#include "Car.h"


Car::Car() {
}


Car::~Car() {
}
```

Headler.h

```
#pragma once

Kia.cpp
#include "stdafx.h"
#include "Kia.h"


Kia::Kia()
{
}

// меняем модель
void Kia::SetModel() {
	car.model = "Kia";
}

// меняем установку особого параметра
void Kia::SetHeatingMirrors() {
	car.heatingMirrors = true;
```

```
}

// считывание из файла
bool Kia::GetInFile(ifstream &in) {
	car.model = "Kia";
	if (!(in >> car.modelavto))
		return false;

	if (!(in >> car.color))
		return false;

	if (!(in >> car.engineType))
		return false;

	if (!(in >> car.engineVolume))
		return false;

	if (!(in >> car.dimention))
		return false;

	if (!(in >> car.year))
		return false;

	if (!(in >> car.doorCount))
		return false;

	if (!(in >> car.mark))
		return false;

	if (!(in >> car.VolumeTrunk))
		return false;

	if (!(in >> car.TrunkRoof))
		return false;

	if (!(in >> car.heatingMirrors))
		return false;

	if (!(in >> car.heatingSeats))
		return false;

	if (!(in >> car.AKPP))
		return false;

	return true;
}

// вывод
void Kia::Print() {
	cout << endl;
	PrintModel();
	PrintModelavto()	;
```

```
        PrintColor();
        PrintEngineType();
        PrintEngineVolume();
        PrintDimention();
        PrintYear();
        PrintDoorCount();
        PrintMark();
        PrintVolumeTrunk();
        PrintDifference();
        cout << endl;
}

// чтение с клавиатуры
void Kia::Read() {
        string color, dimention, mark, engineType, modelavto;
        int year, doorCount;
        double engineVolume, VolumeTrunk;

        cout << "Введите модель: ";
        cin >> modelavto;
        SetColor(modelavto);

        cout << "Введите цвет: ";
        cin >> color;
        SetColor(color);

        cout << "Введите тип двигателя: ";
        cin >> engineType;
        SetEngineType(engineType);

        cout << "Введите объём двигателя: ";
        cin >> engineVolume;
        SetEngineVolume(engineVolume);

        cout << "Введите габариты: ";
        cin >> dimention;
        SetDimention(dimention);

        cout << "Введите год выпуска: ";
        cin >> year;
        SetYear(year);

        cout << "Введите количество дверей: ";
        cin >> doorCount;
        SetDoorCount(doorCount);

        cout << "Введите марку шин: ";
        cin >> mark;
        SetMark(mark);

        cout << "Введите объём багажника: ";
        cin >> VolumeTrunk;
```

```
	SetVolumeTrunk(VolumeTrunk);

	SetModel();
	SetHeatingMirrors();
}

// изменение вывода особенности
void Kia::PrintDifference() {
	cout << "Особенность: возможность установки подогрева зеркал заднего вида" <<
endl;
}

// сохранение в файл и вывод на экран
Car Kia::GetAndSave() {
	Print();
	ofstream out("kia.txt", ios::app);
	out << car.modelavto << " ";
	out << car.color << " ";
	out << car.engineType << " ";
	out << car.engineVolume << " ";
	out << car.dimention << " ";
	out << car.year << " ";
	out << car.doorCount << " ";
	out << car.mark << " ";
	out << car.VolumeTrunk << " ";
	out << (car.TrunkRoof ? "1" : "0") << " ";
	out << (car.heatingMirrors ? "1" : "0") << " ";
	out << (car.heatingSeats ? "1" : "0") << " ";
	out << (car.AKPP ? "1" : "0") << endl;

	out.close();

	return car;
}

Kia::~Kia()
{
}
```

Kia.h

```
#pragma once
#include "BaseBuilder.h"

// киа - содержит только переопределённые  методы
class Kia : public BaseBuilder {
public:
	Kia();
	void SetModel();
	void SetHeatingMirrors();
```

```
	bool GetInFile(ifstream &in);
	void Print();
	Car GetAndSave();
	void Read();
	void PrintDifference();

	~Kia();
};
```

Nissan.cpp

```
#include "stdafx.h"
#include "Nissan.h"


Nissan::Nissan()
{
}

void Nissan::SetModel() {
	car.model = "Nissan";
}

void Nissan::SetHeatingSeats() {
	car.heatingSeats = true;
}

bool Nissan::GetInFile(ifstream &in) {
	car.model = "Nissan";

	if (!(in >> car.modelavto))
		return false;

	if (!(in >> car.color))
		return false;

	if (!(in >> car.engineType))
		return false;

	if (!(in >> car.engineVolume))
		return false;

	if (!(in >> car.dimention))
		return false;

	if (!(in >> car.year))
		return false;

	if (!(in >> car.doorCount))
		return false;
```

```
	if (!(in >> car.mark))
		return false;

	if (!(in >> car.VolumeTrunk))
		return false;

	if (!(in >> car.TrunkRoof))
		return false;

	if (!(in >> car.heatingMirrors))
		return false;

	if (!(in >> car.heatingSeats))
		return false;

	if (!(in >> car.AKPP))
		return false;

	return true;
}

void Nissan::Print() {
	cout << endl;
	PrintModelavto();
	PrintModel();
	PrintColor();
	PrintEngineType();
	PrintEngineVolume();
	PrintDimention();
	PrintYear();
	PrintDoorCount();
	PrintMark();
	PrintVolumeTrunk();
	PrintDifference();
	cout << endl;
}

void Nissan::Read() {
	string color, dimention, mark, engineType, modelavto;
	int year, doorCount;
	double engineVolume, VolumeTrunk;

	cout << "Введите модель: ";
	cin >> modelavto;
	SetModelavto(modelavto);

	cout << "Введите цвет: ";
	cin >> color;

	SetColor(color);

	cout << "Введите тип двигателя: ";
```

```
	cin >> engineType;
	SetEngineType(engineType);

	cout << "Введите объём двигателя: ";
	cin >> engineVolume;
	SetEngineVolume(engineVolume);

	cout << "Введите габариты: ";
	cin >> dimention;
	SetDimention(dimention);

	cout << "Введите год выпуска: ";
	cin >> year;
	SetYear(year);

	cout << "Введите количество дверей: ";
	cin >> doorCount;
	SetDoorCount(doorCount);

	cout << "Введите марку шин: ";
	cin >> mark;
	SetMark(mark);

	cout << "Введите объём багажника: ";
	cin >> VolumeTrunk;
	SetVolumeTrunk(VolumeTrunk);

	SetModel();
	SetHeatingSeats();
}

// изменение вывода особенности
void Nissan::PrintDifference() {
	cout << "Особенность: возможность установки подогрева сидений" << endl;
}

Car Nissan::GetAndSave() {
	Print();
	ofstream out("nissan.txt", ios::app);

	out << car.modelavto << " ";
	out << car.color << " ";
	out << car.engineType << " ";
	out << car.engineVolume << " ";
	out << car.dimention << " ";
	out << car.year << " ";
	out << car.doorCount << " ";
	out << car.mark << " ";
	out << car.VolumeTrunk << " ";
	out << (car.TrunkRoof ? "1" : "0") << " ";
	out << (car.heatingMirrors ? "1" : "0") << " ";
	out << (car.heatingSeats ? "1" : "0") << " ";
```

```
	out << (car.AKPP ? "1" : "0") << endl;

	out.close();

	return car;
}

Nissan::~Nissan()
{
}
```

Nissan.h

```
#pragma once
#include "BaseBuilder.h"

// ниссан - содержит только переопределённые  методы
class Nissan : public BaseBuilder {
public:
	Nissan();
	void SetModel();
	void SetHeatingSeats();

	Car GetAndSave();
	bool GetInFile(ifstream &in);
	void Print();
	void Read();
	void PrintDifference();

	~Nissan();
};
```

Kia.cpp

```
#include "stdafx.h"
#include "Kia.h"
Kia::Kia()
{
}
void Kia::SetModel() {
	car.model = "Kia";
}

void Kia::SetHeatingMirrors() {
	car.heatingMirrors = true;
}

bool Kia::GetInFile(ifstream &in) {
	car.model = "Kia";
	if (!(in >> car.modelavto))
		return false;
```

```
	if (!(in >> car.color))
		return false;

	if (!(in >> car.engineType))
		return false;

	if (!(in >> car.engineVolume))
		return false;

	if (!(in >> car.dimention))
		return false;

	if (!(in >> car.year))
		return false;

	if (!(in >> car.doorCount))
		return false;

	if (!(in >> car.mark))
		return false;

	if (!(in >> car.VolumeTrunk))
		return false;

	if (!(in >> car.TrunkRoof))
		return false;

	if (!(in >> car.heatingMirrors))
		return false;

	if (!(in >> car.heatingSeats))
		return false;

	if (!(in >> car.AKPP))
		return false;

	return true;
}

// вывод
void Kia::Print() {
	cout << endl;
	PrintModel();
	PrintModelavto()	;
	PrintColor();
	PrintEngineType();
	PrintEngineVolume();
	PrintDimention();
	PrintYear();
	PrintDoorCount();
	PrintMark();
	PrintVolumeTrunk();
```

```
	PrintDifference();
	cout << endl;
}

void Kia::Read() {
	string color, dimention, mark, engineType, modelavto;
	int year, doorCount;
	double engineVolume, VolumeTrunk;

	cout << "Введите модель: ";
	cin >> modelavto;
	SetColor(modelavto);

	cout << "Введите цвет: ";
	cin >> color;
	SetColor(color);

	cout << "Введите тип двигателя: ";
	cin >> engineType;
	SetEngineType(engineType);

	cout << "Введите объём двигателя: ";
	cin >> engineVolume;
	SetEngineVolume(engineVolume);

	cout << "Введите габариты: ";
	cin >> dimention;
	SetDimention(dimention);

	cout << "Введите год выпуска: ";
	cin >> year;
	SetYear(year);

	cout << "Введите количество дверей: ";
	cin >> doorCount;
	SetDoorCount(doorCount);

	cout << "Введите марку шин: ";
	cin >> mark;
	SetMark(mark);

	cout << "Введите объём багажника: ";
	cin >> VolumeTrunk;
	SetVolumeTrunk(VolumeTrunk);

	SetModel();
	SetHeatingMirrors();
}


void Kia::PrintDifference() {
```

```
	cout << "Особенность: возможность установки подогрева зеркал заднего вида" <<
endl;
}
Car Kia::GetAndSave() {
	Print();
	ofstream out("kia.txt", ios::app);
	out << car.modelavto << " ";
	out << car.color << " ";
	out << car.engineType << " ";
	out << car.engineVolume << " ";
	out << car.dimention << " ";
	out << car.year << " ";
	out << car.doorCount << " ";
	out << car.mark << " ";
	out << car.VolumeTrunk << " ";
	out << (car.TrunkRoof ? "1" : "0") << " ";
	out << (car.heatingMirrors ? "1" : "0") << " ";
	out << (car.heatingSeats ? "1" : "0") << " ";
	out << (car.AKPP ? "1" : "0") << endl;

	out.close();

	return car;
}

Kia::~Kia()
{
}
```

Kia.h

```
#pragma once
#include "BaseBuilder.h"

// киа - содержит только переопределённые  методы
class Kia : public BaseBuilder {
public:
	Kia();
	void SetModel();
	void SetHeatingMirrors();

	bool GetInFile(ifstream &in);
	void Print();
	Car GetAndSave();
	void Read();
	void PrintDifference();

	~Kia();
};
```

Toyota.cpp

```
#include "stdafx.h"
#include "Toyota.h"


Toyota::Toyota()
{
}

void Toyota::SetModel() {
	car.model = "Toyota";
}

void Toyota::SetAKPP() {
	car.AKPP = true;
}

bool Toyota::GetInFile(ifstream &in) {
	car.model = "Toyota";
	if (!(in >> car.modelavto))
		return false;

	if (!(in >> car.color))
		return false;

	if (!(in >> car.engineType))
		return false;

	if (!(in >> car.engineVolume))
		return false;

	if (!(in >> car.dimention))
		return false;

	if (!(in >> car.year))
		return false;

	if (!(in >> car.doorCount))
		return false;

	if (!(in >> car.mark))
		return false;

	if (!(in >> car.VolumeTrunk))
		return false;

	if (!(in >> car.TrunkRoof))
		return false;

	if (!(in >> car.heatingMirrors))
		return false;
```

```
	if (!(in >> car.heatingSeats))
		return false;

	if (!(in >> car.AKPP))
		return false;

	return true;
}

// вывод
void Toyota::Print() {
	cout << endl;
	PrintModel();
	PrintModelavto();
	PrintColor();
	PrintEngineType();
	PrintEngineVolume();
	PrintDimention();
	PrintYear();
	PrintDoorCount();
	PrintMark();
	PrintVolumeTrunk();
	PrintDifference();
	cout << endl;
}


void Toyota::Read() {
	string color, dimention, mark, engineType,modelavto;
	int year, doorCount;
	double engineVolume, VolumeTrunk;

	cout << "Введите модель: ";
	cin >> modelavto;
	SetModelavto(modelavto);

	cout << "Введите цвет: ";
	cin >> color;

	SetColor(color);

	cout << "Введите тип двигателя: ";
	cin >> engineType;
	SetEngineType(engineType);

	cout << "Введите объём двигателя: ";
	cin >> engineVolume;
	SetEngineVolume(engineVolume);

	cout << "Введите габариты: ";
	cin >> dimention;
```

```
	SetDimention(dimention);

	cout << "Введите год выпуска: ";
	cin >> year;
	SetYear(year);

	cout << "Введите количество дверей: ";
	cin >> doorCount;
	SetDoorCount(doorCount);

	cout << "Введите марку шин: ";
	cin >> mark;
	SetMark(mark);

	cout << "Введите объём багажника: ";
	cin >> VolumeTrunk;
	SetVolumeTrunk(VolumeTrunk);

	SetModel();
	SetAKPP();
}

// изменение вывода особенности
void Toyota::PrintDifference() {
	cout << "Особенность: установка АКПП" << endl;
}
Car Toyota::GetAndSave() {
	Print();
	ofstream out("toyota.txt", ios::app);

	out << car.modelavto << " ";
	out << car.color << " ";
	out << car.engineType << " ";
	out << car.engineVolume << " ";
	out << car.dimention << " ";
	out << car.year << " ";
	out << car.doorCount << " ";
	out << car.mark << " ";
	out << car.VolumeTrunk << " ";
	out << (car.TrunkRoof ? "1" : "0") << " ";
	out << (car.heatingMirrors ? "1" : "0") << " ";
	out << (car.heatingSeats ? "1" : "0") << " ";
	out << (car.AKPP ? "1" : "0") << endl;

	out.close();

	return car;
}

Toyota::~Toyota()
{
}
```

Toyota.h

```
#pragma once
#include "BaseBuilder.h"

class Toyota : public BaseBuilder {
public:
	Toyota();
	void SetModel();
	void SetAKPP();

	bool GetInFile(ifstream &in);
	void Print();
	void Read();
	void PrintDifference();

	Car GetAndSave();
	~Toyota();
};
```

Vaz.cpp

```
#include "stdafx.h"
#include "VAZ.h"


VAZ::VAZ()
{
}

void VAZ::SetModel() {
	car.model = "ВАЗ";
}

void VAZ::SetTrunkRoof() {
	car.TrunkRoof = true;
}

bool VAZ::GetInFile(ifstream &in) {
	car.model = "ВАЗ";
	if (!(in >> car.modelavto))
		return false;

	if (!(in >> car.color))
		return false;


	if (!(in >> car.engineType))
		return false;

	if (!(in >> car.engineVolume))
```

```
		return false;

	if (!(in >> car.dimention))
		return false;

	if (!(in >> car.year))
		return false;

	if (!(in >> car.doorCount))
		return false;

	if (!(in >> car.mark))
		return false;

	if (!(in >> car.VolumeTrunk))
		return false;

	if (!(in >> car.TrunkRoof))
		return false;

	if (!(in >> car.heatingMirrors))
		return false;

	if (!(in >> car.heatingSeats))
		return false;

	if (!(in >> car.AKPP))
		return false;

	return true;
}

void VAZ::Print() {
	cout << endl;
	PrintModel();
	PrintModelavto();
	PrintColor();
	PrintEngineType();
	PrintEngineVolume();
	PrintDimention();
	PrintYear();
	PrintDoorCount();
	PrintMark();
	PrintVolumeTrunk();
	PrintDifference();
	cout << endl;
}


void VAZ::Read() {
	string color, dimention, mark, engineType ,modelavto;
	int year, doorCount;
```

```
	double engineVolume, VolumeTrunk;

	cout << "Введите Модель: ";
	cin >> modelavto;
	SetModelavto(modelavto);

	cout << "Введите цвет: ";
	cin >> color;
	SetColor(color);

	cout << "Введите тип двигателя: ";
	cin >> engineType;
	SetEngineType(engineType);

	cout << "Введите объём двигателя: ";
	cin >> engineVolume;
	SetEngineVolume(engineVolume);

	cout << "Введите габариты: ";
	cin >> dimention;
	SetDimention(dimention);

	cout << "Введите год выпуска: ";
	cin >> year;
	SetYear(year);

	cout << "Введите количество дверей: ";
	cin >> doorCount;
	SetDoorCount(doorCount);

	cout << "Введите марку шин: ";
	cin >> mark;
	SetMark(mark);

	cout << "Введите объём багажника: ";
	cin >> VolumeTrunk;
	SetVolumeTrunk(VolumeTrunk);

	SetModel();
	SetTrunkRoof();
}

// изменение вывода особенности
void VAZ::PrintDifference() {
	cout << "Особенность: возможность установки багажника на крышу" << endl;
}


// сохранение в файл и вывод на экран
Car VAZ::GetAndSave() {
	Print();
	ofstream out("ВАЗ.txt", ios::app);
```

```
        out << car.modelavto << " ";
        out << car.color << " ";
        out << car.engineType << " ";
        out << car.engineVolume << " ";
        out << car.dimention << " ";
        out << car.year << " ";
        out << car.doorCount << " ";
        out << car.mark << " ";
        out << car.VolumeTrunk << " ";
        out << (car.TrunkRoof ? "1" : "0") << " ";
        out << (car.heatingMirrors ? "1" : "0") << " ";
        out << (car.heatingSeats ? "1" : "0") << " ";
        out << (car.AKPP ? "1" : "0") << endl;

        out.close();

        return car;
}
void VAZ::Edit() {
        string color, dimention, mark, engineType, modelavto;
        int year, doorCount;
        double engineVolume, VolumeTrunk;
        cout << "Введите марку: ";
        cin >> modelavto;
        SetModelavto(modelavto);

        cout << "Введите цвет: ";
        cin >> color;

        SetColor(color);

        cout << "Введите тип двигателя: ";
        cin >> engineType;
        SetEngineType(engineType);

        cout << "Введите объём двигателя: ";
        cin >> engineVolume;
        SetEngineVolume(engineVolume);

        cout << "Введите габариты: ";
        cin >> dimention;
        SetDimention(dimention);

        cout << "Введите год выпуска: ";
        cin >> year;
        SetYear(year);

        cout << "Введите количество дверей: ";
        cin >> doorCount;
        SetDoorCount(doorCount);

        cout << "Введите марку шин: ";
```

```
	cin >> mark;
	SetMark(mark);

	cout << "Введите объём багажника: ";
	cin >> VolumeTrunk;
	SetVolumeTrunk(VolumeTrunk);

	SetModel();
	SetTrunkRoof();
}
VAZ::~VAZ()
{
}
```

Vaz.h

```
#pragma once
#include<fstream>
#include<iostream>
#include<string>
#include "BaseBuilder.h"


class VAZ: public BaseBuilder {
public:
	VAZ();
	void SetModel();
	void SetTrunkRoof();

	Car GetAndSave();
	bool GetInFile(ifstream &in);
	void Print();
	void Read();
	void Edit();
	void PrintDifference();
	~VAZ();
};
```

Stdafx.cpp

```
#include "stdafx.h"
```