**Введение**

Базы данных всегда были важнейшей темой при изучении информационных систем. Однако в последние годы всплеск популярности Интернета и бурное развитие новых технологий для Интернета сделали знание технологии баз данных для многих одним из актуальнейших путей карьеры. Технологии баз данных увели Интернет-приложения далеко от простых брошюрных публикаций, которые характеризовали ранние приложения. В то же время Интернет-технология обеспечивает пользователям стандартизированные и доступные средства публикации содержимого баз данных. Правда, ни одна из этих новых разработок не отменяет необходимости в классических приложениях баз данных, которые появились еще до развития Интернета для нужд бизнеса. Это только расширяет важность знания баз данных.

Цель базы данных — помочь людям и организациям вести учет определенных вещей. На первый взгляд, эта цель кажется скромной, и вы, возможно, удивитесь, зачем нам нужна такая сложная технология и целый курс, посвященный этому предмету. Большинство из нас может вспомнить ситуации, в которых нам требуется отслеживать некоторые вещи. На самых ранних стадиях развития информационных технологий использовались списки — набитые на перфокарте и написанные на магнитной ленте. Со временем, однако, стало ясно, что только немногие проблемы можно решить с помощью таких списков. В следующем разделе мы обсудим такие проблемы, а затем опишем, как построить базы данных для их решения.

Процесс подготовки и решения задач на компьютерах состоит из нескольких этапов:

- постановка задачи;

- построение модели;

- разработка алгоритма;

- написание и отладка программы на языке программирования;

- тестирование программы.

При разработке программ применяют метод структурного программирования: разделение задачи на относительно независимые части и выполнение отладки этих частей. При этом достигается качество и быстрота разработки программ.

Развитие вычислительной техники с самого начала было направлено на облегчение и ускорение ввода, поиска данных. Базы данных используются во многих предприятиях, отраслях науки и т.д. Разработано множество готовых языков программирования с помощью которых можно создавать базы данных.

# 1 Разработка эскизного и технического проектов программы

# 1.1 Задание

Цель курсовой работы - написать приложение, позволяющее работать со структурой данных мотосалона содержащей следующие поля: a) порядковый номер б) номер двигателя в) номер рамы г) марка мотоцикла д) тип мотоцикла е) объем мотоцикла ж) мощность мотоцикла з) цена мотоцикла.

Результатом решения являются:

- исполняемый файл программы, полученный в режиме «Release», прилагаемый к пояснительной записке в электронном виде.

- возможность сортировать, добавлять, удалять, сохранять и загружаться данные внутри рабочей программы.

Для реализации файлового ввода-вывода в программу следует включить заголовок <fstream>. В нем определены некоторые классы, в частности, ifstream , ofstream и fstream . Эти классы являются производными от классов istream , ostream и iostream соответственно. Следует помнить, что классы istream , ostream и iostream , в свою очередь, являются производными от класса ios.

Чтение и запись текстовых файлов осуществляются очень легко. Для этого достаточно применить операторы “<<” и как это обычно делается для консольного ввода-вывода, только вместо потоков cin и cout необходимо подставить поток, связанный с файлом. Например, программа «Мотосалон» создает короткий файл, содержащий порядковый номер, номер двигателя, номер рамы, марку мотоцикла, тип мотоцикла, объем, мощность и цену мотоцикла.

# 1.2 Назначение и область применения

Назначение программы – «Мотосалон» позволяет производить ввод записей с информацией о мотоциклах в базу данных, их сортировку и вывод, а также удаление записей из базы данных, сохранение в файл для последующего продолжения работы с ними или передачи другому оператору.

На первом этапе проектирования базы данных необходимо определить цель создания базы данных, основные ее функции и информацию, которую она должна содержать.

База данных разработана для мотосалона «Стронг Моторс», занимающегося продажей, обслуживанием и прокатом мотоциклов. Мотоциклы заказываются непосредственно с заводов-изготовителей и доставляются для клиентов, в ближайшии для них центы. Данная информационная система будет реализована в различных мотосалонах. Пользователем этой системы будет человек, работающий в мотосалоне, а также посетитель, который захочет узнать подробные характеристики мотоцикла через специальный планшет.

# 1.3 Технические характеристики

# 1.3.1 Постановка задачи

Необходимо написать программу, позволяющее работать со структурой данных по закупке чая содержащей следующие поля. Цель: приобретения навыков работы со структурами и базами данных, а также возможной доработкой программы и использования ее в работе с другими базами данных.

Чай имеет следующие параметры: a) порядковый номер б) номер двигателя в) номер рамы г) марка мотоцикла д) тип мотоцикла е) объем мотоцикла ж) мощность мотоцикла з) цена мотоцикла.

Программа выполняет следующие операции:

1. Вывод баз данных в консоли;
2. Добавление параметров мотоцикла в базе данных;
3. Удаление параметров мотоцикла в базе данных;
4. Поиск по параметрам;
5. Сортировка по параметрам.
6. Вывод информации о разработчике программы.

# 

# 1.3.2 Описание алгоритмов

Данная программа состоит из нескольких алгоритмов, ознакомиться с ними и их описанием вы сможете ниже.

1. Программа написанная по алгоритму «Меню» управляется с помощью номера команды (рис.1). Номера команды вводится с клавиатуры и соответствуют следующим действиям:

1 – Вывод баз данных на экран;

2 – Добавление параметров мотоцикла;

3 – поиск по заданному параметру (и при желании удалить параметры найденного мотоцикла);

4 – сортировка по заданному параметру.

5 – вывод информации о разработчике программы.

1. «Алгоритм для выхода в меню» изображен на рис. 2. и работает следующим образом: пока команда не равна 0, функция просит заново ввести номер команды и не выходит в меню.
2. «Алгоритм добавления параметров мотоцикла» изображен на рис. 3. и работает следующим образом:

3.1. Создание нового динамического массива с размером n+nAdd, где n – количество мотоциклов до добавления, nAdd - количество добавляемых мотоциклов;

3.2. Присваивание всех значении старого массива на новый;

3.3. Инициализация параметров добавлемого мотоцикла.

1. «Алгоритм удаления параметров мотоцикла» изображен на рис. 4. и работает следующим образом:
   1. Замена удаляемого элемента массива на последний;
   2. Создание нового динамического массива с размером n-1, где n – количество мотоциклов до удаления;
   3. Инициализация всех старых элементов на новый, кроме последнего.
2. «Алгоритм сортировки» изображен на рис. 5. и сортирует список по заданному параметру (по номеру двигателя, по номеру рамы, по объему, по мощности, по цене);
3. «Алгоритм поиска» изображен на рис. 6. и ищет элемент массива по заданному параметру (по порядковому номеру, номеру двигателя, по номеру рамы, по марке, по типу).

Алгоритмы

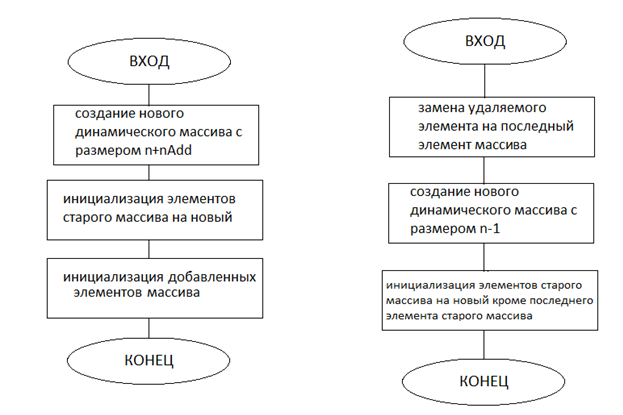
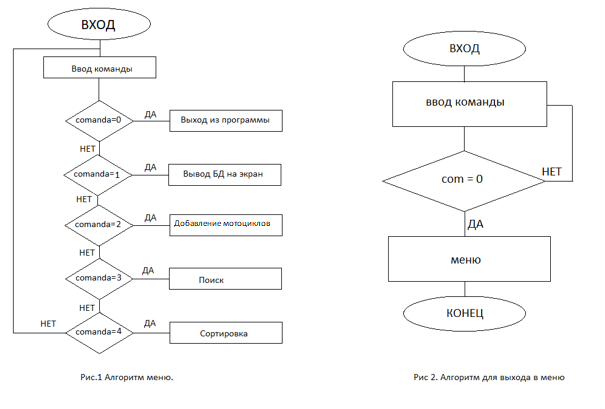


Рис.3. Алгоритм добавления параметров мотоцикла Рис.4. Алгоритм удаления параметров мотоцикла

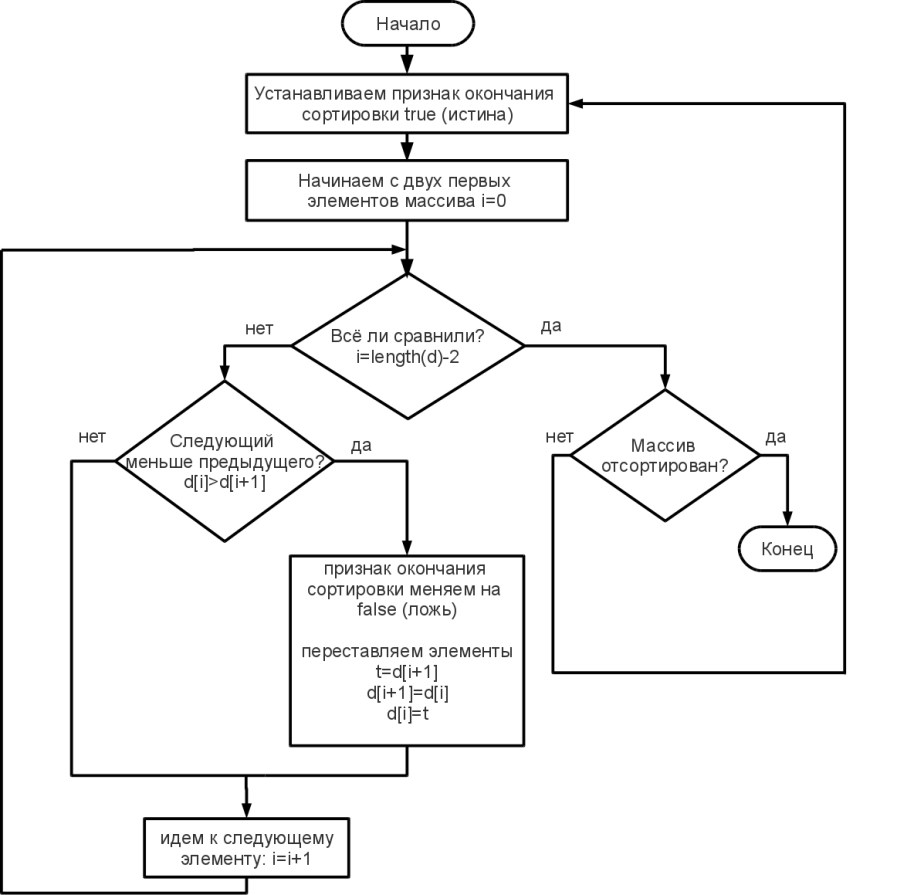


Рис. 5. Алгоритм сортировки

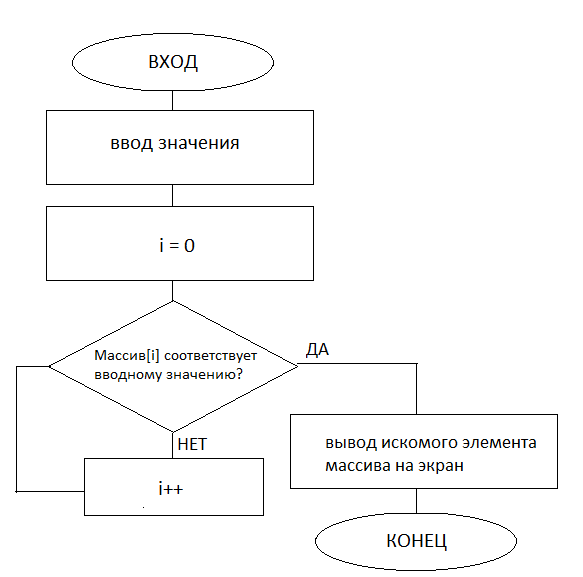


Рис. 6. Алгоритм поиска

# 1.3.3 Организация входных и выходных данных

Входные данные – данные, с которыми работает программа - MotoShow,

которые состоят из:

1. Порядковый номер
2. Номер двигателя
3. Номер рамы
4. Марка
5. Тип
6. Объем

g) Мощность

h) Цена

Для a, f, g, h – используется целочисленные переменные в структуре,

для b,c,d, e – строковые переменные.

struct MotoShow{

int N;

string NumberEngine;

string NumberFrame;

string Marka;

string Type;

int Volume;

int Power;

int Price;

};

Ввод переменных а, f, g, h осуществляется с помощью оператора cin >>, ввод переменных b, c, d, e осуществляется с помощью функции cin.getline().

Выходные данные – результат работы с базой данных представлен целочисленными и строковыми переменными, выводимыми при помощи функций вывода в консоли. Вывод разбавляется текстом, выводимым при помощи cout.

Для работы программы необходимо создать файл с именем «motoshow.db» и в него был записан текст следующим образом: элементы следующего массива должны быть переведены на новую строку и поля структур должны быть разделены символом " ; ", так как этот символ переводит функцию на считываний следующего поля структуры, а перевод на новую строку переводит на считывании нового элемента массива и считает количество. В программе переменная n (количество строк) равна 0, при переводе на новую строку возрастает на единицу.

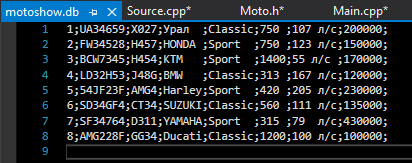


Рис. 7. Результат работы программы «Мотосалон».

Результат работы оператора(Рис. 7): удаление/добавление и сортировка записей в базе данных можно записать в файл motoshow.db.

# 1.3.4 Выбор состава технических и программных средств

Средства, используемые в данной работе, содержатся в интегрированной среде разработки (IDE) Microsoft Visual C++ 2017 (далее VS). Эта среда предоставляет набор инструментов, которые помогают в создании и изменении кода, а также в обнаружении и исправлении ошибок. В VS для организации работы служат проекты и решения. При написании программы на языке C++ с помощью VS первым этапом является выбор типа проекта. Для каждого типа проекта VS устанавливает параметры компилятора и генерирует стартовый код.

# 

# 2 Разработка рабочего проекта

# 2.1 Разработка программы

Программа «Мотосалон» предназначена для функционирования в среде Windows. Это приложение, хотя и разрабатывалось в среде визуального программирования, но использует консольное окно. Таким образом, программирование строится на процессе написания кода, придающего консольному окну и программе в целом необходимую функциональность.

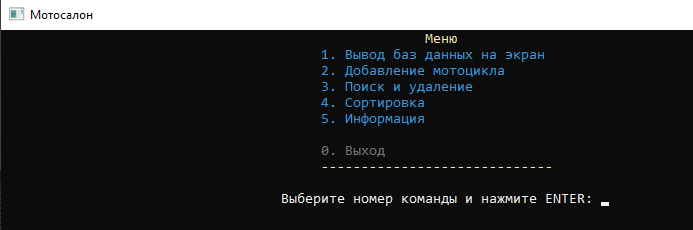


Рис. 8. Меню программы «Мотосалон».

Меню данной программы изображено на рис. 8. и объединяет нескольких функции и их вызывает введенная команда. Если номер команды не существует или вводим буквы в программе, то функция void menu(int n, MotoShow \*moto) вызывает самого себя.

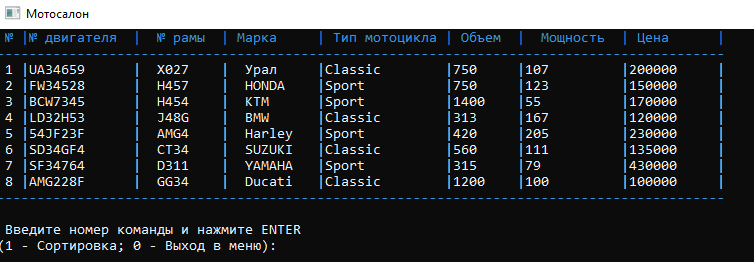


Рис. 9. Вывод баз данных в консоли.

Вывод баз данных в консоли изображен рис. 9 и осуществляется при помощи функции void Draw(int n, MotoShow \*moto) при выборе команд “1” в меню.

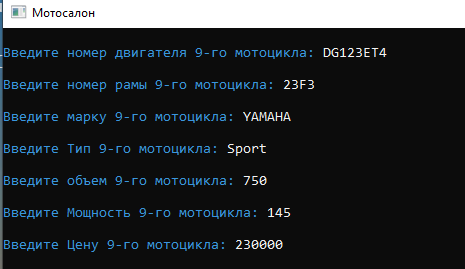


Рис. 10. Заполнение данных добавляемого мотоцикла.

Заполнение данных добавляемого мотоцикла изображено на рис. 10 и осуществляется при помощи функции MotoShow\* AddMoto(int n, MotoShow \*moto) при выборе команд “2” в меню.

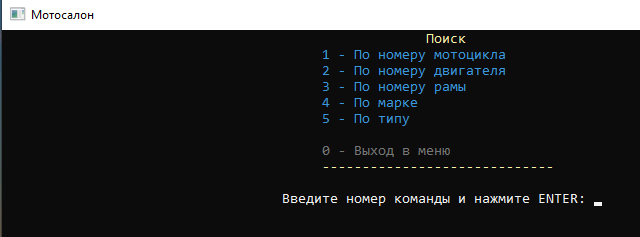


Рис. 11. Поиск.

Подменю поиска изображенo на рис. 11. Поиск чая осуществляется при помощи функции void SearchMoto(int n, MotoShow \*moto) при выборе команд “3” в меню. Поиск осуществляется по следующим параметрам: 1)по порядковому номеру; 2)по номеру двигателя; 3)по номеру рамы; 4)по марке ; 5) по типу.

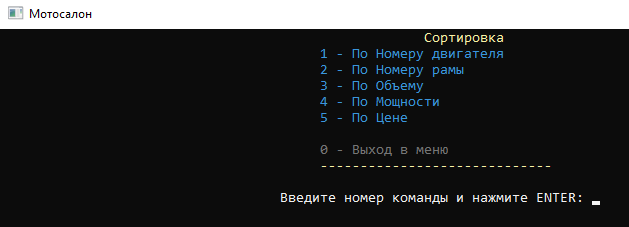


Рис. 12. Сотрировка.

Подменю сортировки изображенo на рис. 12. Сортировка мотоциклов осуществляется при помощи функции MotoShow \*SortMoto(int n, MotoShow \*moto) при выборе команд “4” в меню. В данной функции используется Сортировка вставками ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Insertion sort*). Сортировка осуществляется по следующим параметрам: 1)по номеру двигателя; 2)по номеру рамы; 3)по объему; 4)по мощности; 5)по цене.

При выборе сортировки по объему, мощности или цене программа предлагает выбрать команду сортировки: 1- по возрастанию или 2 – по убыванию.

В данной функции используется Сортировка вставками.

Сортировка вставками ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) *Insertion sort*) — [алгоритм сортировки](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B8), в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. [Вычислительная сложность](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D1%8C) — .

В начальный момент отсортированная последовательность пуста. На каждом шаге алгоритма выбирается один из элементов входных данных и помещается на нужную позицию в уже отсортированной последовательности до тех пор, пока набор входных данных не будет исчерпан. В любой момент времени в отсортированной последовательности элементы удовлетворяют требованиям к выходным данным алгоритма.

Данный алгоритм можно ускорить при помощи использования бинарного поиска для нахождения места текущему элементу в отсортированной части. Проблема с долгим сдвигом массива вправо решается при помощи смены указателей.

# 2.2 Спецификация программы

|  |  |  |
| --- | --- | --- |
| Обозначения | Наименование | Примечание |
| motoshow.exe | Исполняемая программа | Входная точка в программу |
| main.cpp | Исходный код программы | Занимается вызовом основных вычислительных функций |
| menu.cpp | Исходный код программы | Занимается вызовом основных вычислительных функций |
| motoshow.h | Файл включений | Содержит все используемые в программе библиотеки, структуру и некоторые глобальные переменные. |
| motoshow.db | Файл базы данных | Содержит базу данных записанную туда изначально или в ходе работы программы |
|  |  |
| motoshow.sln | Файл проекта | Файл решений проекта (рабочая область проекта), находится в главном каталоге проекта |
| motoshow.iobj | Файл проекта | Файл инкрементальной генерации кода времени соединения LTCG |
| motoshow.ipdb | Файл проекта | Файл инкрементальной генерации кода времени соединения LTCG |
| motoshow.pdb | Файл проекта | Файл инкрементальной генерации кода времени соединения LTCG |
| motoshow.vcxproj | Файл проекта | Файл проекта. Хранит информацию, относящуюся к каждому проекту. (В более ранних версиях этот файл назывался Имя\_проекта.vcproj или Имя\_проекта.dsp |
| motoshow.vcxproj.filters | Файл проекта | Файл фильтров. Указывает, куда поместить файл, который добавляется в решение. Например, H-файл помещается в узле **Файлы заголовков**. |
| motoshow.vcxproj.user | Файл проекта | Файл пользователя миграции. После миграции проекта из Visual Studio 2008 в этом файле появляются данные, преобразованные из любых VSPROPS-файлов. |
| motoshow.suo | Файл проекта | Файл параметров решения. Сохраняет настройки решения, чтобы при любом открытии проекта или файла в решении оно выглядело и вело себя необходимым образом. |

# 

# 2.3 Описание программы

При создании программы «Ведомость по закупкам чая» использовался модульный принцип построения программ. Все исходные модули логически разделены на файлы. Программа состоит из трех файлов: «main.cpp», «menu.cpp», «motoshow.h». Такое разделение позволяет поместить каждую функцию в отдельный .cpp файл что делает код удобным для восприятия и работы.

Заголовочный файл motoshow.h» содержит прототипы всех функций, все подключаемые библиотеки, описание структуры и глобальной переменной n. Далее этот файл подключается ко всем функциям.

Файл «main.срр» - главный, является по своей сути управляющим. Здесь осуществляется вход в программу. Этот модуль содержит настройки консольного окна, локализацию, объявления массивов структур, а так же вызывает функцию menu, находящуюся в файле «menu.cpp»

Программа состоит из следующих функции:

1. main () – точка старта программы.
2. void menu(int n, MotoShow \*moto) – управляется номером команды для использования нужных команд. Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
3. void WriteBeginText() – создает файл и записывает строк. Входной параметр – Отсутствует; Выходной параметр – Отсутствует.
4. void to\_menu(int n, MotoShow \*moto) – предназначена для выхода в меню. Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
5. void gotoxy(int x, int y) – выравнивает баз данных в консоли. Входной параметр – Задающие координаты для выравнивания БД в консоли; Выходной параметр – Отсутствует.
6. int number() – считает количество строк из файла. Входной параметр – Отсутствует; Выходной параметр – Количество продуктов.
7. MotoShow \*Initialization(int n, MotoShow \*moto) – инициализирует массив структуры данных. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
8. void NewMotoRecordFile(int n, MotoShow \*moto) – записывает в файл изменения. Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
9. MotoShow \*Detete(int n, MotoShow \*moto, int comanda\_del) – удаляет comanda\_del-го элемента массива Входной параметр – Массив структуры данных и команда для удаления элемента; Выходной параметр – Отсутствует.
10. MotoShow \*AddMoto(int n, MotoShow \*moto) – предназначена для добавления нужного количество элементов массива структуры данных. Входной параметр – Массив структуры данных. Выходной параметр – Отсутствует.
11. void Draw(int n, MotoShow \*moto) – выводит баз данных в консоли. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
12. void SearchMoto(int n, MotoShow \*moto) – предназначена для поиска элемента массива(и при желаний для дальнейшего удаления элемента массива). Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
13. void SortStr(string \*S, int N, MotoShow \*moto) – сортирует структуру по алфавиту по заданным параметрам. Входные параметры – 1.Массив структуры данных; 2. Массив строк; Выходной параметр - Отсутствует.
14. MotoShow \*SortMoto(int n, MotoShow \*moto) – сортирует структуру по алфавиту/значению по заданным параметрам. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.
15. void information(int n, MotoShow \*moto) - выводит информацию о разработчике программы. Входной параметр – Массив структуры данных; Выходной параметр – Отсутствует.

**2.4 Тестирование программы**

Тестирование (testing) программного обеспечения (ПО) - это процесс исследования ПО с целью выявления ошибок и определения соответствия между реальным и ожидаемым поведением ПО, осуществляемый на основе набора тестов, выбранных определённым образом. В более широком смысле, тестирование ПО - это техника контроля качества программного продукта, включающая в себя проектирование тестов, выполнение тестирования и анализ полученных результатов.

# 3 Внедрение

Программа «Мотосалон» - консольное приложение, занимает объем не более 38 КБ. Для ее выполнения необходим минимальный состав аппаратурных средств: операционная система Windows 7, память 4 Мбайт, память видеокарты 512 Кбайт, жесткий диск 544 Мбайт, экран, клавиатура, мышь.

Для работы программы необходимо создать файл с именем «motoshow.db» и в него был записан текст следующим образом: элементы следующего массива должны быть переведены на новую строку и поля структур должны быть разделены символом “;”, так как этот символ переводит функцию на считывание следующего поля структуры, а перевод на новую строку переводит на считывании нового элемента массива и считает количество. В программе переменная сначала n=0, при переводе на новую строку возрастает на единицу больше.

При наличии файла «motoshow.db» программа вызывает функцию void menu(int n, MotoShow \*moto), которая объединяет нескольких функции и загружает соответствующего массива структур. Функция void menu(int n, MotoShow \*moto) позволяет удалить и добавить элементы при помощи следующей функции MotoShow \*AddMoto(int n, MotoShow \*moto) и MotoShow \*Detete(int n, MotoShow \*moto, int comanda\_del). comanda\_del – индекс массива который сначала заменяется на последний элемент массива, потом удаляется.

При добавлении данный программа пишет следующее сообщение " Данные добавленного мотоцикла сохранены", а при удалении – "Данные удалены успешно".

После выполнения любой команды программа позволяет вернуть в главное меню с помощью функции void to\_menu(int n, MotoShow \*moto), которая выводит следующее сообщение "Для выхода в меню введите "0" и нажмите ENTER: ".

# 

# Список использованной литературы

1. Архангельский, А.Я. Язык C++ в C++ Builder : справочное и методическое пособие / А.Я. Архангельский .-- М. : Бином, 2017 – 942 с.

1. Бьерн Страуструп – Язык программирования C++. Специальное издание, 2017

3. Дейтел, Х.М.; Дейтел, П.Дж. Как программировать на С++; М.: Бином; Издание 4-е, 2015. **- 390** c

1. Саттер, Герб Новые сложные задачи на C++; Вильямс, 2016. - 272 c.
2. Кубенский, А.А. Структуры и алгоритмы обработки данных: объектно- ориентированный подход и реализация на С++ / А.А. Кубенский,- СПб.: БХВ- Петербург, 2004- 464 с.
3. Спольски Дж. Джоэл: и снова о программировании: Пер. с англ. — СПб.: Символ-Плюс, 2015. — 320 с.
4. А. Мешков, Ю. Тихомиров «Visual C++ и MFC» - СПб.: БХВ-Петербург. 2016 — 1017с.

# Приложение

# Листинг программы

# «База данных Мотосалон»

**================Заголовочный файл Moto.h====================**

#ifndef Moto

#define Moto

#include<stdio.h>

#include<conio.h>

#include<string>

#include<iostream>

#include<fstream>

#include<windows.h>

using namespace std;

struct MotoShow

{

int N;//порядковый номер

string NumberEngine;//номер двигателя

string NumberFrame;//номер рамы

string Marka;//марка мотоцикла

string Type;//тип мотоцикла

int Volume;//объем

int Power;//мощность

int Price;//цена

};

void menu(int n, MotoShow\* moto);

void WriteBeginText();

void to\_menu(int n, MotoShow\* moto);

void gotoxy(int x, int y);

void NewMotoRecordFile(int n, MotoShow\* moto);

int number();

MotoShow\* Initialization(int n, MotoShow\* moto);

MotoShow\* Detete(int n, MotoShow\* moto, int comanda\_del);

MotoShow\* AddMoto(int n, MotoShow\* moto);

void SortStr(string\* S, int N, MotoShow\* moto);

MotoShow\* SortMoto(int n, MotoShow\* moto);

void Draw(int n, MotoShow\* moto);

void SearchMoto(int n, MotoShow\* moto);

void menu(int n, MotoShow\* moto);

void information(int n, MotoShow\* moto);

#endif

**==================Файл исходного кода Main.cpp================**

#include "Motor.h"

int n = 0; // Количество мотоциклов

string FailName = "motoshow.db"; // Путь к файлу и/или имя файла

int main()

{

system("title Мотосалон"); //Заголовок

setlocale(LC\_ALL, "ru");

WriteBeginText();

ifstream file; // Чтение файла

file.open(FailName); // Открытие файла

if (file.is\_open() == true) // Если файл открыт можно работать программой

{

file.close(); // Закрытие файла

n = number();

MotoShow\* moto = new MotoShow[n];

Initialization(n, moto);

menu(n, moto);

delete[] moto; //удаление динамической памяти

\_getch(); // Задержка программы

}

else

{

cout << "\t ------------------" << "\n\t | Файл не открыт | \n" << "\t ------------------";

Sleep(10000); // Если файл не открывается программа выключится через 10000 мкс(т. е. 10 секунд).

}

return 0;

}

==================**Файл исходного кода Menu.cpp===============**

#include "Moto.h"

string FileName = "motoshow.db"; // Путь к файлу и/или имя файла

HDC hdc = GetDC(GetConsoleWindow());

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE); //получаем дескриптор

//Вывод меню, выбор действия и перенаправление на функции в зависимости от выбора+

void menu(int n, struct MotoShow\* moto)

{ /\*

Данная функция управляется номером команды для использования нужных команд

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 14); // Желтый

cout << "\t\t\t\t\t\t Меню\n";

SetConsoleTextAttribute(hStdOut, 3); // Циан

cout << "\t\t\t\t\t1. Вывод баз данных на экран\n";

cout << "\t\t\t\t\t2. Добавление мотоцикла\n";

cout << "\t\t\t\t\t3. Поиск и удаление\n";

cout << "\t\t\t\t\t4. Сортировка\n";

cout << "\t\t\t\t\t5. Информация\n";

SetConsoleTextAttribute(hStdOut, 8); // Серый

cout << "\n\t\t\t\t\t0. Выход\n";

SetConsoleTextAttribute(hStdOut, 14); // Желтый

cout << "\t\t\t\t\t-----------------------------\n";

int comanda;

SetConsoleTextAttribute(hStdOut, 15); // белый цвет

cout << "\n\t\t\t\t Выберите номер команды и нажмите ENTER: ";

while (!(cin >> comanda))

{

cin.clear();

while (cin.get() != '\n');

menu(n, moto);

}

switch (comanda)

{

case 0: exit(0); //выход

case 1: Draw(n, moto); break; //1 - вывод базы данных на экран(menu)

case 2: AddMoto(n, moto); break; //2 - добавление мотоцикла(menu)

case 3: SearchMoto(n, moto); break; //3 - поиск и удаление(menu)

case 4: SortMoto(n, moto); break; //4 - сортировка(menu)

//case 5: information(n, moto); break; //5 - информация

default: menu(n, moto);

}

}

//1 - вывод базы данных на экран(menu)+

void Draw(int n, struct MotoShow\* moto)

{

/\*

Данная функция выводит БД в консоли

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 3);

gotoxy(1, 0);

cout << "№ |";

gotoxy(4, 0);

cout << "№ двигателя |";

gotoxy(20, 0);

cout << "№ рамы |";

gotoxy(30, 0);

cout << "Марка |";

gotoxy(42, 0);

cout << "Тип мотоцикла |";

gotoxy(58,0);

cout << "Объем |" ;

gotoxy(68, 0);

cout << "Мощность |";

gotoxy(80, 0);

cout << "Цена |";

cout << "\n-------------------------------------------------------------------------------------------";

SetConsoleTextAttribute(hStdOut, 15);

for (int i = 0; i < n; i++)

{

gotoxy(1, i + 2);

cout << i + 1;

gotoxy(4, i + 2);

cout << moto[i].NumberEngine;

gotoxy(20, i + 2);

cout << moto[i].NumberFrame;

gotoxy(31, i + 2);

cout << moto[i].Marka;

gotoxy(41, i + 2);

cout << moto[i].Type;

gotoxy(57, i + 2);

cout << moto[i].Volume;

gotoxy(66, i + 2);

cout << moto[i].Power;

gotoxy(79, i + 2);

cout << moto[i].Price;

gotoxy(3, i + 2);

cout << "|";

gotoxy(17, i + 2);

cout << "|";

gotoxy(28, i + 2);

cout << "|";

gotoxy(40, i + 2);

cout << "|";

gotoxy(56, i + 2);

cout << "|";

gotoxy(65, i + 2);

cout << "|";

gotoxy(78, i + 2);

cout << "|";

gotoxy(90, i + 2);

cout << "|";

}

SetConsoleTextAttribute(hStdOut, 3);

cout << "\n-------------------------------------------------------------------------------------------\n";

SetConsoleTextAttribute(hStdOut, 15);

int com;

cout << "\n Введите номер команды и нажмите ENTER \n(1 - Сортировка; 0 - Выход в меню): ";

while (!(cin >> com))

{

cin.clear();

while (cin.get() != '\n');

Draw(n, moto);

}

switch (com)

{

case 0: menu(n, moto); break;

case 1: SortMoto(n, moto); break;

default: Draw(n, moto);

}

}

//Данные для Базы Данных+

void WriteBeginText()

{

/\*

Данная функция создает файл и записывает строк

Входной параметр - Отсутствует;

Выходной параметр - Отсутствует.

\*/

ofstream fout;

fout.open(FileName);

fout << "1;UA34659;X027;Урал ;Classic;750 ;107 л/с;200000;" << endl;

fout << "2;FW34528;H457;HONDA ;Sport ;750 ;123 л/с;150000;" << endl;

fout << "3;BCW7345;H454;KTM ;Sport ;1400;55 л/с ;170000;" << endl;

fout << "4;LD32H53;J48G;BMW ;Classic;313 ;167 л/с;120000;" << endl;

fout << "5;54JF23F;AMG4;Harley;Sport ;420 ;205 л/с;230000;" << endl;

fout << "6;SD34GF4;CT34;SUZUKI;Classic;560 ;111 л/с;135000;" << endl;

fout << "7;SF34764;D311;YAMAHA;Sport ;315 ;79 л/с;430000;" << endl;

fout << "8;AMG228F;GG34;Ducati;Classic;1200;100 л/с;100000;" << endl;

fout.close();

}

//выход в главное меню и проверка на ввод(нет ссылок)+

void to\_menu(int n, struct MotoShow\* moto)

{

/\*

Данная функция предназначена для выхода в меню

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

SetConsoleTextAttribute(hStdOut, 15);

int comanda;

cout << "\n\t\t\tДля выхода в меню введите \"0\" и нажмите ENTER: ";

while (!(cin >> comanda) || comanda != 0)

{

cin.clear(); //Очишает ошибочно введенной команды

while (cin.get() != '\n');

cout << "\n\t\t\tДля выхода в меню введите \"0\" и нажмите ENTER: ";

}

menu(n, moto);

}

//выравнивание базы данных(нет ссылок)+

void gotoxy(int x, int y)

{

/\*

Данная функция выравнивает БД в консоли

Входной параметр - Задающие координаты для выравнивания БД в консоли;

Выходной параметр - Отсутствует.

\*/

COORD p = { x, y };

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE), p);

}

//Считывание кол-ва строк из файла+

int number()

{

/\*

Данная функция считает количество строк из файла

Входной параметр - Отсутствует;

Выходной параметр - Количество продуктов.

\*/

int count = 0;

ifstream file;

file.open(FileName);

while (!file.eof())

{

if (file.get() == '\n')

count++;

}

MotoShow\* Pr = new MotoShow[count];

return count;

}

//Инициализация массивоф данными+

MotoShow\* Initialization(int n, struct MotoShow\* moto)

{

/\*

Данная функция инициализирует массив структуры данных

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

ifstream fin;

fin.open(FileName);

char N1[3], Engine1[20], Frame1[20], Marka1[20], Type1[20], Volume1[20], Power1[20], Price1[20];

//---------Инициализация масивов символов к нулевому символу--------------------

for (int i = 0; i < 3; i++) {

N1[i] = '\0';

}

for (int i = 0; i < 20; i++) {

Engine1[i] = Frame1[i] = Marka1[i] = Type1[i] = '\0';

}

for (int i = 0; i < 20; i++) {

Volume1[i] = Power1[i] = Price1[i] = '\0';

}

//------------------------------------------------------------------------------

for (int i = 0; i < n; i++)

{

int count = 0;

char ch;

while ((ch = fin.get()) != ';')

{

N1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Engine1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Frame1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Marka1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Type1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Volume1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Power1[count] = ch; count++;

}

count = 0;

while ((ch = fin.get()) != ';')

{

Price1[count] = ch; count++;

}

count = 0;

moto[i].N = atoi(N1);

moto[i].NumberEngine = Engine1;

moto[i].NumberFrame = Frame1;

moto[i].Marka = Marka1;

moto[i].Type = Type1;

moto[i].Volume = atoi(Volume1);

moto[i].Power = atoi(Power1);

moto[i].Price = atoi(Price1);

}

fin.close();

return NULL;

}

//2 - добавление мотоциклов(menu)+

MotoShow\* AddMoto(int n, struct MotoShow\* moto)

{

/\*

Данная функция предназначена для добавления нужного количество элементов массива структуры данных

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

int nAdd;

SetConsoleTextAttribute(hStdOut, 3);

cout << "Введите количество добавляемых мотоциклов: ";

while (!(cin >> nAdd))

{

cin.clear();

while (cin.get() != '\n');

cout << endl << "\t\tОШИБКА!\nВведите заново количество добавленных мотоциклов: ";

}

MotoShow\* MotoNew = new MotoShow[n + nAdd];

for (int i(0); i < n; i++)

{

MotoNew[i] = moto[i];

}

system("cls");

delete[] moto;

moto = MotoNew;

for (int i = n; i < (n + nAdd); i++)

{

char engine[40];

char frame[40];

char marka[40];

char type[40];

int volume[40];

int power[40];

int price[40];

char f[48];

SetConsoleCP(1251); //Перевод в кодировку

SetConsoleOutputCP(1251);

//==============================

SetConsoleTextAttribute(hStdOut, 3);

cout << endl << "Введите номер двигателя " << i + 1 << "-го мотоцикла: ";

SetConsoleTextAttribute(hStdOut, 15);

cin.getline(f, 48); //Для задержки

cin.getline(engine, 40);

moto[i].NumberEngine = (string)engine;

//==============================

SetConsoleTextAttribute(hStdOut, 3);

cout << endl << "Введите номер рамы " << i + 1 << "-го мотоцикла: ";

SetConsoleTextAttribute(hStdOut, 15);

cin.getline(frame, 40);

moto[i].NumberFrame = (string)frame;

//==============================

SetConsoleTextAttribute(hStdOut, 3);

cout << endl << "Введите марку " << i + 1 << "-го мотоцикла: ";

SetConsoleTextAttribute(hStdOut, 15);

cin.getline(marka, 40);

moto[i].Marka = (string)marka;

//==============================

SetConsoleTextAttribute(hStdOut, 3);

cout << endl << "Введите Тип " << i + 1 << "-го мотоцикла: ";

SetConsoleTextAttribute(hStdOut, 15);

cin.getline(type, 40);

moto[i].Type = (string)type;

//==============================

SetConsoleTextAttribute(hStdOut, 3);

cout << endl << "Введите объем " << i + 1 << "-го мотоцикла: ";

SetConsoleTextAttribute(hStdOut, 15);

cin >> moto[i].Volume;

//==============================

SetConsoleTextAttribute(hStdOut, 3);

cout << endl << "Введите Мощность " << i + 1 << "-го мотоцикла: ";

SetConsoleTextAttribute(hStdOut, 15);

cin >> moto[i].Power;

//==============================

SetConsoleTextAttribute(hStdOut, 3);

cout << endl << "Введите Цену " << i + 1 << "-го мотоцикла: ";

SetConsoleTextAttribute(hStdOut, 15);

cin >> moto[i].Price;

SetConsoleTextAttribute(hStdOut, 3);

system("cls");

}

ofstream fout;

fout.open(FileName, ios\_base::app);

for (int i = n; i < (n + nAdd); i++)

{

fout << i + 1 << ";" << MotoNew[i].NumberEngine << ";" << MotoNew[i].NumberFrame << ";" << MotoNew[i].Marka;

fout << ";" << MotoNew[i].Type << ";" << MotoNew[i].Volume << ";" << MotoNew[i].Power << ";" << MotoNew[i].Price << ";\n";

}

fout.close();

n += nAdd;

SetConsoleTextAttribute(hStdOut, 1);

cout << "\n\n\t\t\t ------------------------------------------";

SetConsoleTextAttribute(hStdOut, 8);

cout << "\n\t\t\t | Данные добавленных мотоциклов сохранены | \n";

SetConsoleTextAttribute(hStdOut, 1);

cout << "\t\t\t ------------------------------------------\n";

to\_menu(n, moto);

return NULL;

}

//3 - поиск и удаление мотоцикла(menu)+

void SearchMoto(int n, struct MotoShow\* moto)

{

/\*

Данная функция предназначена для поиска элемента массива(и при желаний для дальнейшего удаления элемента массива)

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

SetConsoleCP(1251); //Перевод в кодировку

SetConsoleOutputCP(1251);

system("cls");

SetConsoleTextAttribute(hStdOut, 14);

cout << "\t\t\t\t\t\t Поиск\n";

SetConsoleTextAttribute(hStdOut, 3);

cout << "\t\t\t\t\t1 - По номеру мотоцикла\n";

cout << "\t\t\t\t\t2 - По номеру двигателя\n";

cout << "\t\t\t\t\t3 - По номеру рамы\n";

cout << "\t\t\t\t\t4 - По марке\n";

cout << "\t\t\t\t\t5 - По типу\n";

SetConsoleTextAttribute(hStdOut, 8);

cout << "\n\t\t\t\t\t0 - Выход в меню\n";

SetConsoleTextAttribute(hStdOut, 14);

cout << "\t\t\t\t\t-----------------------------\n";

int comanda;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

while (!(cin >> comanda))

{

cin.clear();

while (cin.get() != '\n');

SearchMoto(n, moto);

}

switch (comanda)

{

case 1:

{

system("cls");

int No, No1;

cout << "\n\tВведите № искомого мотоцикла (1 - " << n << "): ";

while (!(cin >> No1) || No1 > n || No1 < 1)

{

cin.clear();

while (cin.get() != '\n');

system("cls");

cout << "Вы ввели неправильно порядковый номера мотоцикла\n";

cout << "\n\tВведите заново № искомого мотоцикла (1 - " << n << "): ";

}

No = No1 - 1;

system("cls");

cout << "\t№ ";

SetConsoleTextAttribute(hStdOut, 10);

cout << No1;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\tНомер двигателя: " << moto[No].NumberEngine << "\n\tНомер рамы: " << moto[No].NumberFrame << "\n\tМарка: " << moto[No].Marka << "\n\tТип: " << moto[No].Type;

cout << "\n\tОбъем: " << moto[No].Volume << "\n\tМощность: " << moto[No].Power << "\n\tЦена: " << moto[No].Price << endl << endl;

cout << "\n\t\t\t\tКоманды\n\t\t1 - Удаление данного элемента\n\t\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, moto); break;

case 1: Detete(n, moto, No); break;

}

};

break;

case 2:

{

system("cls");

string code;

int ComDel = 0;

cout << "\n\tВведите номер двигателя: ";

cin >> code;

int checking = 0;

for (int i(0); i < n; i++)

{

if (code == moto[i].NumberEngine)

{

cout << "\t№ " << (i + 1);

cout << "\n\tНомер двигателя: ";

SetConsoleTextAttribute(hStdOut, 10);

cout << moto[i].NumberEngine;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\tНомер рамы: " << moto[i].NumberFrame << "\n\tМарка: " << moto[i].Marka << "\n\tТип: " << moto[i].Type;

cout << "\n\tОбъем: " << moto[i].Volume << "\n\tМощность: " << moto[i].Power << "\n\tЦена: " << moto[i].Price << endl << endl;

ComDel = i;

checking++;

}

}

if (checking == 0)

{

system("cls");

cout << "\n\t Номер двигателя не найден\n";

to\_menu(n, moto);

}

if (checking > 1)

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

if (com == 0) {

menu(n, moto);

}

if (com == 1) {

cout << "\n Введите номер удаляемого элемента: ";

int c; cin >> c;

ComDel = c - 1;

Detete(n, moto, ComDel);

}

}

else

{

cout << "\t\t\t\tКоманды\n\t\t1 - Удаление данного элемента\n\t\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, moto); break;

case 1: Detete(n, moto, ComDel); break;

}

}

};

break;

case 3:

{

system("cls");

string LifePr;

int checking = 0;

int ComDel = 0;

cout << "\n\tВведите номер рамы: ";

while (!(cin >> LifePr))

{

cin.clear();

while (cin.get() != '\n');

system("cls");

cout << "\n\tВведите номер рамыь: ";

}

for (int i(0); i < n; i++)

{

if (LifePr == moto[i].NumberFrame)

{

cout << "\n\tМарка: " << moto[i].Marka << "\n\tТип: " << moto[i].Type;

cout << "\n\tОбъем: " << moto[i].Volume << "\n\tМощность: " << moto[i].Power << "\n\tЦена: " << moto[i].Price << "\n\tНомер рамы: ";

SetConsoleTextAttribute(hStdOut, 10);

cout << moto[i].NumberFrame;

SetConsoleTextAttribute(hStdOut, 15);

ComDel = i;

checking++;

}

}

if (checking == 0)

{

system("cls");

cout << "\n\tНомер рамы не найден\n";

to\_menu(n, moto);

}

if (checking > 1)

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

if (com == 0) {

menu(n, moto);

}

if (com == 1) {

cout << "\n Введите номер удаляемого элемента: ";

int c; cin >> c;

ComDel = c - 1;

Detete(n, moto, ComDel);

}

}

else

{

cout << "\t\t\t\tКоманды\n\t\t1 - Удаление данного элемента\n\t\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, moto); break;

case 1: Detete(n, moto, ComDel); break;

}

}

};

break;

case 4:

{

system("cls");

string Numb;

int ComDel = 0;

cout << "\n\tВведите марку: ";

cin >> Numb;

int checking = 0;

for (int i(0); i < n; i++)

{

if (Numb == moto[i].Marka)

{

cout << "\t№ " << (i + 1);

cout << "\n\tМарка: ";

SetConsoleTextAttribute(hStdOut, 10);

cout << moto[i].Marka;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\tНомер двигателя: " << moto[i].NumberEngine << "\n\tНомер рамы: " << moto[i].NumberFrame << "\n\tТип: " << moto[i].Type;

cout << "\n\tОбъем: " << moto[i].Volume << "\n\tМощность: " << moto[i].Power << "\n\tЦена: " << moto[i].Price << endl << endl;

ComDel = i;

checking++;

}

}

if (checking == 0)

{

system("cls");

cout << "\n\t Марка мотоцикла не найдена\n";

to\_menu(n, moto);

}

if (checking > 1)

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

if (com == 0) {

menu(n, moto);

}

if (com == 1) {

cout << "\n Введите номер удаляемого элемента: ";

int c; cin >> c;

ComDel = c - 1;

Detete(n, moto, ComDel);

}

}

else

{

cout << "\t\t\t\tКоманды\n\t\t1 - Удаление данного элемента\n\t\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, moto); break;

case 1: Detete(n, moto, ComDel); break;

}

}

}

break;

case 5:

{

system("cls");

string Number;

int ComDel = 0;

cout << "\n\tВведите тип мотоцикла: ";

cin >> Number;

int checking = 0;

for (int i(0); i < n; i++)

{

if (Number == moto[i].Marka)

{

cout << "\t№ " << (i + 1);

cout << "\n\tТип: ";

SetConsoleTextAttribute(hStdOut, 10);

cout << moto[i].Type;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\tНомер двигателя: " << moto[i].NumberEngine << "\n\tНомер рамы: " << moto[i].NumberFrame << "\n\tМарка: " << moto[i].Marka;

cout << "\n\tОбъем: " << moto[i].Volume << "\n\tМощность: " << moto[i].Power << "\n\tЦена: " << moto[i].Price << endl << endl;

ComDel = i;

checking++;

}

}

if (checking == 0)

{

system("cls");

cout << "\n\t Тип мотоцикла не найдена\n";

to\_menu(n, moto);

}

if (checking > 1)

{

cout << "\n\t\tКоманды\n\t1 - Удаление элемента\n\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

if (com == 0) {

menu(n, moto);

}

if (com == 1) {

cout << "\n Введите номер удаляемого элемента: ";

int c; cin >> c;

ComDel = c - 1;

Detete(n, moto, ComDel);

}

}

else

{

cout << "\t\t\t\tКоманды\n\t\t1 - Удаление данного элемента\n\t\t0 - Выход в меню";

int com;

cout << "\n\n Введите номер команды и нажмите ENTER: ";

while (!(cin >> com) || com > 1 || com < 0)

{

cin.clear();

while (cin.get() != '\n');

cout << "\n Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 0: menu(n, moto); break;

case 1: Detete(n, moto, ComDel); break;

}

}

}

break;

case 0: menu(n, moto);

default: SearchMoto(n, moto);

}

}

//Добавление мотоцикла +

void NewMotoRecordFile(int n, MotoShow\* moto)

{

/\*

Данная функция записывает в файл изменения

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

ofstream fout;

fout.open(FileName);

for (int i = 0; i < (n - 1); i++)

{

fout << i + 1 << ";" << moto[i].NumberEngine << ";" << moto[i].NumberFrame << ";" << moto[i].Marka;

fout << ";" << moto[i].Type << ";" << moto[i].Volume << ";" << moto[i].Power << ";" << moto[i].Price << ";\n";

}

fout.close();

}

//Удаление мотоцикла+

MotoShow\* Detete(int n, MotoShow\* moto, int comanda\_del)

{

/\*

Данная функция удаляет comanda\_del-го элемента массива

Входной параметр - Массив структуры данных и команда для удаления элемента;

Выходной параметр - Отсутствует.

\*/

//------------------УДАЛЕНИЕ ЭЛЕМЕНТА-----------------------------------------------------------------------

//------- Шаг 1. Замена удаляемого элемента на крайний элемент

string temp\_Engine = moto[n - 1].NumberEngine;

moto[n - 1].NumberEngine = moto[comanda\_del].NumberEngine;

moto[comanda\_del].NumberEngine = temp\_Engine;

string name\_Frame = moto[n - 1].NumberFrame;

moto[n - 1].NumberFrame = moto[comanda\_del].NumberFrame;

moto[comanda\_del].NumberFrame = name\_Frame;

string temp\_Marka = moto[n - 1].Marka;

moto[n - 1].Marka = moto[comanda\_del].Marka;

moto[comanda\_del].Marka = temp\_Marka;

string temp\_Type = moto[n - 1].Type;

moto[n - 1].Type = moto[comanda\_del].Type;

moto[comanda\_del].Type = temp\_Type;

int temp\_Volume = moto[n - 1].Volume;

moto[n - 1].Volume = moto[comanda\_del].Volume;

moto[comanda\_del].Volume = temp\_Volume;

int temp\_Power = moto[n - 1].Power;

moto[n - 1].Power = moto[comanda\_del].Power;

moto[comanda\_del].Power = temp\_Power;

int temp\_Price = moto[n - 1].Price;

moto[n - 1].Price = moto[comanda\_del].Price;

moto[comanda\_del].Price = temp\_Price;

//-------- Шаг 2. Новый масив

MotoShow\* PrNew = new MotoShow[n - 1];

//------- Шаг 3. Инициализация всех элементов к новому массиву кроме крайного

for (int i(0); i < (n - 1); i++)

{

PrNew[i] = moto[i];

}

//------ Шаг 4. Удаление старого массива

delete[] moto;

moto = PrNew;

//-------------------------------------

//---------------Запись в файл ---------------------------------

NewMotoRecordFile(n, moto);

//---------------------------------------------------------------

system("cls");

cout << "\n\n\n\n\t\tДанные удалены успешно.\n";

n--;

to\_menu(n, moto);

return NULL;

}

//4 - сортировка мотоциклов(menu)

MotoShow\* SortMoto(int n, struct MotoShow\* moto)

{

/\*

Данная функция сортирует структуру по алфавиту/значению по заданным параметрам

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 14);

cout << "\t\t\t\t\t\t Сортировка\n";

SetConsoleTextAttribute(hStdOut, 3);

cout << "\t\t\t\t\t1 - По Номеру двигателя \n";

cout << "\t\t\t\t\t2 - По Номеру рамы\n";

cout << "\t\t\t\t\t3 - По Объему\n";

cout << "\t\t\t\t\t4 - По Мощности\n";

cout << "\t\t\t\t\t5 - По Цене\n";

SetConsoleTextAttribute(hStdOut, 8);

cout << "\n\t\t\t\t\t0 - Выход в меню\n";

SetConsoleTextAttribute(hStdOut, 14);

cout << "\t\t\t\t\t-----------------------------\n";

int comanda;

SetConsoleTextAttribute(hStdOut, 15);

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

while (!(cin >> comanda))

{

cin.clear();

while (cin.get() != '\n');

SortMoto(n, moto);

}

switch (comanda)

{

case 0: menu(n, moto); break;

case 1:

{

system("cls");

string\* str1 = new string[n];

for (int i = 0; i < n; i++)

{

str1[i] = moto[i].NumberEngine;

}

SortStr(str1, n, moto);

Draw(n, moto);

};

break;

case 2:

{

system("cls");

string\* str1 = new string[n];

for (int i = 0; i < n; i++)

{

str1[i] = moto[i].NumberFrame;

}

SortStr(str1, n, moto);

Draw(n, moto);

};

break;

case 3:

{

system("cls");

SetConsoleTextAttribute(hStdOut, 3);

cout << "\t\t\t\t\t1 - Возрастанию\n";

cout << "\t\t\t\t\t2 - Убыванию\n";

SetConsoleTextAttribute(hStdOut, 15);

int com;

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

while (!(cin >> com))

{

cin.clear();

while (cin.get() != '\n');

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 1:

{

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

if (moto[i].Volume > moto[j].Volume)

{

swap(moto[i], moto[j]);

}

}

};

break;

case 2:

{

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

if (moto[i].Volume < moto[j].Volume)

{

swap(moto[i], moto[j]);

}

}

};

break;

default: SortMoto(n, moto);

}

};

break;

case 4:

{

system("cls");

SetConsoleTextAttribute(hStdOut, 3);

cout << "\t\t\t\t\t1 - Возрастанию\n";

cout << "\t\t\t\t\t2 - Убыванию\n";

SetConsoleTextAttribute(hStdOut, 15);

int com;

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

while (!(cin >> com))

{

cin.clear();

while (cin.get() != '\n');

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 1:

{

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

if (moto[i].Power > moto[j].Power)

{

swap(moto[i], moto[j]);

}

}

};

break;

case 2:

{

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

if (moto[i].Power < moto[j].Power)

{

swap(moto[i], moto[j]);

}

}

};

break;

default: SortMoto(n, moto);

}

};

break;

case 5:

{

system("cls");

SetConsoleTextAttribute(hStdOut, 3);

cout << "\t\t\t\t\t1 - Возрастанию\n";

cout << "\t\t\t\t\t2 - Убыванию\n";

SetConsoleTextAttribute(hStdOut, 15);

int com;

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

while (!(cin >> com))

{

cin.clear();

while (cin.get() != '\n');

cout << "\n\t\t\t\t Введите номер команды и нажмите ENTER: ";

}

switch (com)

{

case 1:

{

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

if (moto[i].Price > moto[j].Price)

{

swap(moto[i], moto[j]);

}

}

};

break;

case 2:

{

for (int i = 0; i < n; i++)

{

for (int j = i + 1; j < n; j++)

if (moto[i].Price < moto[j].Price)

{

swap(moto[i], moto[j]);

}

}

};

break;

default: SortMoto(n, moto);

}

};

break;

default: SortMoto(n, moto);

}

for (int i = 0; i < n; i++)

{

moto[i].N = i + 1;

}

NewMotoRecordFile(n, moto);

Draw(n, moto);

return NULL;

}

//Сортировка

void SortStr(string\* S, int N, MotoShow\* moto)

{

/\*

Данная функция сортирует структуру по алфавиту по заданным параметрам

Входные параметры - 1.Массив структуры данных; 2. Массив строк;

Выходной параметр - Отсутствует.

\*/

for (int i = 0; i < N - 1; i++) {

for (int j = i + 1; j < N; j++)

{

if (strcmp(S[i].c\_str(), S[j].c\_str()) > 0){

swap(moto[i], moto[j]);

}

}

}

}

//5 - Вывод данных о разработчике

void information(int n, struct MotoShow\* moto)

{

/\*

Данная функция показывает информацию о разработчике программы;

Входной параметр - Массив структуры данных;

Выходной параметр - Отсутствует.

\*/

system("cls");

SetConsoleTextAttribute(hStdOut, 8);

gotoxy(45, 1);

cout << "Информация о разработчике";

gotoxy(30, 3);

cout << "Написал - Оганезов Павел ";

gotoxy(30, 4);

cout << "Возраст - 19 лет";

gotoxy(30, 5);

cout << "Курс - 2";

SetConsoleTextAttribute(hStdOut, 8);

int comanda;

gotoxy(30, 10);

cout << "Для выхода в меню введите \"0\" и нажмите ENTER: ";

while (!(cin >> comanda) || comanda != 0)

{

cin.clear();

while (cin.get() != '\n');

gotoxy(30, 10);

cout << "Для выхода в меню введите \"0\" и нажмите ENTER: ";

}

menu(n, moto);

}