**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ…………………………………………………………………….....5 1.ОБЗОР ЛИТЕРАТУРЫ…………………………………..……………….........6

1.1 Обзор методов и алгоритмов постановленной задачи …………………...6

2.ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ..................................................8

2.1 Class Transport….……………………….………………….…...…….….…8

2.2 Class Legkovie…………………………………..……………………….…...8

2.3 Class Avtobusi….…………………………..………..………………….........9

2.4 Class Salon….…………………………………………………………..........9

2.5 Class BY….………………………………………..………..…...………..…9

2.6 Class Microavtobus………………………………………...……...……..…10

2.7 Class Electrobus……………………………………………...……………...11

2.8 Class Shablon<A>………………………………………………………..….11

2.9 Class Text\_files<A>………………………………………………………....12

2.10 Class Exeption………………………………………………………...……12

3.РАЗРАБОТКА ПРОГРАММЫХ МОДУЛЕЙ………......................................13 4.РЕЗУЛЬТАТЫ РАБОТЫ…………...................................................................16 ЗАКЛЮЧЕНИЕ.....................................................................................................20 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ............................................21 ПРИЛОЖЕНИЕ А СТРУКТУРНАЯ СХЕМА....................................................22 ПРИЛОЖЕНИЕ Б СХЕМА АЛГОРИТМА........................................................23 ПРИЛОЖЕНИЕ В ДИАГРАММА КЛАССОВ..................................................24 ПРИЛОЖЕНИЕ Г КОД ПРОГРАММЫ.............................................................25

ПРИЛОЖЕНИЕ Д ВЕДОМОСТЬ ДОКУМЕНТОВ……..................................25

**ВВЕДЕНИЕ**

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. В общем С++ - язык широкого пользования, на котором можно создавать практически любые виды программ.

1. Название C ++ указывает на эволюционный характер изменений по сравнению с C. “++” - это оператор увеличения C.
2. C ++ является одним из преобладающих языков для разработки всех видов технического и коммерческого программного обеспечения.
3. C ++ представляет объектно-ориентированное программирование, отсутствующее в C. Как и другие вещи, C ++ поддерживает четыре основные функции ООП: инкапсуляцию, полиморфизм, абстракцию и наследование.
4. C ++ получил функции ООП от языка программирования Simula67.
5. Функция - это минимальное требование для запуска программы на C ++.
6. C и C ++ были изобретены в одном и том же месте, то есть в AT & T (American Telephone & Telegraph) bell laboratories, расположенной в США.
7. Не чисто объектно-ориентированный: мы можем написать код на C ++ без использования классов, и он будет компилироваться без отображения какого-либо сообщения об ошибке. Язык имеет некоторые расширения по сравнению с C, которые делают ООП и универсальное программирование более удобными.
8. Многие языки программирования находятся под влиянием c ++, некоторые из которых включают C #, Java и даже более новые версии C.
9. Концепция ссылочных переменных: перегрузка операторов заимствована из языка программирования Algol 68.
10. Основной причиной успеха C ++ является то, что он поддерживает различные стили программирования.
11. Это многопарадигмальный язык программирования, который поддерживает не только парадигму ООП, но и многие другие парадигмы.

C ++ стандартизирован [Международной организацией по стандартизации](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO), последняя версия стандарта ратифицирована и опубликована ISO в декабре 2020 года как [*ISO / IEC 14882:2020*](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B_standard#Standardization) (неофициально известный как [C ++ 20](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B20)). Язык программирования C ++ был первоначально стандартизирован в 1998 году как *ISO / IEC 14882: 1998*, который затем был изменен стандартами [C ++ 03](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B03), [C ++ 11](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B11), [C ++ 14](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B14) и [C ++ 17](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B17). Текущий стандарт [C ++ 20](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B20) заменяет их новыми функциями и расширенной [стандартной библиотекой](https://translated.turbopages.org/proxy_u/en-ru.ru.18ab9744-6399de21-79878bf3-74722d776562/https/en.wikipedia.org/wiki/C%2B%2B_Standard_library).

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание [операционных систем](https://ru.m.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), разнообразных прикладных программ, [драйверов](https://ru.m.wikipedia.org/wiki/%D0%94%D1%80%D0%B0%D0%B9%D0%B2%D0%B5%D1%80) устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также компьютерных игр.

С++ является мощным языком, унаследовав от Си богатые возможности по работе с памятью. Поэтому нередко С++ находит свое применение в системном программировании, в частности, при создании операционных систем, драйверов, различных утилит, антивирусов и т.д. К слову сказать, ОС Windows большей частью написана на С++. Но только системным программированием применение данного языка не ограничивается. С++ можно использовать в программах любого уровня, где важны скорость работы и производительность. Нередко он применяется для создания графических приложений, различных прикладных программ. Также особенно часто его используют для создания игр с богатой насыщенной визуализацией. В общем С++ - язык широкого пользования, на котором можно создавать практически любые виды программ.

С++ является компилируемым языком, а это значит, что компилятор транслирует исходный код на С++ в исполняемый файл, который содержит набор машинных инструкций.

**1 ОБЗОР ЛИТЕРАТУРЫ**

В качестве теории и материала использовалось четвертое издание книги «Объектно-ориентированное программирование в С++» от Роберта Лафоре.

Так же использовались методическое пособие “Объектно-ориентированное программирование на языке С++” автора Луцик, Ю. А., Электронный ресурс «Структура программного обеспечения ПК»,Луцик, Ю. А. Объектно-ориентированное программирование на языке С++: учеб. пособие по курсу «Объектно-ориентированное программирование» для студ. спец. «Вычислительные машины, системы и сети»

В каждом источнике есть полезная информация, которая поможет в написании программы, пояснения различных аспектов ООП и примеры.

* 1. **Постановка задачи**

Задачей является написать информационную систему «Авторынок». Программа должна иметь удобный интерфейс с необходимыми пунктами меню. Информация должна храниться в нескольких файлах, связанных определённым образом. Программа должна позволять узнать информацию об автомобилях и иметь другие функции в зависимости от специфики.

Программа должна выполнять определённые действия с данными, CRUD-операции. В программе должна быть реализована иерархия классов, должны использоваться контейнеры, должна производиться обработка исключительных ситуаций. Функции редактирования, добавления и удаления информации должны иметь возможность отмены.

* 1. **Обзор методов и алгоритмов решения поставленной задачи**

Необходимо написать программу, которая будет иметь несколько разделов с разными классами транспорта. Каждый раздел будет иметь собственный файл для хранения информации. В программе будет реализована возможность авторизации с получением не полного и полного функционала программы. Полный будет позволять добавлять, изменять, просматривать, искать, удалять информацию, а второй – только возможность просмотреть список транспорта и выполнить поиск.

Для реализации данной программы буду реализовывать двунаправленую очередь. Таким образом буду осуществлять связь между объектами одного класса транспорта. Для работы с разными классами транспорта буду использовать шаблонный класс, который позволит применять одни и те же методы к разным классам транспорта.

Метод добавления будет позволять беспрерывно вводить новые объекты класса, динамически выделяя для них память. Метод просмотра будет демонстрировать список транспорта выбранной категории. Метод изменения позволит выбрать из списка нужный объект для именения его характеристик. Будет возможность очистки всей очереди и удаления конкретных ее элементов. Так же можно будет осуществить поиск, создавая прототип искомого объекта. Будет возможность отменить действие, загружая старую информацию из файла в программу. При выходе в главное меню будет происходить автоматическое сохранение данных в соответствующий файл, а при выборе нужной категории для работы информация будет читаться из него.

Блок хранения данных будет состоять из 5-ти файлов, с которыми работает программа. В четырёх из них хранятся данные об автомобилях, а в 5-ом храниться пароль для его проверки в процессе авторизации от имени администратора, реализуемого в программе.

После запуска программы и выбора конкретной категории транспорта для дальнейшей работы, информация будет загружена из хранилища в оперативную память компьютера, для последующей работы с ней.

В программе представлено несколько структурных блоков:

-блок в котором выбирается категория транспорта

-блок в котором выбирается метод, который будет применён для работы с объектом.

-блок изменения полей объектов.

Все блоки имеют удобный пользовательский интерфейс позволяющий быстро выполнять нужные функции и имеет возможность возврата к предыдущему блоку.

**2 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ**

В данном разделе пояснительной записки детально рассмотримфункционирование программного обеспечения.

При разработки данной программы используется парадигма ООП, соответственно все модули этой программы представляют собой классы, которые связанны друг с другом определённым образом. На схеме ГУИР.400201.105 РР.1 изображена диаграмма классов, показывающая связь между используемыми классами.

В программе должны быть реализованы операции редактирования, добавления, удаления, настройки, поиска информации, необходимо предусмотреть операцию отмены последних действий. В программе используется 3-уровневое наследование.

В программу будет вводиться информация о различном транспорте, которая в дальнейшем будет храниться в соответствующих текстовых файлах, из которых она будет выводиться.

**Текстовой файл** — это файл, компонентами которого являются символьные строки переменной длины, заканчивающиеся специальным маркером конца строки.

Из-за преимуществ (универсальность, устойчивость) текстовых файлов в программе используются именно они.

Следующие таблицы отображают содержание текстовых файлов, использованных для хранения характеристик автомобилей:

Таблица 2.1 – Содержимое текстового файла класса автомобилей из салона

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Модель | Кузов | Макс. скорость | Гарантия | Тест-драйв | Цена |
| BMW | M3 | Седан | 260 | Да | Нет | 30000 |

Таблица 2.2 – Содержание текстового файла класса автомобилей Б/у

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Название | Модель | Кузов | Макс. скорость | Город продажи | Пробег | Цена |
| BMW | E46 | Купе | 250 | Брест | 405000 | 5000 |

Таблица 2.3 – Содержание текстового файла класса микроавтобусов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Макс. скорость | Кол-во мест | Топливо | Расход | Цена |
| Fiat | 160 | 7 | Дизель | 6 | 2000 |

Таблица 2.4 – Содержание текстового файла класса электробусов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название | Макс. скорость | Кол-во мест | Топливо | Потребл. энергии | Цена |
| МАЗ | 90 | 40 | Газ | 250 | 20000 |

**2.1 Файл Class Transport**

В protected секции класса используется тип данных string (название автомобиля), int (цена автомобиля, максимальная скорость автомобиля), Transport является базовым классом.

В public секции класса Transport используются следующие методы:

- метод int getMS() позволяет получить максимальную скорость автомобиля;

- метод string getName() позволяет получить название автомобиля;

- метод int getPrice() позволяет получить стоимость автомобиля;

**2.2 Файл Class Legkovie**

В protected секции класса используется тип данных string (модель, кузов автомобиля).

В public секции класса **legkovie** используются следующие методы:

- метод string getModel() позволяет получить модель автомобиля;

- метод string getKyzov() позволяет получить тип кузова автомобиля;

**2.3 Файл Class Avtobusi**

В protected секции класса используется тип данных int (количество мест) и string(вид топлива).

В public секции класса используются следующие методы:

- метод int getKM() позволяет получить количество мест;

- метод string getToplivo() позволяет получить вид топлива;

**2.4 Файл Class Salon**

Программа представляет собой 2 файла, один заголовочный (salon.h), один основной (salon.cpp). Сама программа использует пользовательский тип данных salon из файла salon.h.

В protected секции класса используется тип данных string (присутствие гарантии, присутствие тест-драйва).

В public секции класса используются следующие методы:

- метод string getGarant() позволяет получить состояние гарантии;

- метод double getTest() позволяет получить состояние тест-драйва;

- метод void setting() позволяет изменять поля объектов класса salon;

- метод void shapka() позволяет выводить шапку над характеристиками при выводе информации;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& in, salon& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& out, salon& obj);

- перегрузка оператора ввода в файл friend ifstream& operator>>(ifstream& in, salon& obj);

- перегрузка оператора вывода из файла friend ofstream& operator<<(ofstream& out, salon& obj);

- перегрузка оператора bool operator==(salon& obj) позволяет определить действия, которые будет выполнять оператор.

**2.5 Файл Class BY**

Программа представляет собой 2 файла, один заголовочный (BY.h), один основной (BY.cpp). Сама программа использует пользовательский тип данных BY из файла BY.h.

В protected секции класса используется тип данных int (пробег) и string(город продажи).

В public секции класса используются следующие методы:

- метод string getGorod() позволяет получить название города продажи;

- метод int getProbeg() позволяет получить стипендию студента;

- метод void setting() позволяет изменять поля объектов класса BY;

- метод void shapka() позволяет выводить шапку над характеристиками при выводе информации;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& in, BY& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& out, BY& obj);

- перегрузка оператора ввода в файл friend ifstream& operator>>(ifstream& in, BY& obj);

- перегрузка оператора вывода из файла friend ofstream& operator<<(ofstream& out, BY& obj);

- перегрузка оператора bool operator==(BY& obj) позволяет определить действия, которые будет выполнять оператор.

**2.6 Файл Class Microavtobus**

Программа представляет собой 2 файла, один заголовочный (microavtobus.h), один основной (microavtobus.cpp). Сама программа использует пользовательский тип данных microavtobus из файла microavtobus.h.

В protected секции класса используется тип данных int (расход топлива).

В public секции класса используются следующие методы:

- метод int getRashod() позволяет получить информацию о расходе топлива;

- метод void setting() позволяет изменять поля объектов класса microavtobus;

- метод void shapka() позволяет выводить шапку над характеристиками при выводе информации;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& in, microavtobus& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& out, microavtobus& obj);

- перегрузка оператора ввода в файл friend ifstream& operator>>(ifstream& in, microavtobus& obj);

- перегрузка оператора вывода из файла friend ofstream& operator<<(ofstream& out, microavtobus& obj);

- перегрузка оператора bool operator==(microavtobus& obj) позволяет определить действия, которые будет выполнять оператор.

**2.7 Файл Class Electrobus**

Программа представляет собой 2 файла, один заголовочный (electrobus.h), один основной (electrobus.cpp). Сама программа использует пользовательский тип данных electrobus из файла electrobus.h.

В protected секции класса используется тип данных int (потребление энергии).

В public секции класса используются следующие методы:

- метод int getPE() позволяет получить информацию о расходе энергии;

- метод void setting() позволяет изменять поля объектов класса electrobus;

- метод void shapka() позволяет выводить шапку над характеристиками при выводе информации;

- перегрузка оператора потокового ввода friend istream& operator>>(istream& in, electrobus& obj);

- перегрузка оператора потокового вывода friend ostream& operator<<(ostream& out, electrobus& obj);

- перегрузка оператора ввода в файлfriend ifstream& operator>>(ifstream& in, electrobus& obj);

- перегрузка оператора вывода из файла friend ofstream& operator<<(ofstream& out, electrobus& obj);

- перегрузка оператора bool operator==(electrobus& obj) позволяет определить действия, которые будет выполнять оператор.

**2.8 Файл Class Shablon<A>**

Программа представляет собой 2 файла, один заголовочный (Shablon.h), один основной (Shablon.cpp).

В private секции класса используется тип данных int (размер двунаправленной очереди), тип данных ohered\* (указатели на первый и последний элемент двунаправленной очереди).

В public секции класса используются следующие методы:

- метод add() добавляет объект;

- метод delete() удаляет объект;

- метод del\_all() удаляет все объекты;

- метод getfirst() возвращает указатель на объект ;

- метод getObj() возвращает объект ;

- метод nastr() позволяет управлять объектами, изменять ,удалять ;

- метод out() выводит объект ;

- метод poisk() по параметру находит объект ;

- метод print\_f(ofstream&) записывает объект в файл ;

- метод readF() считывает объект из файла ;

**2.9 Файл Class Text\_files<A>**

В private секции класса используется тип данных ofstream(объект файлового вывода), ifstream (объект файлового вывода)

**2.10 Файл Class Exeption**

Программа представляет собой 2 файла, один заголовочный (Exeption.h), один основной (Exeption.cpp). Сама программа использует пользовательский тип данных BY из файла Exeption.h.

В private секции класса используется тип данных int (номер ошибки) и string(вид/название ошибки).

Используются следующие методы:

- метод string chek\_str() метод обработки исключения со строкой

- метод int chek\_num() метод обработки исключения с числом

**3 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ**

Пользователь выбирает один из разделов с автомобилями с которыми он хочет работать, где в дальнейшем ему предоставляется не полный функционал программы. В случае, если пользователь авторизуется как администратор, ему будут предоставлены все функции программы. В случае ошибочного ввода каких-либо характеристик, происходит повторный ввод и вывод ошибки на экран.

Важными модулями данной программы являются методы поиска и изменения объектов классов.

При использовании программы могут возникнуть случаи, когда пользователю необходимо исправить какую-то ошибку или изменить значения характеристики одного из объектов класса. Чтобы не переписывать заново все характеристики объекта, необходимо создать метод, который позволит изменять конкретные характеристики, не трогая остальные. Для этого нужно дать пользователю возможность выбрать объект, который он хочет настроить и затем дать ему выбрать нужную характеристику, после чего заменить ее значение на новое. Чтобы дать пользователю выбор, будет использоваться указатель на элементы двунаправленной очереди, а чтобы выбрать нужную характеристику будет использоваться переключатель «switch()».

Для изменения данных вызывается метод «nastr()».

1. Начало

2. Входные данные:

2.1. ochered<А>\* t – указатель на структуру.

2.2. switch(\_getch()); - переключатель, позволяющий выбрать элемент для изменения.

3. Выходные данные:

4. t = t->right; - указатель на следующее значение.

5. t->setting()-вызов метода, который реализован для каждого дочернего класса.

6. Конец

Чтобы дать пользователю возможность быстро найти подходящие по определённым характеристикам объекты класса нужно создать метод поиска, в котором будут приниматься не все поля, а те которые захочет заполнить пользователь. Таким образом метод будет осуществлять сравнение по введённым пользователем характеристикам, а те что он пропустил, будут считаться безразличными и при сравнении не будут учитываться. Найденные объекты класса будут выводиться на экран в виде таблицы с шапкой характеристик, так же как это выглядит при просмотре.

Для реализации метода необходимо использовать указатель на элемент очереди, цикл, который будет выполняться пока указатель не дойдёт до конца очереди. Так же понадобиться перегрузка оператора сравнения двух объектов, которые смогут игнорировать пропущенные поля, т.е. те, в которые пользователь введёт условные значения, предназначенные для пропуска полей.

Из-за того, что каждый класс имеет разные значения необходимо реализовать перегрузку сравнения в каждом дочернем классе.

Для поиска данных в очереди вызывается метод «poisk(A peremennaia)»

1. Начало

2. Входные данные

2.1. A peremennaia– создание объекта класса с которым работает пользователь.

2.2. ochered <А>\* obj = p\_first – указатель на первый элемент двунаправленной очереди.

3. Выходные данные

3.1 object – объект элемента очереди, который будет сравниваться с введенным объектом.

4. Цикл while, который будет выполняться, пока верно условие obj != NULL.

5. cout << obj->object- вывод найденных объектов класса.

6. Конец

**4 РЕЗУЛЬТАТЫ РАБОТЫ**

В результате проделанной работы была создана программа, позволяющая работать с данными о транспорте разных категорий. Данная программа подойдёт для использования компаниями, предоставляющих транспорт в аренду или для маркетплейсов, на которых продают транспорт. Так же эта программа может послужить основой для создания приложения для продажи автомобилей частными лицами.

Далее приведен пример работы программы.

После запуска программы появляется главное меню (рис 5.1). В нем пользователь получает возможность перейти к одной из 4-х категорий и пройти авторизацию от имени администратора.

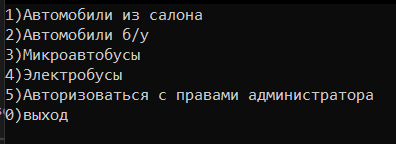


Рисунок 5.1 – Меню программы

Как показано на рисунке 5.1 пользователю предоставляется меню для выбора класса транспорта с которым он желает работать. Что бы получить доступ ко всем функциям, можно авторизоваться как администратор.

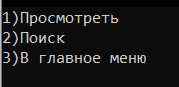


Рисунок 5.2 – Меню функций доступных пользователю

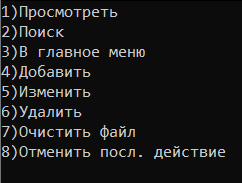


Рисунок 5.3 – Меню функций доступных администратору

Рассмотрим подробнее функционал программы, предоставленный администратору:

1. При нажатии «1» нам будет показан список всего транспорта выбранной категории с соответствующей информацией (рис. 5.4)

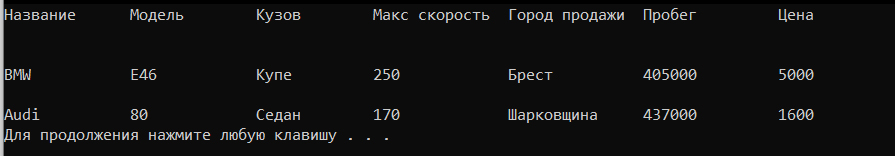


Рисунок 5.4 – Демонстрация транспорта категории

1. При нажатии «2» вы сможете воспользоваться поиском в выбранной категории (рис. 5.5)

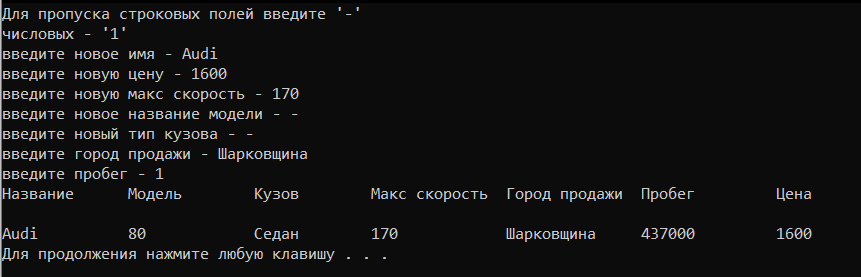


Рисунок 5.5 – Пример выполнения поиска

1. При нажатии «3» вы вернётесь в главное меню и все данные, которые были добавлены или изменены будут сохранены в соответствующем файле.
2. При нажатии «4» вы сможете добавить новый транспорт выбранной категории (рис.5.6)

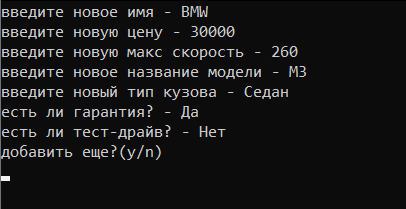


Рисунок 5.6 – Ввод нового автомобиля в категорию

1. При нажатии «5» вы попадёте в меню изменения, где программа предложит вам выбрать объект для изменения его данных, листая элементы очереди. (рис. 5.7)

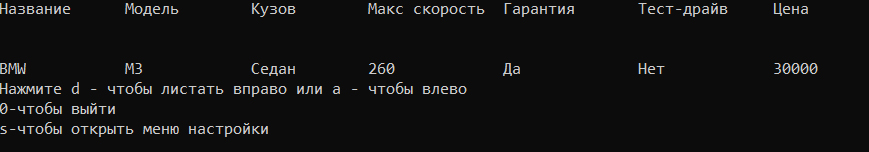


Рисунок 5.7 – Выбор автомобиля для изменения информации о нём

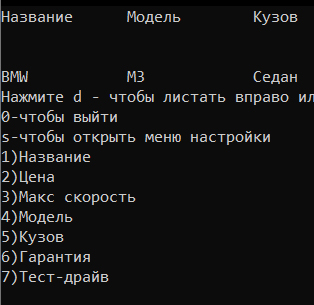


Рисунок 5.8 – Меню изменения информации о выбранном автомобиле

1. При нажатии «6» как и в изменении программа предложит выбрать удаляемый элемент, листая элементы очереди (рис. 5.9)

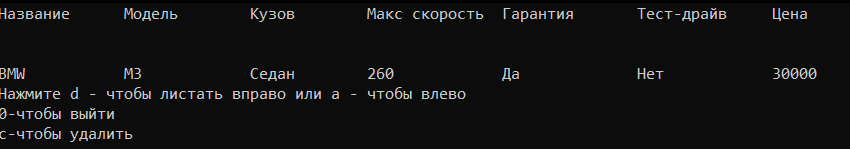


Рисунок 5.9– Удаление элемента из очереди

1. При нажатии «7» вы можете очистить файл (рис. 5.10). В этом случае программа удаляет все элементы очереди в выбранной ранее категории.

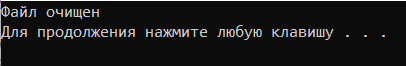


Рисунок 5.10 – Очистка файла

1. При нажатии «8» отменяться все действия, которые были совершены, до сохранения в файл. Осуществляется это удалением всех элементов очереди и повторного чтения старой информации из файла.

Также при авторизации от имени администратора, программа позволяет изменить пароль, применяемый для получения доступа к полному функционалу программы(рис.5.11).

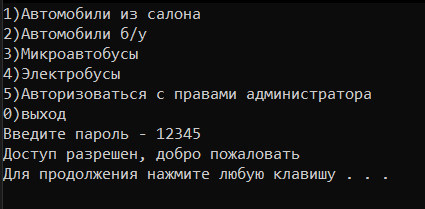


Рисунок 5.11 – Пример авторизации от имени администратора

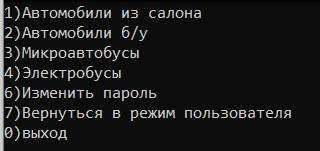


Рисунок 5.12 – Внешний вид главного меню при авторизации от имени администратора

**ЗАКЛЮЧЕНИЕ**

Хотя C++ и описывается как [мультипарадигмальный](https://en.wikipedia.org/wiki/C%2B%2B) язык программирования, на практике большинство программистов используют C++ сугубо как объектно-ориентированный язык (обобщенное программирование используется для «дополнения» ООП).  
 Предполагается, что ООП – это инструмент, одна из многих парадигм, которыми может пользоваться программист для решения задач в коде. Однако, как подсказывает мой опыт, ООП большинством профессионалов принимается за золотой стандарт разработки программного обеспечения. В принципе, разработка решения начинается с определения того, какие объекты нам понадобятся. Решение конкретной задачи начинается уже после того, как код будет распределен по объектам. При переходе к такому объектно-ориентированному мышлению ООП превращается из инструмента в целый инструментарий.

Программа написана в среде разработки Visual studio 2022.

**ЛИТЕРАТУРА**

[1].Р. Лафоре Объектно-ориентированное программирование в С++ 4-е издание 2004г.

[2].Ю.А. Луцик Объектно-ориентированное программирование на языке С++ Минск. БГУИР. 2008г.

[3].Структура программного обеспечения ПК [Электронный ресурс]. – Режим доступа: [https://studopedia.ru/4\_4312\_tema--struktura-programmnogo-obespecheniya-pk.html – Дата доступа 20.10.2019](https://studopedia.ru/4_4312_tema--struktura-programmnogo-obespecheniya-pk.html%20–%20Дата%20доступа%2020.10.2019).

[4].Луцик, Ю. А. Объектно-ориентированное программирование на языке С++: учеб. пособие по курсу «Объектно-ориентированное программирование» для студ. спец. «Вычислительные машины, системы и сети» всех форм обуч. / Ю. А. Луцик, А. М. Ковальчук, И. В. Лукьянова. – Минск: БГУИР, 2003. – 203 с.:ил.

**ПРИЛОЖЕНИЕ А**

*(обязательное)*

Cхема алгоритма

**ПРИЛОЖЕНИЕ Б**

*(обязательное)*

Cхема алгоритма

**ПРИЛОЖЕНИЕ В**

*(обязательное)*

Диаграмма классов

**ПРИЛОЖЕНИЕ Г**

*(обязательное)*

Код программы

**ПРИЛОЖЕНИЕ Д**

*(обязательное)*

Ведомость документов