**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

**Курсовая работа**

**по дисциплине «Программирование»**

Тема: **«Разработка электронной картотеки»**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | | Аршин А.Д |  |
|  |

Санкт-Петербург

2024

**Введение**

**Цель работы.**

Создание программы для управления электронной картотекой, которая позволяет хранить, редактировать, искать, сортировать и выводить данные по объектам предметной области.

**Задание**

Создать электронную картотеку, хранящуюся на диске, и программу, обеспечивающую взаимодействие с ней.

Программа должна выполнять следующие действия:

* занесение данных в электронную картотеку;
* внесение изменений (исключение, корректировка, добавление);
* поиск данных по различным признакам;
* сортировку по различным признакам;
* вывод результатов на экран и сохранение на диске.

Выбор подлежащих выполнению команд должен быть реализован с помощью основного меню и вложенных меню.

Задача должна быть структурирована и отдельные части должны быть оформлены как функции.

Исходные данные должны вводиться с клавиатуры. В процессе обработки картотека должна храниться в памяти компьютера в виде списков и массивов структур, связанных указателями. Типы списков и структур выбираются исходя из предметной области.

Картотека составляется по выбранной предметной области.

В программе должно быть реализовано простейшее меню. Выполнение программы должно быть многократным по желанию пользователя. Данные первоначально считываются из файла (файлов), в процессе работы данные вводятся с клавиатуры.

Примерный перечень пунктов меню:

0: Справка

1: Добавление карточек об объектах предметной области

2: Редактирование карточек

3: Удаление карточек

4: Вывод картотеки

5: Поиск карточек по параметру

6: Сортировка картотеки по параметру

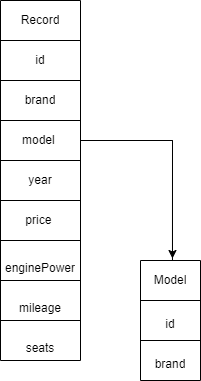
7: Выход

**Постановка задачи**: необходимо разработать программу, которая будет выполнять следующие функции:

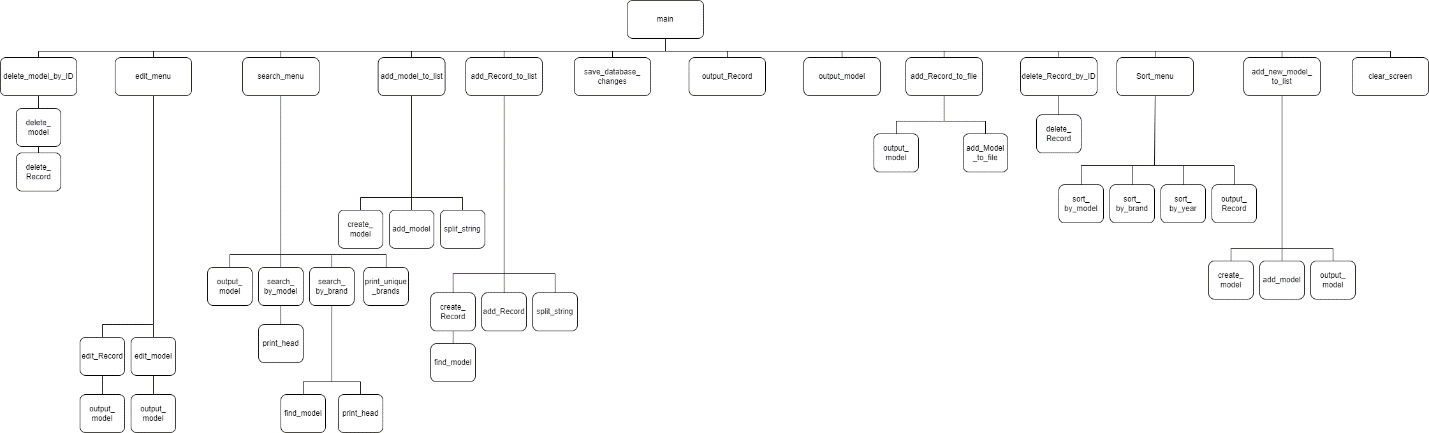
1. Занесение данных в электронную картотеку.
2. Внесение изменений в картотеку (исключение, корректировка, добавление).
3. Поиск данных по различным признакам.
4. Сортировка данных по различным признакам.
5. Вывод результатов на экран и сохранение на диске.
6. Реализация выбора команд с помощью основного и вложенных меню.

**Описание решения**: Программа будет структурирована на отдельные функции, которые будут отвечать за различные действия с картотекой. Данные будут храниться в памяти компьютера в виде списков и массивов структур, связанных указателями. Типы списков и структур будут выбраны в соответствии с предметной областью.

**Схема архитектуры данных:**

****

**Структура вызовов функций:**

****

**Описание переменных:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **save\_choice** | **char** | Хранит выбор пользователя о сохранении изменений перед выходом |
| 2 | **sep** | **char** | Хранит разделительный символ, используемый в операциях с файлами, который установлен в **';'** |
| 3 | **option** | **int** | Хранит выбор пользователя из меню |
| 4 | **id\_to\_delete** | **int** | Хранит идентификатор записи, которую необходимо удалить |
| 5 | **model\_id\_to\_delete** | **int** | Хранит идентификатор модели, которую необходимо удалить |
| 6 | **dph** | **DHD\*** | Указатель на начало связанного списка моделей. Этот указатель используется для управления операциями, связанными с данными моделей |
| 7 | **mph** | **MHD\*** | Указатель на начало связанного списка записей. Этот указатель используется для управления операциями, связанными с данными записей. |

**struct Record:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **id** | **int** | Хранит идентификатор записи, который служит для её уникальной идентификации в базе данных или в списке |
| 2 | **brand** | **char** | Указатель на строку, представляющую марку автомобиля, содержащуюся в данной записи |
| 3 | **model** | **DIR** | Указатель на тип **DIR**, который, возможно, используется для хранения информации о модели автомобиля, например, путь к файлу или директории, связанной с этой моделью |
| 4 | **year** | **int** | Содержит год выпуска автомобиля, который является числовым представлением года производства данной записи |
| 5 | **price** | **int** | Хранит цену автомобиля, выраженную в целочисленном формате |
| 6 | **enginePower** | **float** | Содержит мощность двигателя автомобиля, представленную в вещественном формате |
| 7 | **mileage** | **float** | Хранит информацию о пробеге автомобиля, также представленную в вещественном формате |
| 8 | **seats** | **int[3]** | Массив из трех целых чисел, представляющих количество мест в автомобиле, например, количество сидений |
| 9 | **prev** | **struct Record** | Указатель на предыдущую запись типа **Record** в связанном списке. Этот указатель используется для навигации назад по списку |
| 10 | **next** | **struct Record** | Указатель на следующую запись типа **Record** в связанном списке. Этот указатель используется для навигации вперед по списку |

**struct Record\_head**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **first** | **struct Record** | Указатель на первую запись типа **Record** в связанном списке. Этот указатель используется для обозначения начала списка |
| 2 | **last** | **struct Record** | Указатель на последнюю запись типа **Record** в связанном списке. Этот указатель используется для обозначения конца списка |

**struct model:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **id** | **int** | Хранит идентификатор модели, который используется для её уникальной идентификации |
| 2 | **brand** | **char** | Указатель на строку, содержащую название бренда модели |
| 3 | **next** | **struct model** | Указатель на следующую структуру типа **model** в связанном списке. Этот указатель используется для навигации вперед по списку. |
| 4 | **prev** | **struct model** | Указатель на предыдущую структуру типа **model** в связанном списке. Этот указатель используется для навигации назад по списку. |

**struct model\_head:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **first** | **struct model** | Указатель на первую модель в связанном списке моделей. Этот указатель используется для обозначения начала списка |
| 2 | **last** | **struct model** | Указатель на последнюю модель в связанном списке моделей. Этот указатель используется для обозначения конца списка |

**struct UniqueBrand:**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **id** | int | Целочисленное поле, которое представляет идентификатор бренда |
| 2 | **brand** | char | Массив символов фиксированной длины 100, представляющий название бренда. |

**Функция MHD \*make\_Record\_head():**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **ph** | **MHD** | Этот указатель создается для представления головы (head) связанного списка записей (Record). Внутри функции он инициализируется и возвращает для использования в других частях программы |

**Функция DHD \*make\_model\_head():**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **ph** | **DHD** | Этот указатель создается для представления головы (head) связанного списка моделей (model). Внутри функции он инициализируется и возвращает для использования в других частях программы |

**Функция DIR \*find\_model(DHD \*dph, int id):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **current** | **DIR** | Этот указатель используется для указания на текущую модель в связанном списке моделей (**DHD**). В начале выполнения функции он инициализируется значением **NULL**, а затем изменяется в цикле, чтобы перебирать модели в списке. |
| 2 | **flag** | **int** | Этот флаг используется для контроля выполнения цикла. Когда находится модель с идентификатором, соответствующим запрашиваемому **id**, значение флага устанавливается в 1, что приводит к завершению цикла |

**Функция MOV \*create\_Record(int id\_mov, char \*Record\_brand, int id, int Record\_year, int Record\_price, float Record\_enginePower, float Record\_mileage, int watch\_seats[3], DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **new\_Record** | **MOV** | Этот указатель представляет новую запись (Record). Он инициализируется как **NULL** и затем выделяет память под новую запись с помощью функции **malloc** |
| 2 | **Record\_model** | **DIR** | Этот указатель используется для указания на модель записи (Record). Выделяется память под него с помощью **malloc**, затем используется функция **find\_model()** для поиска модели с заданным идентификатором в списке моделей (**dph**). |
| 3 | **brand** | **char** | Этот указатель используется для хранения строки с названием бренда. Выделяется память под него с помощью **malloc**, затем копируется содержимое **Record\_brand** в выделенную область с помощью **memcpy**. |

**Функция DIR \*create\_model(int id, char \*Record\_model):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **new\_model** | **DIR** | Этот указатель представляет новую модель. Он инициализируется как **NULL** и затем выделяет память под новую модель с помощью функции **malloc** |
| 2 | **model** | **char** | Этот указатель используется для хранения строки с названием модели. Выделяется память под него с помощью **malloc**, затем копируется содержимое **Record\_model** в выделенную область с помощью **memcpy** |

**Функция void add\_Record(MHD \*ph, MOV \*new\_node, MOV \*current\_node):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **ph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей. Функция использует его для добавления новой записи. |
| 2 | **new\_node** | **MOV** | Этот указатель представляет новую запись, которая должна быть добавлена в список. Функция использует этот указатель для обновления указателей на следующую и предыдущую записи |
| 3 | **current\_node** | **MOV** | Этот указатель представляет текущую запись в списке, после которой должна быть добавлена новая запись. Функция использует этот указатель для определения местоположения новой записи в списке. |

**Функция void add\_model(DHD \*dph, DIR \*new\_node, DIR \*current\_node):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. Функция использует его для добавления новой модели |
| 2 | **new\_node** | **DIR** | Этот указатель представляет новую модель, которая должна быть добавлена в список. Функция использует этот указатель для обновления указателей на следующую и предыдущую модели |
| 3 | **current\_node** | **DIR** | Этот указатель представляет текущую модель в списке, после которой должна быть добавлена новая модель. Функция использует этот указатель для определения местоположения новой модели в списке |

**Функция void delete\_Record(MHD \*mph, MOV \*current\_Record):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей. Функция использует его для обновления указателей на первую и последнюю записи в случае удаления одной из них. |
| 2 | **current\_Record** | **MOV** | Этот указатель представляет удаляемую запись (Record). Функция использует его для удаления записи из связанного списка. |

**Функция void delete\_model(DHD \*dph, MHD \*mph, DIR \*current\_model):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей (model). Функция использует его для обновления указателей на первую и последнюю модели в случае удаления одной из них. |
| 2 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Record). Функция использует его для удаления всех записей, связанных с удаляемой моделью. |
| 3 | **current\_model** | **DIR** | Этот указатель представляет удаляемую модель (model). Функция использует его для удаления модели из связанного списка и обновления указателей на предыдущую и следующую модели |
| 4 | **current\_Record** | **MOV** | Этот указатель используется для перебора всех записей (Record) в списке и проверки, связаны ли они с удаляемой моделью. |
| 5 | **next\_Record** | **MOV** | Этот указатель используется для хранения ссылки на следующую запись (Record) в списке перед удалением текущей записи |

**Функция void output\_Record(MHD \*mph, int n, int reverse):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Record). Функция использует его для доступа к записям для вывода. |
| 2 | **n** | **int** | Этот аргумент определяет количество заголовков, которые будут выведены перед выводом записей |
| 3 | **reverse** | **int** | Этот аргумент указывает, нужно ли выводить записи в обратном порядке. Если **reverse** равен 1, записи будут выводиться в обратном порядке, начиная с последней записи. Если **reverse** равен 0, записи будут выводиться в обычном порядке, начиная с первой записи |
| 4 | **current** | **MOV** | Этот указатель используется для перемещения по списку записей (Record) при выводе. В начале выполнения функции он инициализируется указателем на первую запись или последнюю запись в зависимости от значения **reverse**. |

**Функция void output\_model(DHD \*dph, int n, int reverse):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. Функция использует его для доступа к моделям для вывода. |
| 2 | **n** | **int** | Этот аргумент определяет количество заголовков, которые будут выведены перед выводом моделей. |
| 3 | **reverse** | **int** | Этот аргумент указывает, нужно ли выводить модели в обратном порядке. Если **reverse** равен 1, модели будут выводиться в обратном порядке, начиная с последней модели. Если **reverse** равен 0, модели будут выводиться в обычном порядке, начиная с первой модели. |
| 4 | **current** | **DIR** | Этот указатель используется для перемещения по списку моделей при выводе. В начале выполнения функции он инициализируется указателем на первую модель или последнюю модель в зависимости от значения **reverse**. |

**Функция void split\_string(char \*inputString, char \*\*words, int \*wordCount, char delimiter):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **inputString** | **char** | Эта переменная представляет входную строку, которая будет разбита на подстроки с использованием разделителя. |
| 2 | **words** | **char** | Это массив указателей на символьные строки, которые будут содержать разделенные слова |
| 3 | **wordCount** | **int** | Это указатель на целое число, которое будет содержать количество разделенных слов. |
| 4 | **delimiter** | **char** | Этот символ определяет разделитель, который будет использоваться для разделения входной строки на подстроки. |
| 5 | **wordIndex** | **int** | Эта переменная используется для отслеживания текущего индекса в массиве **words** |
| 6 | **wordStart** | **int** | Эта переменная указывает на начало текущего слова во входной строке |
| 7 | **wordLength** | **int** | Эта переменная указывает на длину текущего слова во входной строке. |
| 8 | **inWord** | **int** | та переменная используется для указания, находится ли текущая позиция внутри слова или за его пределами. |
| 9 | **i** | **int** | Эта переменная используется в качестве счетчика для итерации по символам во входной строке |

**Функция void add\_Record\_to\_list(char \*filebrand, MHD \*mph, char sep, DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **filebrand** | **char** | Эта переменная содержит имя файла, из которого будут считываться данные для создания новых записей (Record) |
| 2 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Record), в который будут добавляться новые записи |
| 3 | **sep** | **char** | Этот символ представляет разделитель, который будет использоваться при разделении строк во входном файле |
| 4 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей (model), который будет использоваться для создания связей между записями и моделями. |
| 5 | **line** | **char** | Этот массив содержит строку, которая считывается из файла. |
| 6 | **words** | **char** | Этот массив содержит указатели на слова, которые были разделены из строки с помощью функции **split\_string()** |
| 7 | **wordCount** | **int** | Это целочисленная переменная, которая содержит количество слов в текущей строке |
| 8 | **seats** | **int** | Этот массив содержит количество мест в автомобиле, которое будет прочитано из входных данных |
| 9 | **i** | **int** | Это переменная используется в качестве счетчика в цикле для освобождения памяти, выделенной для разделенных слов |
| 10 | **new\_Record** | **MOV** | Этот указатель представляет новую запись (Record), которая будет создана на основе данных из входного файла |

**Функция void add\_model\_to\_list(char \*filebrand, DHD \*dph, char sep):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **filebrand** | **char** | Эта переменная содержит имя файла, из которого будут считываться данные для создания новых моделей. |
| 2 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей, в который будут добавляться новые модели. |
| 3 | **sep** | **char** | Этот символ представляет разделитель, который будет использоваться для разделения строк во входном файле. |
| 4 | **line** | **char** | Этот массив содержит строку, которая считывается из файла. |
| 5 | **words** | **char** | Этот массив содержит указатели на слова, которые были разделены из строки с помощью функции **split\_string()**. |
| 6 | **wordCount** | **int** | Это целочисленная переменная, которая содержит количество слов в текущей строке. |
| 7 | **i** | **int** | Эта переменная используется в качестве счетчика в цикле для освобождения памяти, выделенной для разделенных слов. |
| 8 | **new\_model** | **DIR** | Этот указатель представляет новую модель, которая будет создана на основе данных из входного файла |
| 9 | **file** | **FILE** | Этот указатель представляет файл, из которого будут считываться данные для создания новых моделей. |

**Функция void add\_Record\_to\_file(char \*filebrand, MHD \*mph, char sep, DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **filebrand** | **char** | Эта переменная содержит имя файла, в который будут добавляться данные о новой записи (Record) |
| 2 | **id\_mov** | **int** | Эта переменная представляет идентификатор новой записи, который будет присвоен записи после добавления в файл. По умолчанию установлено значение 1, но если в файле уже есть записи, то идентификатор новой записи будет на единицу больше последнего идентификатора в файле |
| 3 | **file** | **FILE** | Этот указатель представляет файл, в который будут добавляться данные о новой записи (Record). Если файл не удается открыть, выводится сообщение об ошибке |
| 4 | **id** | **int** | заполняются пользователем с клавиатуры |
| 5 | **year** | **int** | заполняются пользователем с клавиатуры |
| 6 | **price** | **int** | заполняются пользователем с клавиатуры |
| 7 | **seats** | **int** | заполняются пользователем с клавиатуры |
| 8 | **brand** | **char** | заполняются пользователем с клавиатуры |
| 9 | **enginePower** | **float** | заполняются пользователем с клавиатуры |
| 10 | **mileage** | **float** | заполняются пользователем с клавиатуры |

**Функция int add\_Model\_to\_file(char \*filemodel, DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **filemodel** | **char** | Эта переменная содержит имя файла, в который будут добавляться данные о новой модели. |
| 2 | **id** | **int** | Эта переменная представляет идентификатор новой модели, который будет присвоен модели после добавления в файл. Если в файле уже есть модели, идентификатор новой модели будет на единицу больше последнего идентификатора в файле. Если файл пуст, устанавливается значение 1. |
| 3 | **model** | **char** | Этот массив используется для хранения имени новой модели, которое вводится пользователем с клавиатуры. |
| 4 | **file** | **FILE** | Этот указатель представляет файл, в который будут добавляться данные о новой модели. Если файл не удается открыть, выводится сообщение об ошибке |

**Функция void delete\_model\_by\_ID(DHD \*dph, MHD \*mph, int id):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. |
| 2 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Records) |
| 3 | **id** | **int** | Эта переменная представляет идентификатор модели, которую необходимо удалить из списка. |
| 4 | **current\_model** | **DIR** | Этот указатель используется для перебора моделей в связанном списке моделей. Инициализируется как указатель на первую модель в списке. |

**Функция void delete\_Record\_by\_ID(MHD \*mph, int id):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Records). |
| 2 | **id** | **int** | Эта переменная представляет идентификатор записи, которую необходимо удалить из списка |
| 3 | **current\_Record** | **MOV** | Этот указатель используется для перебора записей в связанном списке записей (Records). Инициализируется как указатель на первую запись в списке |
| 4 | **next\_Record** | **MOV** | Этот указатель используется для хранения адреса следующей записи после текущей записи. Используется для сохранения ссылки на следующую запись перед удалением текущей записи из списка. |

**Функция void search\_by\_model(MHD \*mph, DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Records) |
| 2 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. |
| 3 | **found** | **int** | Эта переменная используется для отслеживания того, была ли найдена хотя бы одна запись с указанным идентификатором модели. Инициализируется значением 0 (ложь), что означает, что записи с указанным идентификатором модели еще не найдены. |
| 4 | **current\_Record** | **MOV** | Этот указатель используется для перебора записей в связанном списке записей (Records). Инициализируется как указатель на первую запись в списке. |
| 5 | **model\_id** | **int** | Эта переменная используется для хранения идентификатора модели, по которому будет осуществляться поиск записей в базе данных. |

**Функция void search\_by\_brand(MHD \*mph, DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Records). |
| 2 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. |
| 3 | **brand\_id** | **int** | Эта переменная используется для хранения идентификатора бренда, по которому будет осуществляться поиск записей в базе данных. |
| 4 | **brand** | **DIR** | Этот указатель используется для хранения найденного бренда по его идентификатору. |
| 5 | **current\_Record** | **MOV** | Этот указатель используется для перебора записей в связанном списке записей (Records). Инициализируется как указатель на первую запись в списке. |
| 6 | **found** | **int** | Эта переменная используется для отслеживания того, были ли найдены записи с указанным идентификатором бренда. Инициализируется значением 0 (ложь), что означает, что записи с указанным идентификатором бренда еще не найдены. |
| 7 | **brand\_name** | **char** | Этот массив используется для хранения имени бренда, который был найден по его идентификатору. Используется для вывода имени бренда в сообщениях пользователю. |

**Функция void sort\_by\_year(MHD \*mph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Records). |
| 2 | **current** | **MOV** | Этот указатель используется для перемещения по связанному списку записей (Records) во время сортировки |
| 3 | **temp** | **MOV** | Этот временный указатель используется для обмена местами двух соседних элементов списка при сортировке. |
| 4 | **swapped** | **int** | Эта переменная используется для отслеживания того, были ли произведены какие-либо обмены элементов списка при сортировке. Если значение **swapped** равно 1, это означает, что как минимум один обмен был выполнен, и сортировка должна продолжаться. |

**Функция void sort\_by\_brand(MHD \*mph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Records). |
| 2 | **current** | **MOV** | Этот указатель используется для перемещения по связанному списку записей (Records) во время сортировки |
| 3 | **temp** | **MOV** | Этот временный указатель используется для обмена местами двух соседних элементов списка при сортировке |
| 4 | **swapped** | **int** | Эта переменная используется для отслеживания того, были ли произведены какие-либо обмены элементов списка при сортировке. Если значение **swapped** равно 1, это означает, что как минимум один обмен был выполнен, и сортировка должна продолжаться. |

**Функция void sort\_by\_model(MHD \*mph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей (Records). |
| 2 | **current** | **MOV** | Этот указатель используется для перемещения по связанному списку записей (Records) во время сортировки |
| 3 | **temp** | **MOV** | Этот временный указатель используется для обмена местами двух соседних элементов списка при сортировке. |
| 4 | **swapped** | **int** | Эта переменная используется для отслеживания того, были ли произведены какие-либо обмены элементов списка при сортировке. Если значение **swapped** равно 1, это означает, что как минимум один обмен был выполнен, и сортировка должна продолжаться. |

**Функция void add\_new\_model\_to\_list(DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. |
| 2 | **model** | **char** | Этот массив используется для хранения имени новой модели, введенного пользователем. |
| 3 | **new\_model** | **DIR** | Этот указатель используется для хранения адреса новой структуры модели, созданной функцией **create\_model()** |
| 4 | **id** | **int** | Эта переменная используется для хранения уникального идентификатора новой модели. Он вычисляется как id последней модели плюс один. |

**Функция void edit\_Record(MHD \*mph, DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей. |
| 2 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. |
| 3 | **new\_brand** | **char** | Этот массив используется для хранения нового названия марки, введенного пользователем для редактирования записи. |
| 4 | **new\_model\_id** | **int** | Эта переменная используется для хранения нового идентификатора модели, введенного пользователем для редактирования записи. |
| 5 | **id\_to\_edit** | **int** | Эта переменная используется для хранения идентификатора записи, которую пользователь хочет отредактировать. |
| 6 | **edit\_line\_option** | **int** | Эта переменная используется для хранения выбранной пользователем опции для редактирования. |
| 7 | **record\_found** | **int** | Эта переменная используется для отслеживания того, была ли найдена запись с указанным пользователем идентификатором. |
| 8 | **current\_Record** | **MOV** | Этот указатель используется для перебора списка записей и поиска записи для редактирования |

**Функция void edit\_model(DHD \*dph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. |
| 2 | **new\_brand** | **char** | Этот массив используется для хранения нового названия модели, введенного пользователем для редактирования. |
| 3 | **current\_model** | **DIR** | Этот указатель используется для перебора списка моделей и поиска модели для редактирования. |
| 4 | **id\_to\_edit** | **int** | Эта переменная используется для хранения идентификатора модели, которую пользователь хочет отредактировать. |
| 5 | **model\_found** | **int** | Эта переменная используется для отслеживания того, была ли найдена модель с указанным пользователем идентификатором. |

**Функция void save\_database\_changes(char \*Record\_filename, char \*model\_filename, MHD \*mph, DHD \*dph, char sep):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **Record\_filename** | **char** | Этот указатель содержит имя файла для сохранения изменений в базе данных записей. |
| 2 | **model\_filename** | **char** | Этот указатель содержит имя файла для сохранения изменений в базе данных моделей |
| 3 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей. |
| 4 | **dph** | **DHD** | Этот указатель представляет голову (head) связанного списка моделей. |
| 5 | **sep** | **char** | Этот символ представляет разделитель, который будет использоваться при сохранении данных в файле моделей. |
| 6 | **Record\_file** | **FILE** | Этот указатель используется для работы с файлом записей. |
| 7 | **model\_file** | **FILE** | Этот указатель используется для работы с файлом моделей. |
| 8 | **current\_Record** | **MOV** | Этот указатель используется для перебора списка записей в базе данных записей. |
| 9 | **current\_model** | **DIR** | Этот указатель используется для перебора списка моделей в базе данных моделей. |

**Функция void print\_unique\_brands(MHD \*mph):**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Имя переменно | Тип | Назначение |
| 1 | **mph** | **MHD** | Этот указатель представляет голову (head) связанного списка записей. |
| 2 | **current\_Record** | **MOV** | Этот указатель используется для перебора списка записей в базе данных записей. |
| 3 | **unique\_brands** | **char** | Этот массив используется для хранения уникальных брендов (марок) автомобилей. Каждая строка массива представляет собой уникальный бренд. |
| 4 | **unique\_ids** | **int** | Этот массив используется для хранения уникальных идентификаторов моделей, соответствующих уникальным брендам. Каждый элемент массива соответствует уникальному бренду и содержит идентификатор модели, связанной с этим брендом |
| 5 | **unique\_count** | **int** | Эта переменная используется для отслеживания количества уникальных брендов, найденных в базе данных записей. |
| 6 | **i** | **int** | Эта переменная используется в циклах для итерации по элементам массивов и других операциях. |

**Контрольные примеры**

**Пример 1:**

Choose the option

0 - for EXIT program

1 - for show the Record data

2 - for show the model data

3 - add data to database

4 - delete data from the database

5 - search menu

6 - sort menu

7 - edit data menu

8 - SHOW directory

9 - add new model to list

10 - remove model from list

Enter the option: 1

Your selection is SHOW THE Record DATA

| ID | brand | model | Year | Price | enginePower | mileage | Seats |

+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+

| 0 | Toyota | Crown | 1994 | 9601 | 200.0 | 360.0 | 5 5 5 |

| 1 | Audi | A4 | 2010 | 40000 | 300.0 | 25.1 | 3 1 5 |

| 2 | Nissan | Skyline | 2023 | 45000 | 156.0 | 15.7 | 3 1 5 |

| 3 | Nissan | Skyline | 1997 | 40000 | 123.0 | 320.5 | 1 3 5 |

| 4 | Bmw | N-series | 1993 | 12002 | 213.0 | 333.3 | 4 1 5 |

| 5 | Audi | A4 | 2019 | 22000 | 234.0 | 140.8 | 2 3 5 |

| 6 | Mazda | Rs | 2023 | 45000 | 123.0 | 20.4 | 3 3 5 |

| 7 | Toyota | Crown | 2003 | 30000 | 234.0 | 25.9 | 4 1 5 |

| 8 | Nissan | Skyline | 2013 | 163950 | 234.0 | 598.0 | 3 1 5 |

| 9 | Ford | Mustang | 2023 | 12550 | 234.0 | 518.0 | 4 4 5 |

| 10 | Toyota | Crown | 2013 | 16537 | 321.0 | 558.0 | 1 1 5 |

| 11 | renault | logan | 2020 | 1 | 1.0 | 1.0 | 1 1 1 |

| 12 | ww | duster | 2010 | 200 | 1.0 | 1.0 | 1 1 1 |

Press ENTER to continue

**Пример 2:**

Choose the option

0 - for EXIT program

1 - for show the Record data

2 - for show the model data

3 - add data to database

4 - delete data from the database

5 - search menu

6 - sort menu

7 - edit data menu

8 - SHOW directory

9 - add new model to list

10 - remove model from list

Enter the option: 5

Your selection is SEARCH MENU

Select an option to search:

1 - Search by model

2 - Search by brand

Enter option: 1

Your selection is SEARCH DATA BY MODEL

| ID | model |

+----+---------------------------+

| 0 | Crown |

| 1 | A4 |

| 2 | Skyline |

| 3 | N-series |

| 4 | Rs |

| 5 | Mustang |

| 6 | logan |

| 7 | duster |

Enter the model ID to search: 0

Records with model ID '0':

| ID | brand | model | Year | Price | enginePower | mileage | Seats |

+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+

| 0 | Toyota | Crown | 1994 | 9601 | 200.0 | 360.0 | 5 5 5 |

| 7 | Toyota | Crown | 2003 | 30000 | 234.0 | 25.9 | 4 1 5 |

| 10 | Toyota | Crown | 2013 | 16537 | 321.0 | 558.0 | 1 1 5 |

**Пример 3:**

Choose the option

0 - for EXIT program

1 - for show the Record data

2 - for show the model data

3 - add data to database

4 - delete data from the database

5 - search menu

6 - sort menu

7 - edit data menu

8 - SHOW directory

9 - add new model to list

10 - remove model from list

Enter the option: 2

Your selection is SHOW THE model DATA

| ID | model |

+----+---------------------------+

| 0 | Crown |

| 1 | A4 |

| 2 | Skyline |

| 3 | N-series |

| 4 | Rs |

| 5 | Mustang |

| 6 | logan |

| 7 | duster |

Choose the option

0 - for EXIT program

1 - for show the Record data

2 - for show the model data

3 - add data to database

4 - delete data from the database

5 - search menu

6 - sort menu

7 - edit data menu

8 - SHOW directory

9 - add new model to list

10 - remove model from list

Enter the option: 1

Your selection is SHOW THE Record DATA

| ID | brand | model | Year | Price | enginePower | mileage | Seats |

+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+

| 0 | Toyota | Crown | 1994 | 9601 | 200.0 | 360.0 | 5 5 5 |

| 1 | Audi | A4 | 2010 | 40000 | 300.0 | 25.1 | 3 1 5 |

| 2 | Nissan | Skyline | 2023 | 45000 | 156.0 | 15.7 | 3 1 5 |

| 3 | Nissan | Skyline | 1997 | 40000 | 123.0 | 320.5 | 1 3 5 |

| 4 | Bmw | N-series | 1993 | 12002 | 213.0 | 333.3 | 4 1 5 |

| 5 | Audi | A4 | 2019 | 22000 | 234.0 | 140.8 | 2 3 5 |

| 6 | Mazda | Rs | 2023 | 45000 | 123.0 | 20.4 | 3 3 5 |

| 7 | Toyota | Crown | 2003 | 30000 | 234.0 | 25.9 | 4 1 5 |

| 8 | Nissan | Skyline | 2013 | 163950 | 234.0 | 598.0 | 3 1 5 |

| 9 | Ford | Mustang | 2023 | 12550 | 234.0 | 518.0 | 4 4 5 |

| 10 | Toyota | Crown | 2013 | 16537 | 321.0 | 558.0 | 1 1 5 |

| 11 | renault | logan | 2020 | 1 | 1.0 | 1.0 | 1 1 1 |

| 12 | ww | duster | 2010 | 200 | 1.0 | 1.0 | 1 1 1 |

Press ENTER to continue

Choose the option

0 - for EXIT program

1 - for show the Record data

2 - for show the model data

3 - add data to database

4 - delete data from the database

5 - search menu

6 - sort menu

7 - edit data menu

8 - SHOW directory

9 - add new model to list

10 - remove model from list

Enter the option: 10

Your selection is REMOVE MODEL FROM LIST

| ID | model |

+----+---------------------------+

| 0 | Crown |

| 1 | A4 |

| 2 | Skyline |

| 3 | N-series |

| 4 | Rs |

| 5 | Mustang |

| 6 | logan |

| 7 | duster |

Enter the ID of the model you want to delete: 0

Choose the option

0 - for EXIT program

1 - for show the Record data

2 - for show the model data

3 - add data to database

4 - delete data from the database

5 - search menu

6 - sort menu

7 - edit data menu

8 - SHOW directory

9 - add new model to list

10 - remove model from list

Enter the option: 2

Your selection is SHOW THE model DATA

| ID | model |

+----+---------------------------+

| 1 | A4 |

| 2 | Skyline |

| 3 | N-series |

| 4 | Rs |

| 5 | Mustang |

| 6 | logan |

| 7 | duster |

Press ENTER to continue

Choose the option

0 - for EXIT program

1 - for show the Record data

2 - for show the model data

3 - add data to database

4 - delete data from the database

5 - search menu

6 - sort menu

7 - edit data menu

8 - SHOW directory

9 - add new model to list

10 - remove model from list

Enter the option: 1

Your selection is SHOW THE Record DATA

| ID | brand | model | Year | Price | enginePower | mileage | Seats |

+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+

| 1 | Audi | A4 | 2010 | 40000 | 300.0 | 25.1 | 3 1 5 |

| 2 | Nissan | Skyline | 2023 | 45000 | 156.0 | 15.7 | 3 1 5 |

| 3 | Nissan | Skyline | 1997 | 40000 | 123.0 | 320.5 | 1 3 5 |

| 4 | Bmw | N-series | 1993 | 12002 | 213.0 | 333.3 | 4 1 5 |

| 5 | Audi | A4 | 2019 | 22000 | 234.0 | 140.8 | 2 3 5 |

| 6 | Mazda | Rs | 2023 | 45000 | 123.0 | 20.4 | 3 3 5 |

| 8 | Nissan | Skyline | 2013 | 163950 | 234.0 | 598.0 | 3 1 5 |

| 9 | Ford | Mustang | 2023 | 12550 | 234.0 | 518.0 | 4 4 5 |

| 11 | renault | logan | 2020 | 1 | 1.0 | 1.0 | 1 1 1 |

| 12 | ww | duster | 2010 | 200 | 1.0 | 1.0 | 1 1 1 |

Press ENTER to continue

**Пример 4**

**Choose the option**

**0 - for EXIT program**

**1 - for show the Record data**

**2 - for show the model data**

**3 - add data to database**

**4 - delete data from the database**

**5 - search menu**

**6 - sort menu**

**7 - edit data menu**

**8 - SHOW directory**

**9 - add new model to list**

**10 - remove model from list**

**Enter the option: 4**

**Your selection is DELETE RECORD BY ID**

**| ID | brand | model | Year | Price | enginePower | mileage | Seats |**

**+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+**

**| 0 | Toyota | Crown | 1994 | 9601 | 200.0 | 360.0 | 5 5 5 |**

**| 1 | Audi | A4 | 2010 | 40000 | 300.0 | 25.1 | 3 1 5 |**

**| 2 | Nissan | Skyline | 2023 | 45000 | 156.0 | 15.7 | 3 1 5 |**

**| 3 | Nissan | Skyline | 1997 | 40000 | 123.0 | 320.5 | 1 3 5 |**

**| 4 | Bmw | N-series | 1993 | 12002 | 213.0 | 333.3 | 4 1 5 |**

**| 5 | Audi | A4 | 2019 | 22000 | 234.0 | 140.8 | 2 3 5 |**

**| 6 | Mazda | Rs | 2023 | 45000 | 123.0 | 20.4 | 3 3 5 |**

**| 7 | Toyota | Crown | 2003 | 30000 | 234.0 | 25.9 | 4 1 5 |**

**| 8 | Nissan | Skyline | 2013 | 163950 | 234.0 | 598.0 | 3 1 5 |**

**| 9 | Ford | Mustang | 2023 | 12550 | 234.0 | 518.0 | 4 4 5 |**

**| 10 | Toyota | Crown | 2013 | 16537 | 321.0 | 558.0 | 1 1 5 |**

**| 11 | renault | logan | 2020 | 1 | 1.0 | 1.0 | 1 1 1 |**

**Enter the ID of the record you want to delete:0**

**Updated database:**

**| ID | brand | model | Year | Price | enginePower | mileage | Seats |**

**+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+**

**| 1 | Audi | A4 | 2010 | 40000 | 300.0 | 25.1 | 3 1 5 |**

**| 2 | Nissan | Skyline | 2023 | 45000 | 156.0 | 15.7 | 3 1 5 |**

**| 3 | Nissan | Skyline | 1997 | 40000 | 123.0 | 320.5 | 1 3 5 |**

**| 4 | Bmw | N-series | 1993 | 12002 | 213.0 | 333.3 | 4 1 5 |**

**| 5 | Audi | A4 | 2019 | 22000 | 234.0 | 140.8 | 2 3 5 |**

**| 6 | Mazda | Rs | 2023 | 45000 | 123.0 | 20.4 | 3 3 5 |**

**| 7 | Toyota | Crown | 2003 | 30000 | 234.0 | 25.9 | 4 1 5 |**

**| 8 | Nissan | Skyline | 2013 | 163950 | 234.0 | 598.0 | 3 1 5 |**

**| 9 | Ford | Mustang | 2023 | 12550 | 234.0 | 518.0 | 4 4 5 |**

**| 10 | Toyota | Crown | 2013 | 16537 | 321.0 | 558.0 | 1 1 5 |**

**| 11 | renault | logan | 2020 | 1 | 1.0 | 1.0 | 1 1 1 |**

**Press ENTER to continue**

**Пример 5:**

**Choose the option**

**0 - for EXIT program**

**1 - for show the Record data**

**2 - for show the model data**

**3 - add data to database**

**4 - delete data from the database**

**5 - search menu**

**6 - sort menu**

**7 - edit data menu**

**8 - SHOW directory**

**9 - add new model to list**

**10 - remove model from list**

**Enter the option: 5**

**Your selection is SEARCH MENU**

**Select an option to search:**

**1 - Search by model**

**2 - Search by brand**

**Enter option: 2**

**Your selection is SEARCH DATA BY BRAND**

**+----+-----------+**

**| ID | Brand |**

**+----+-----------+**

**| 1 | Audi|**

**| 2 | Nissan|**

**| 3 | Bmw|**

**| 4 | Mazda|**

**| 0 | Toyota|**

**| 5 | Ford|**

**| 6 | renault|**

**+----+-----------+**

**Enter the brand ID to search: 2**

**Records with brand 'Crown':**

**| ID | brand | model | Year | Price | enginePower | mileage | Seats |**

**+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+**

**| 2 | Nissan | Skyline | 2023 | 45000 | 156.0 | 15.7 | 3 1 5 |**

**| 3 | Nissan | Skyline | 1997 | 40000 | 123.0 | 320.5 | 1 3 5 |**

**| 8 | Nissan | Skyline | 2013 | 163950 | 234.0 | 598.0 | 3 1 5 |**

**Press ENTER to continue**

**Пример 6:**

**Choose the option**

**0 - for EXIT program**

**1 - for show the Record data**

**2 - for show the model data**

**3 - add data to database**

**4 - delete data from the database**

**5 - search menu**

**6 - sort menu**

**7 - edit data menu**

**8 - SHOW directory**

**9 - add new model to list**

**10 - remove model from list**

**Enter the option: 7**

**Your selection is EDIT DATA**

**Select an option to edit:**

**1 - Edit Record**

**2 - Edit model**

**Enter option: 2**

**| ID | model |**

**+----+---------------------------+**

**| 0 | Crown |**

**| 1 | A4 |**

**| 2 | Skyline |**

**| 3 | N-series |**

**| 4 | Rs |**

**| 5 | Mustang |**

**| 6 | logan |**

**| 7 | duster |**

**Enter the ID of the model you want to edit: 0**

**Enter new model name: cucli**

**Model updated successfully.**

**Press ENTER to continue**

**Choose the option**

**0 - for EXIT program**

**1 - for show the Record data**

**2 - for show the model data**

**3 - add data to database**

**4 - delete data from the database**

**5 - search menu**

**6 - sort menu**

**7 - edit data menu**

**8 - SHOW directory**

**9 - add new model to list**

**10 - remove model from list**

**Enter the option: 2**

**Your selection is SHOW THE model DATA**

**| ID | model |**

**+----+---------------------------+**

**| 0 | cucli |**

**| 1 | A4 |**

**| 2 | Skyline |**

**| 3 | N-series |**

**| 4 | Rs |**

**| 5 | Mustang |**

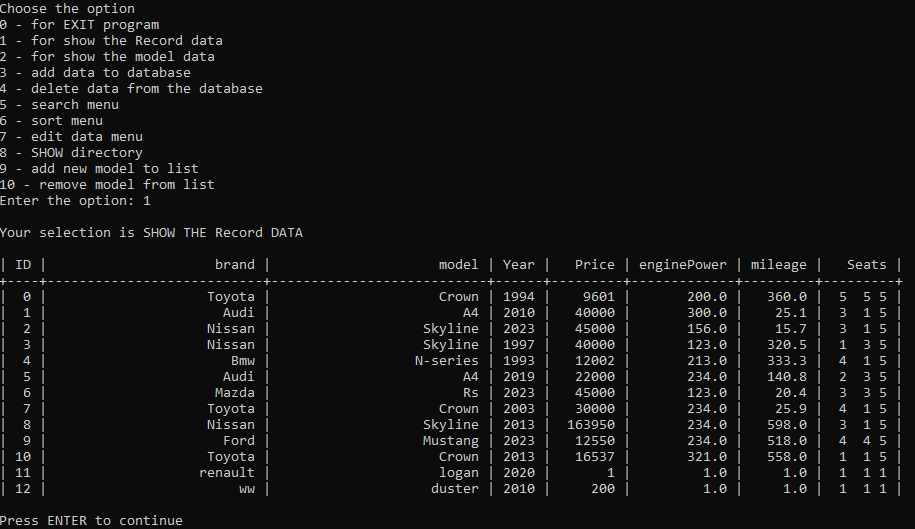
**| 6 | logan |**

**| 7 | duster |**

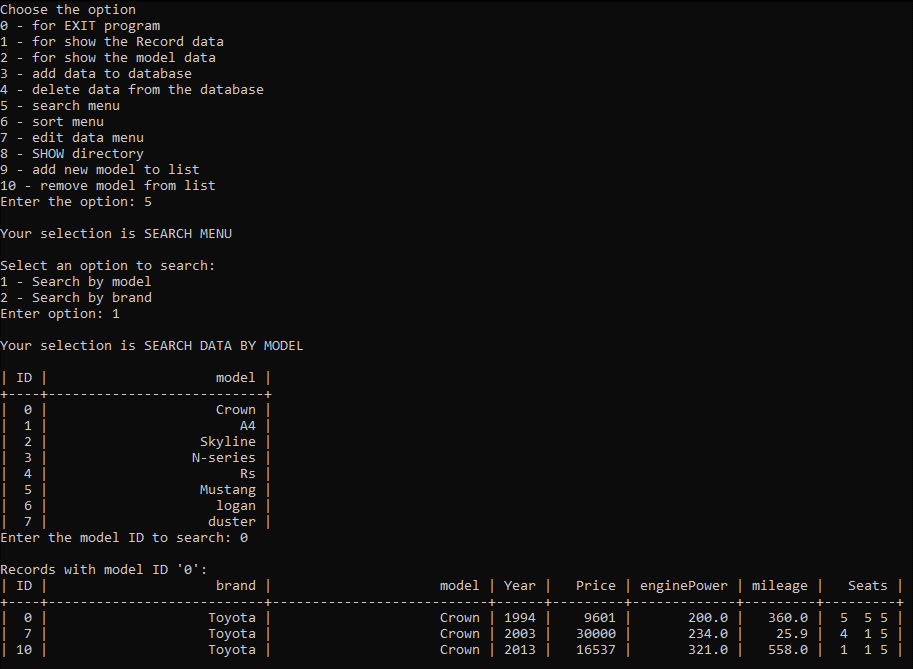
**Press ENTER to continue**

**Примеры выполнения программы**

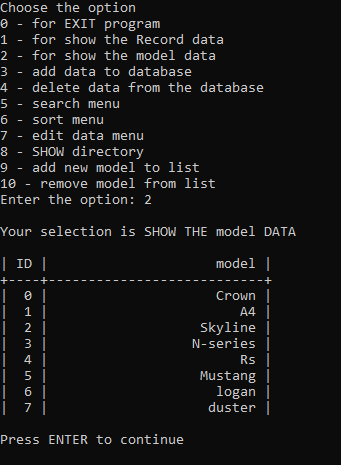
**Пример 1**

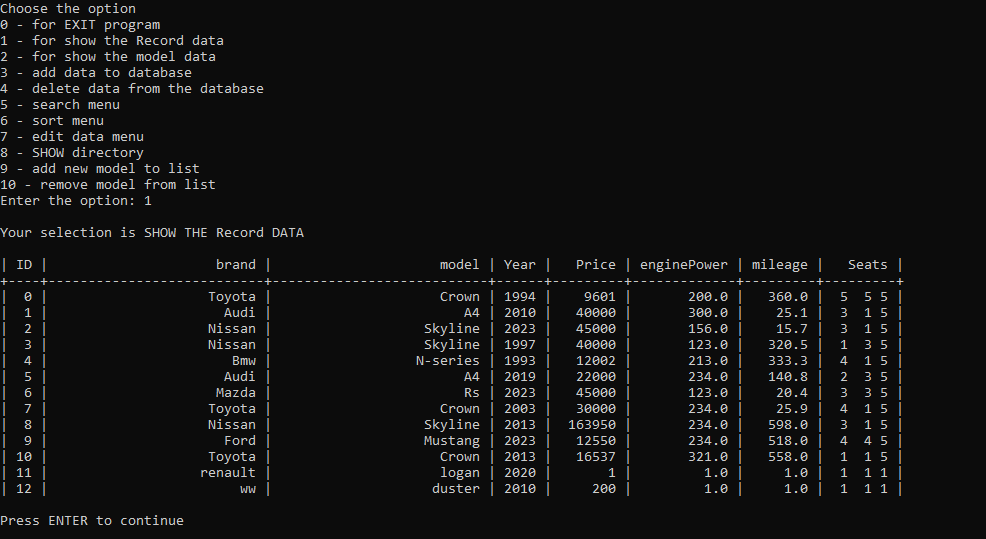


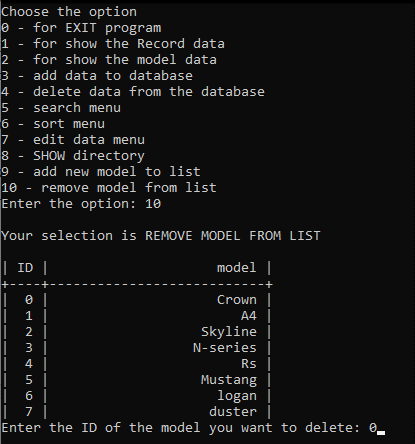
**Пример 2**

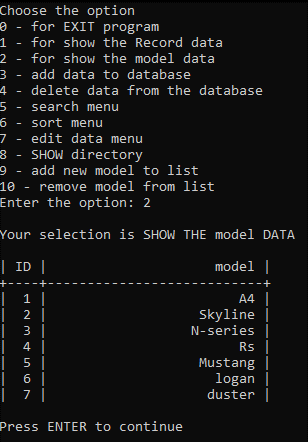


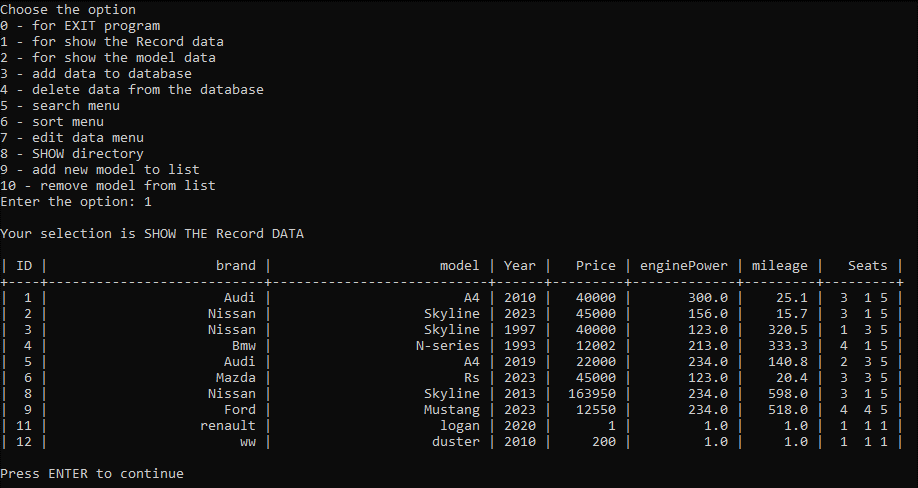
**Пример 3**



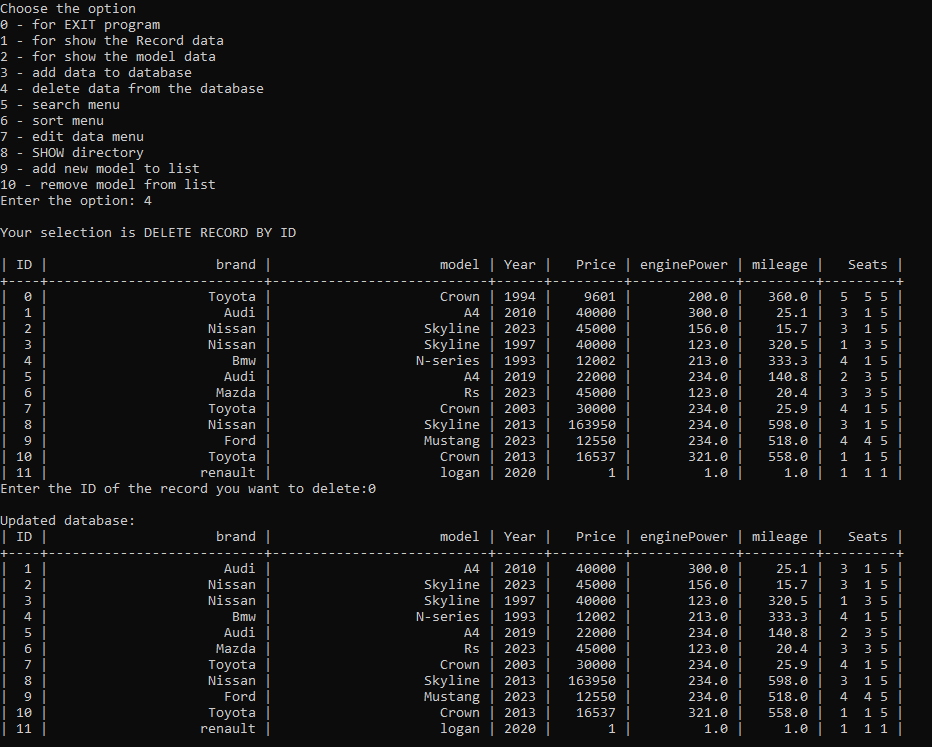




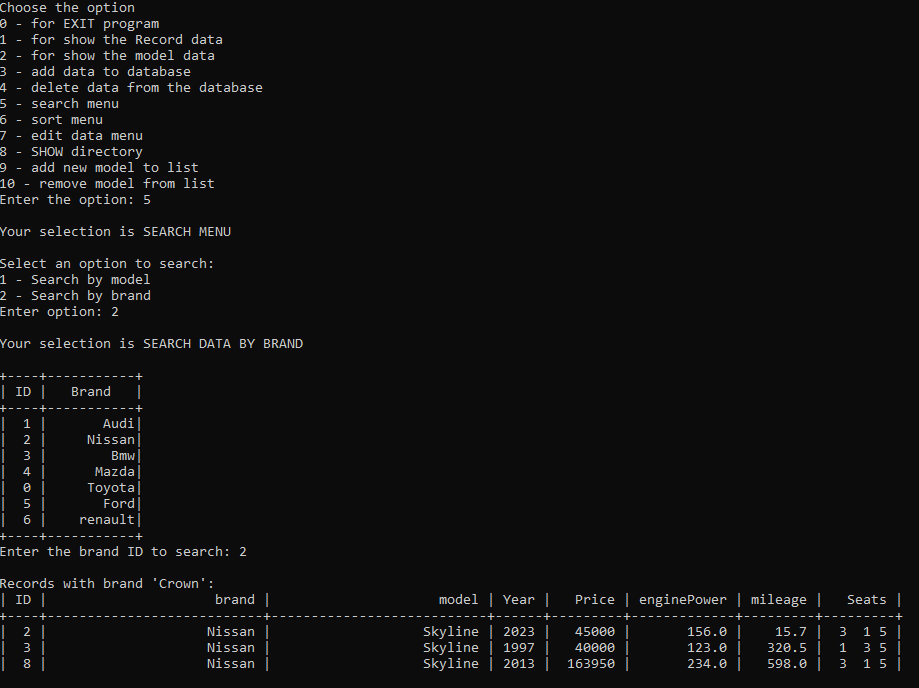




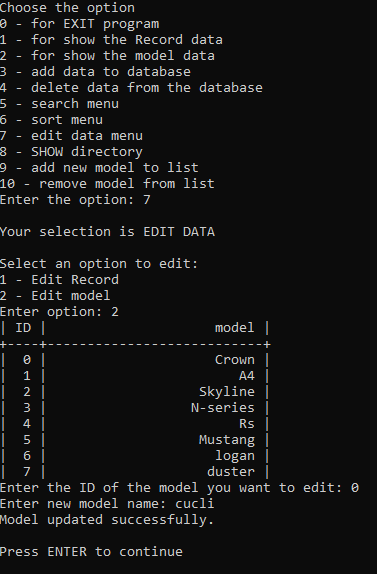
**Пример 4:**

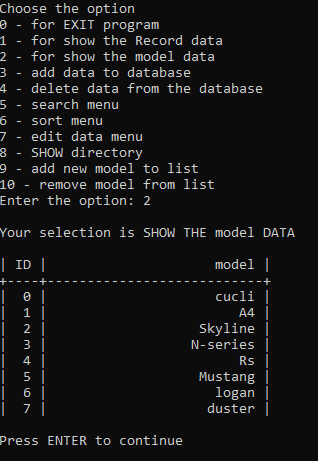


**Пример 5:**



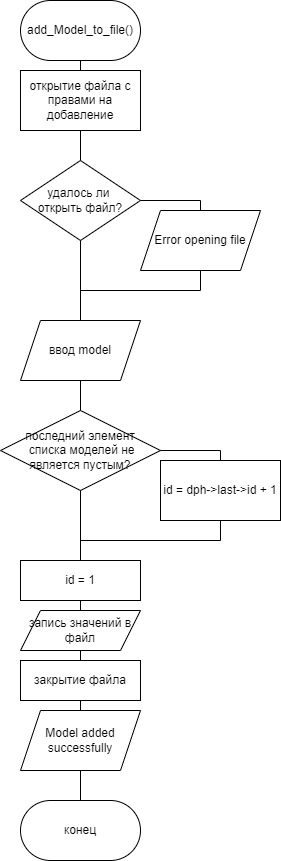
**Пример 6:**

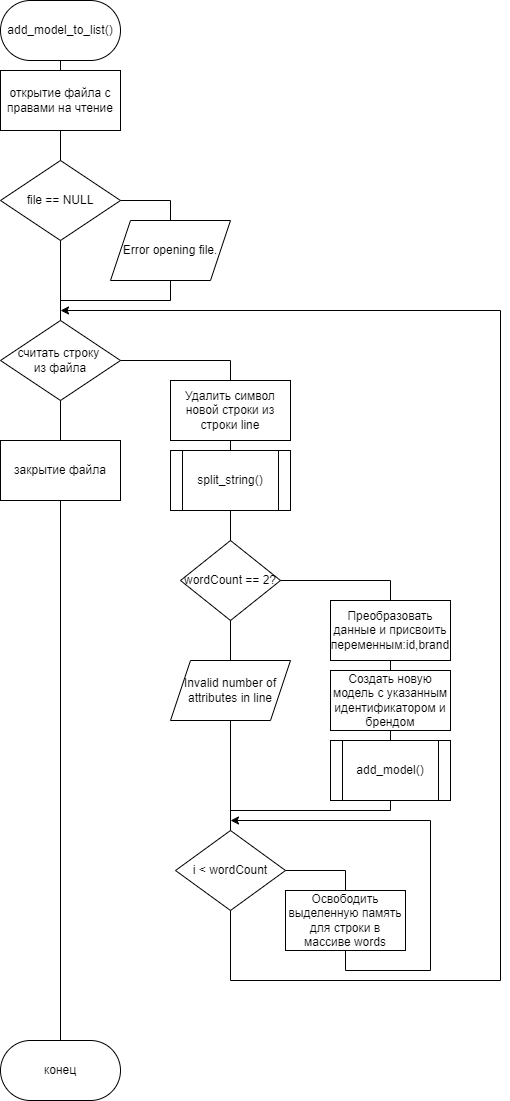


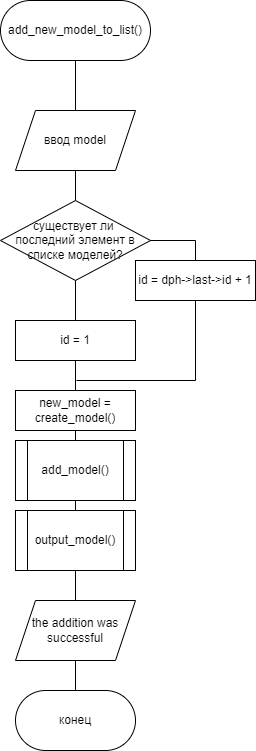


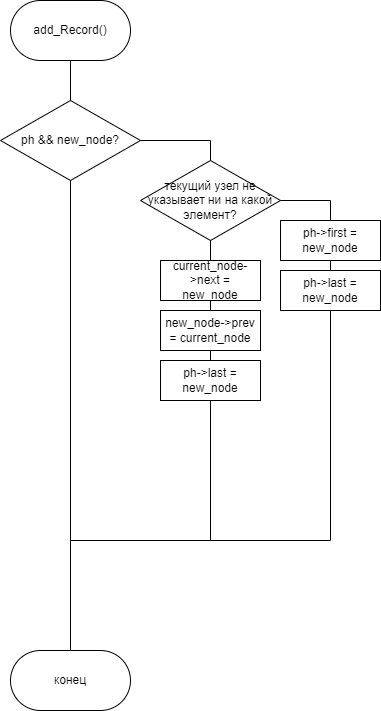
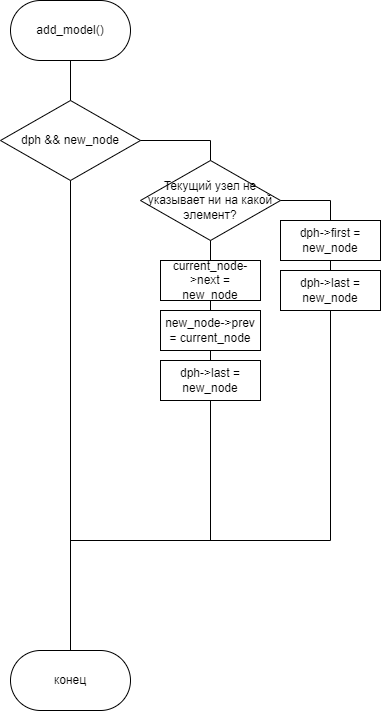
**Схема алгоритма:**

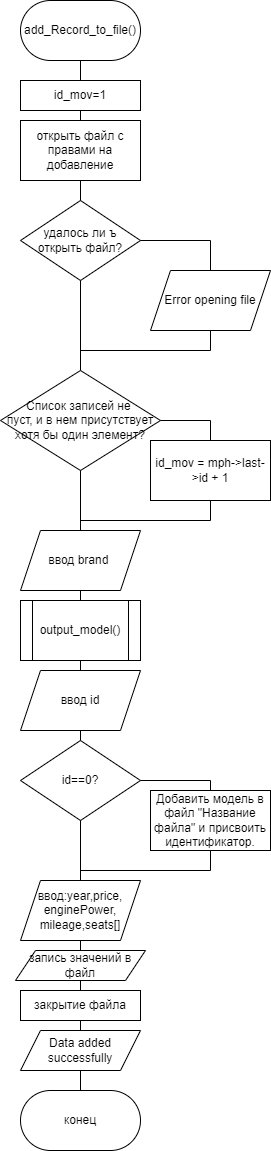
**Блоки-решения (вправо=да, вниз=нет)**

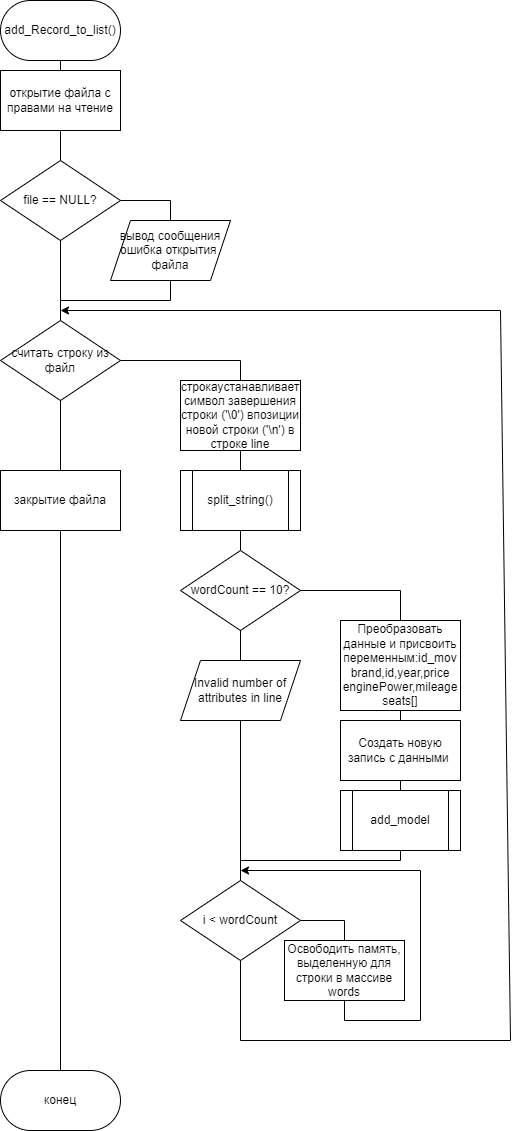
****

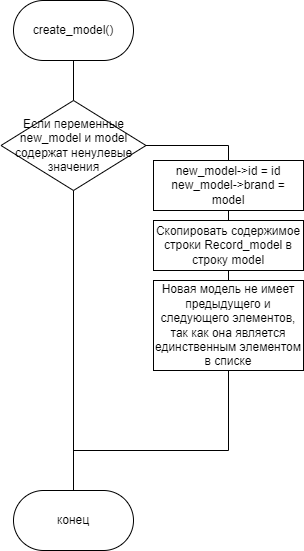
****

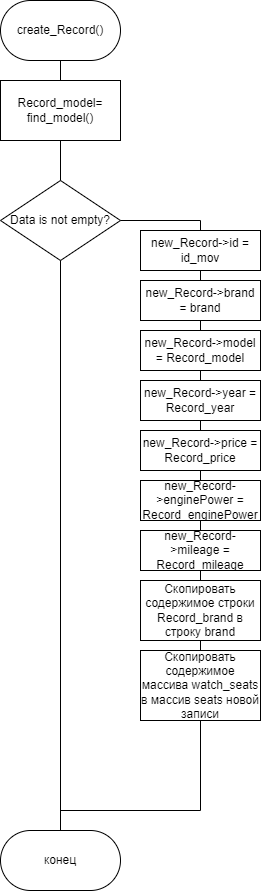
****

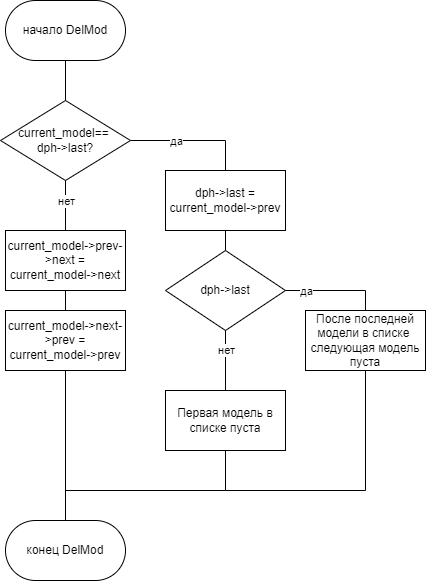
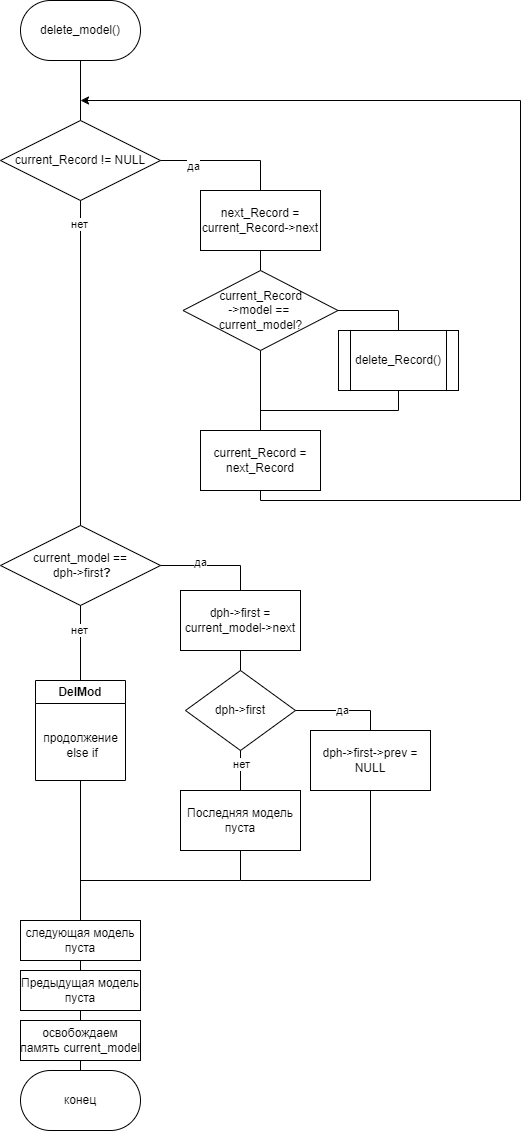
****

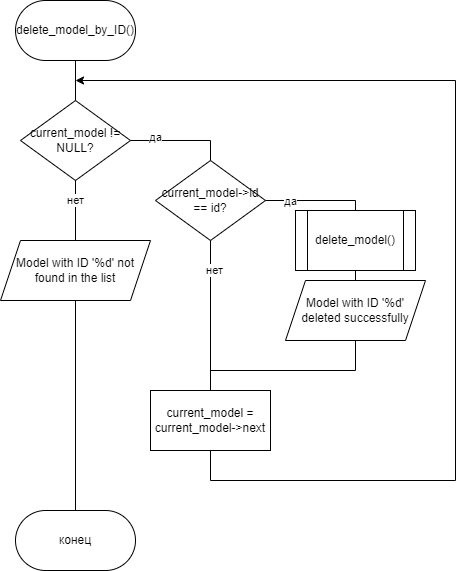
****

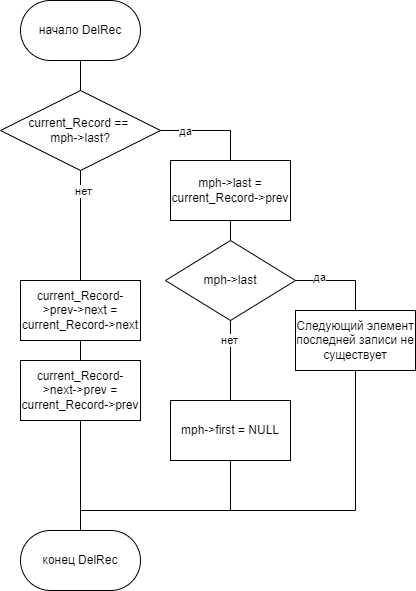
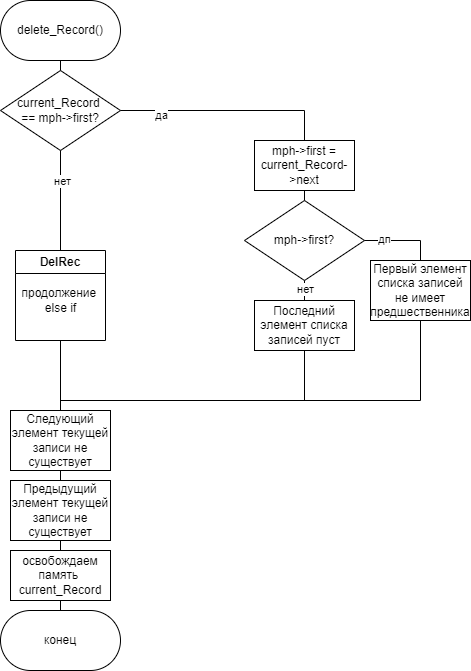
****

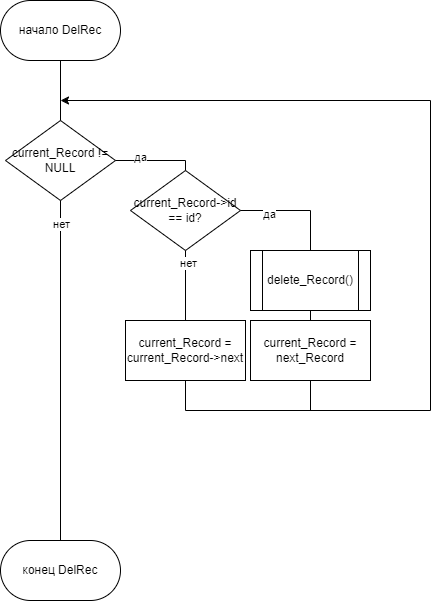
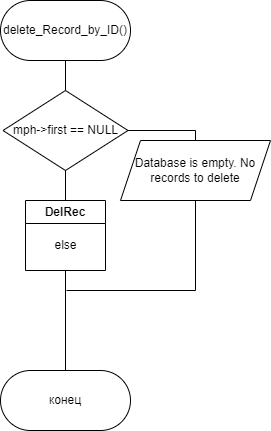
****

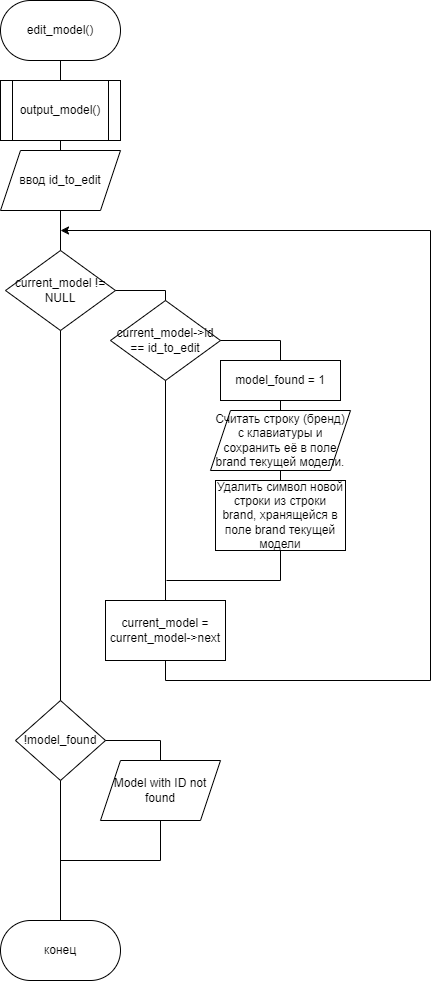
****

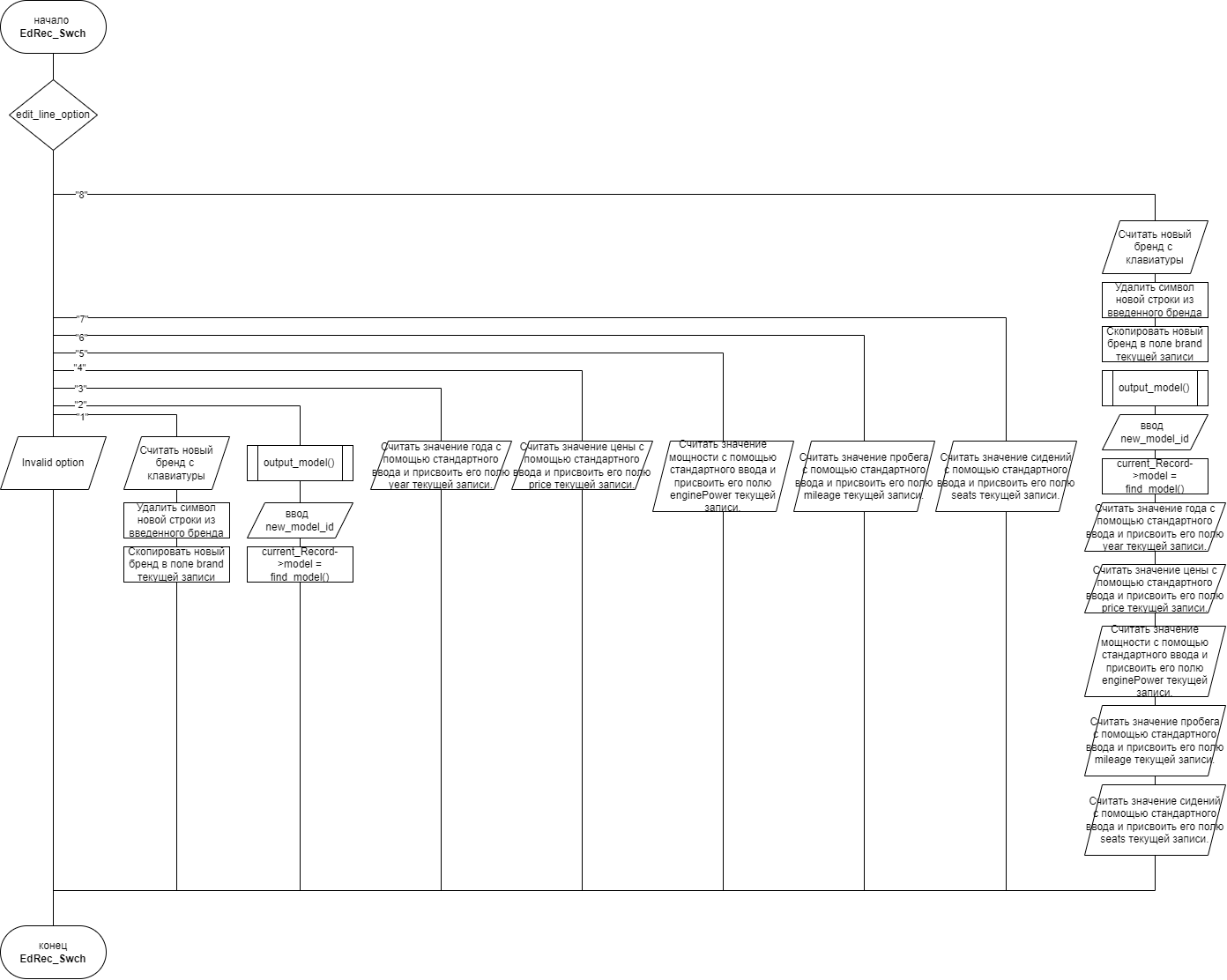
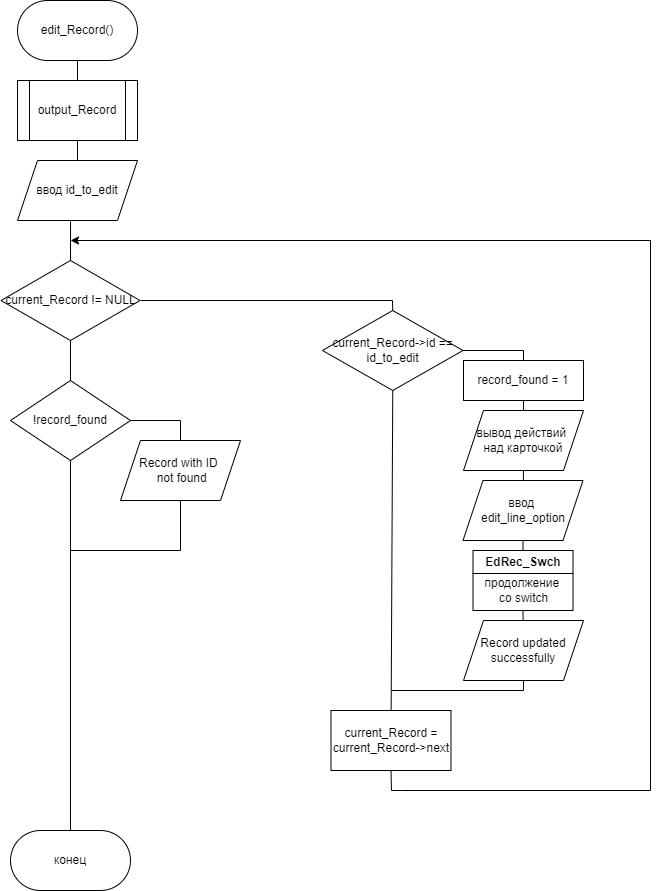
****

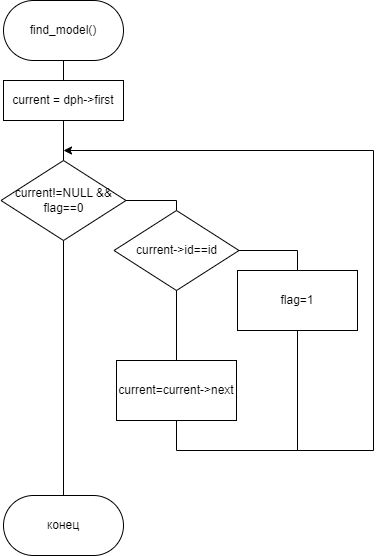
****

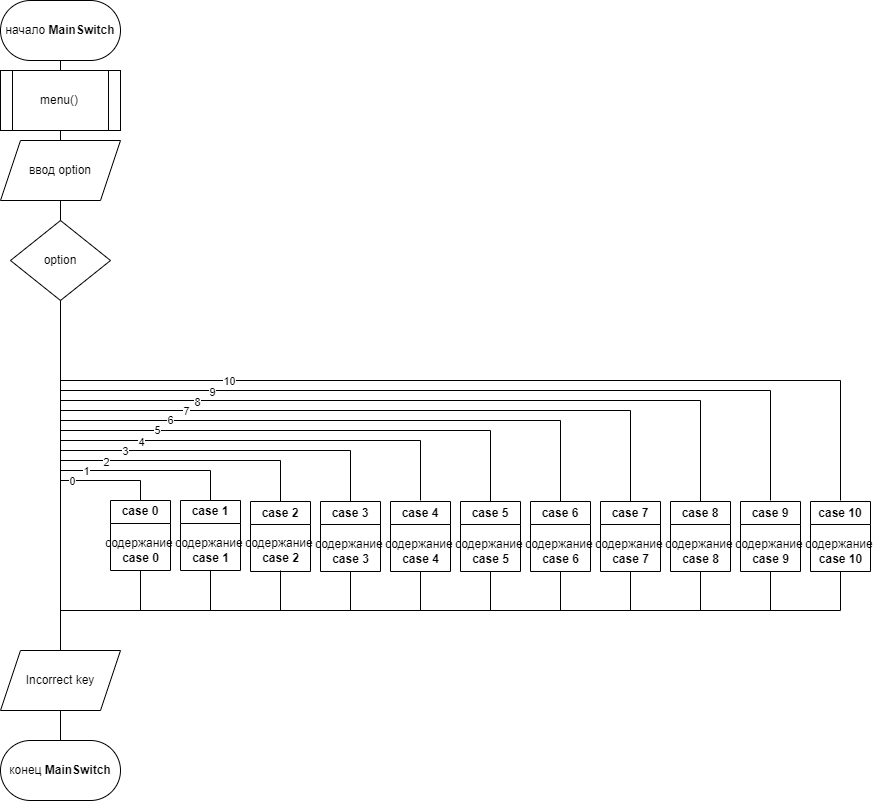
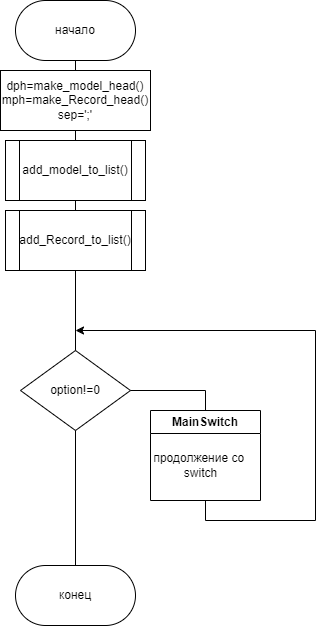
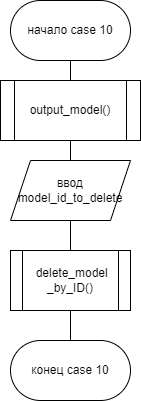
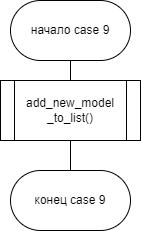
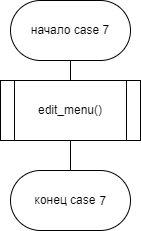
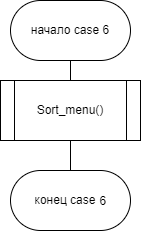
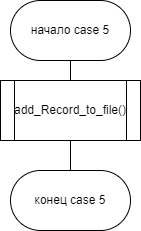
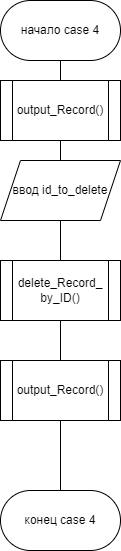
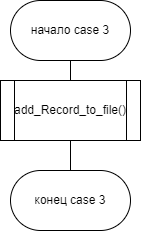
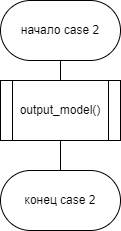
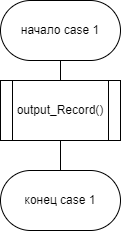
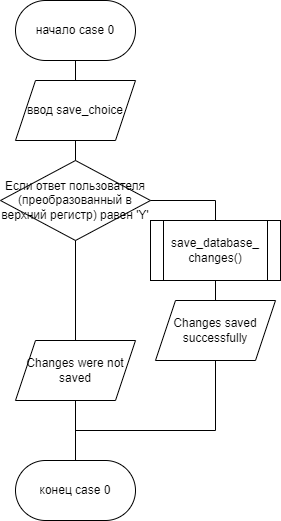
****

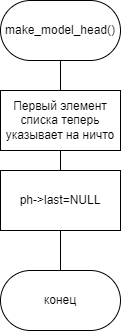
****

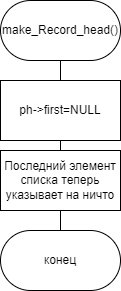
****

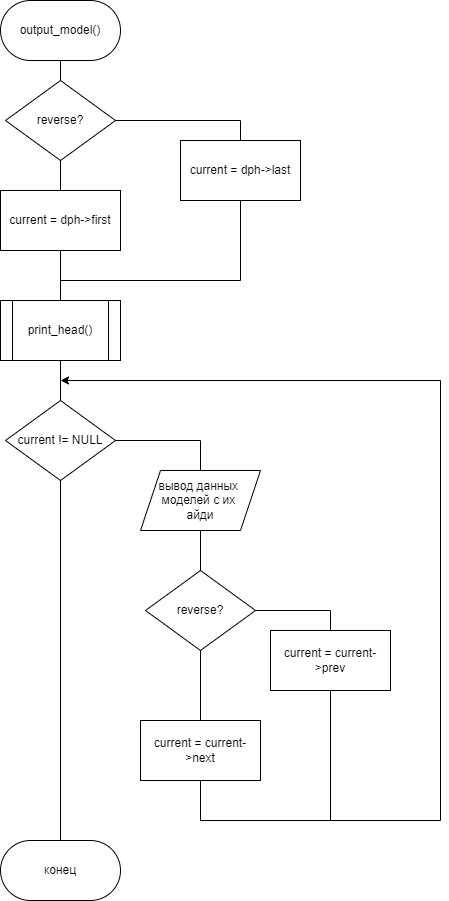
****

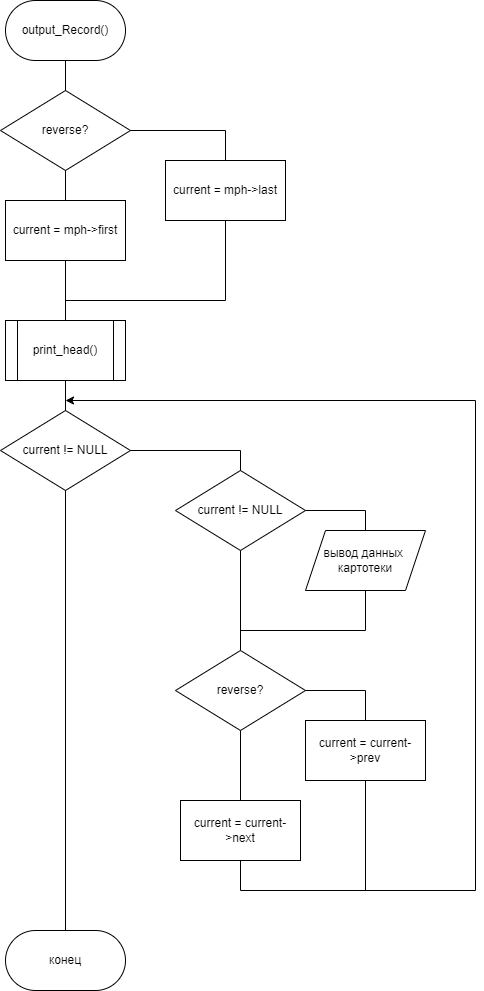
****

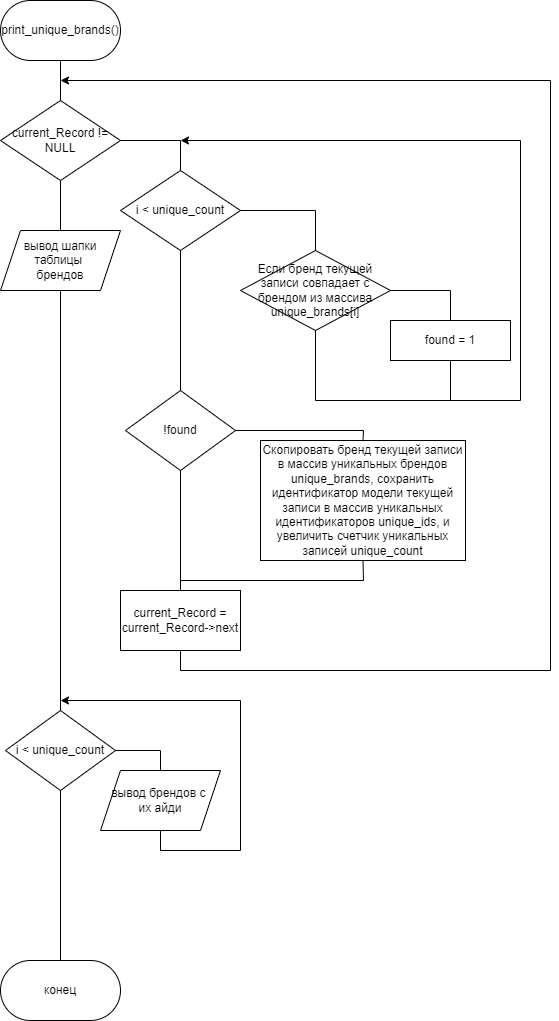
****

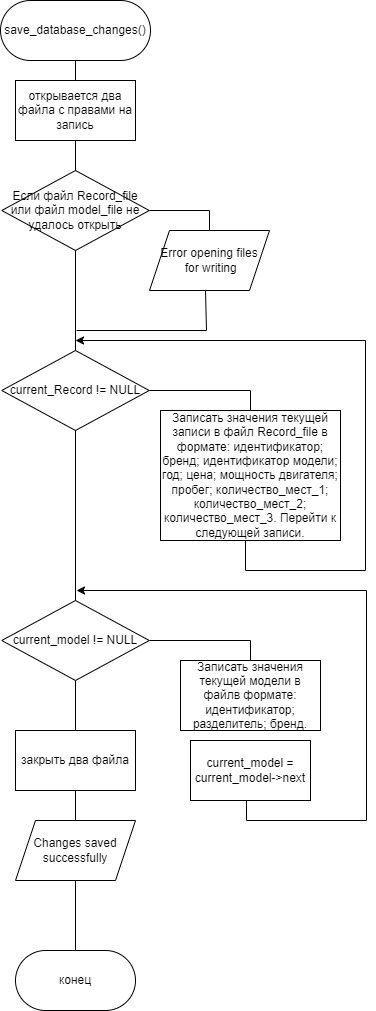
****

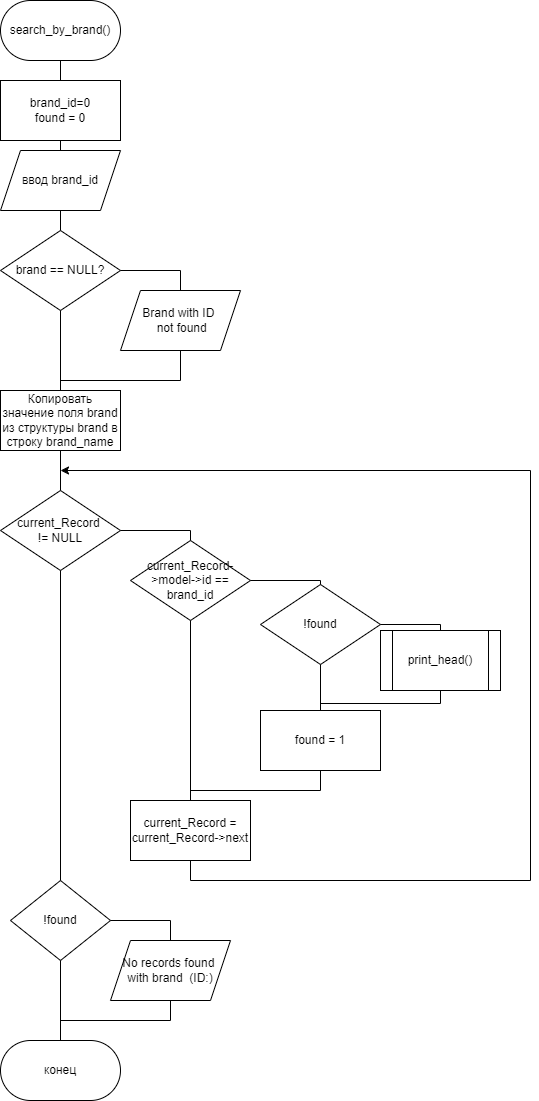
****

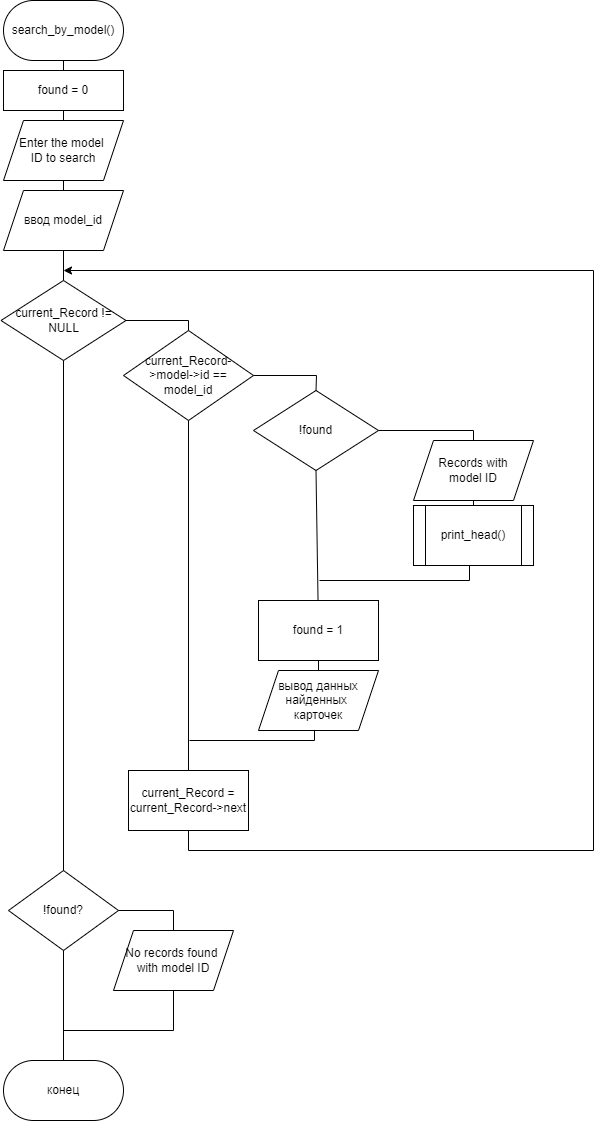
****

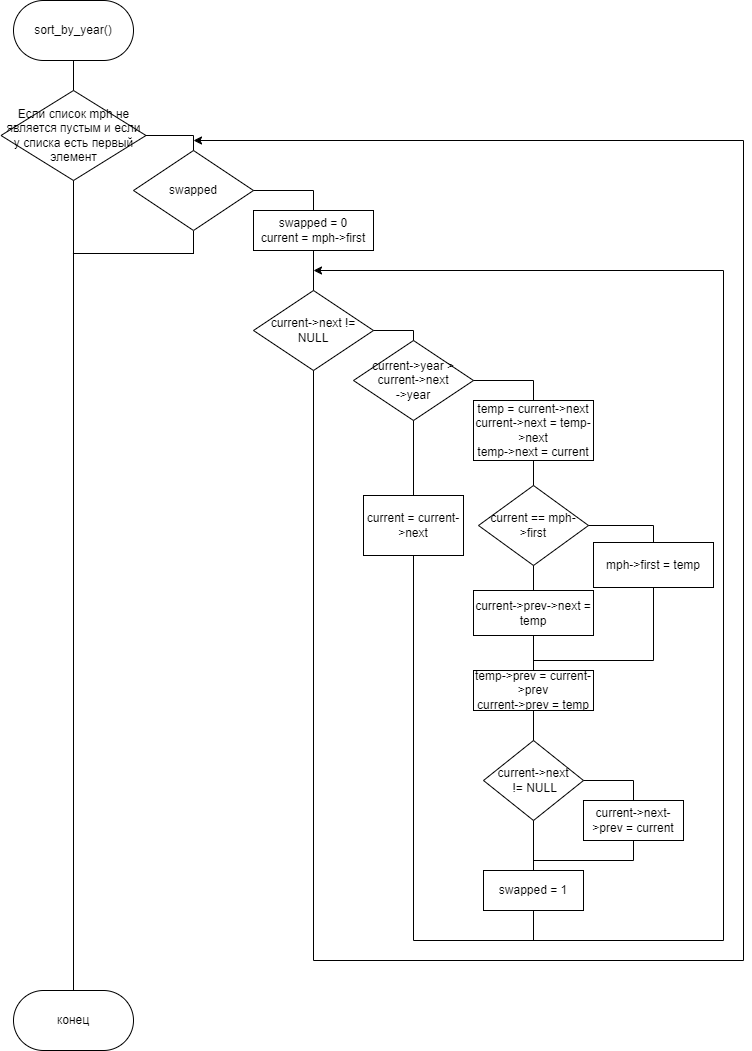
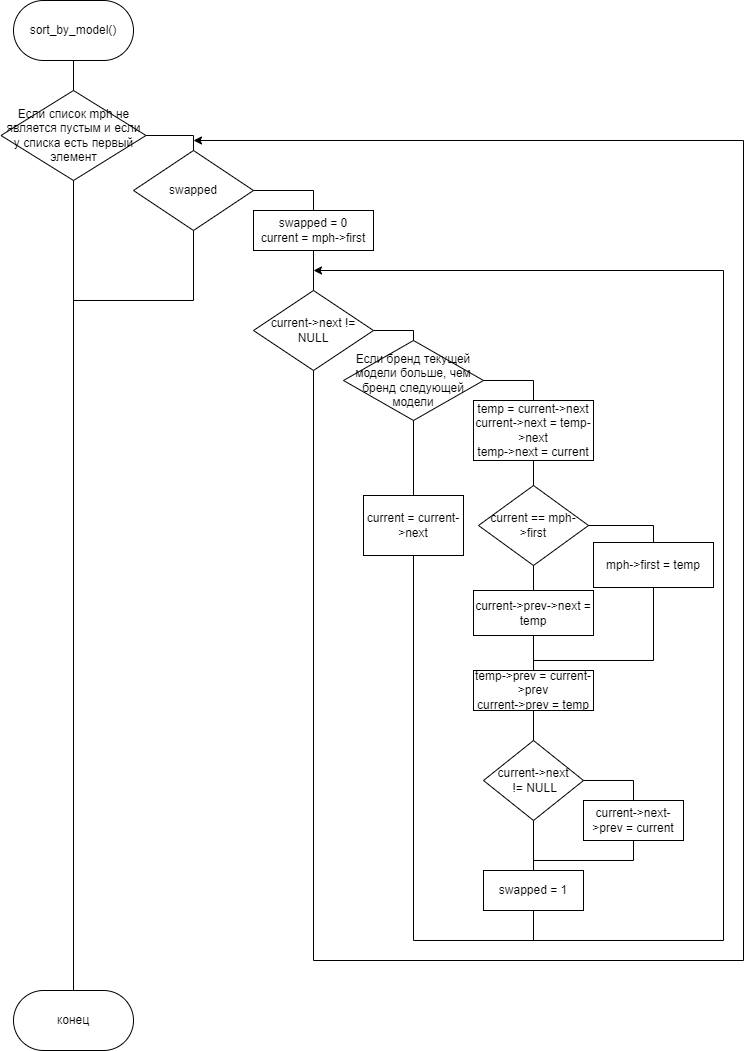
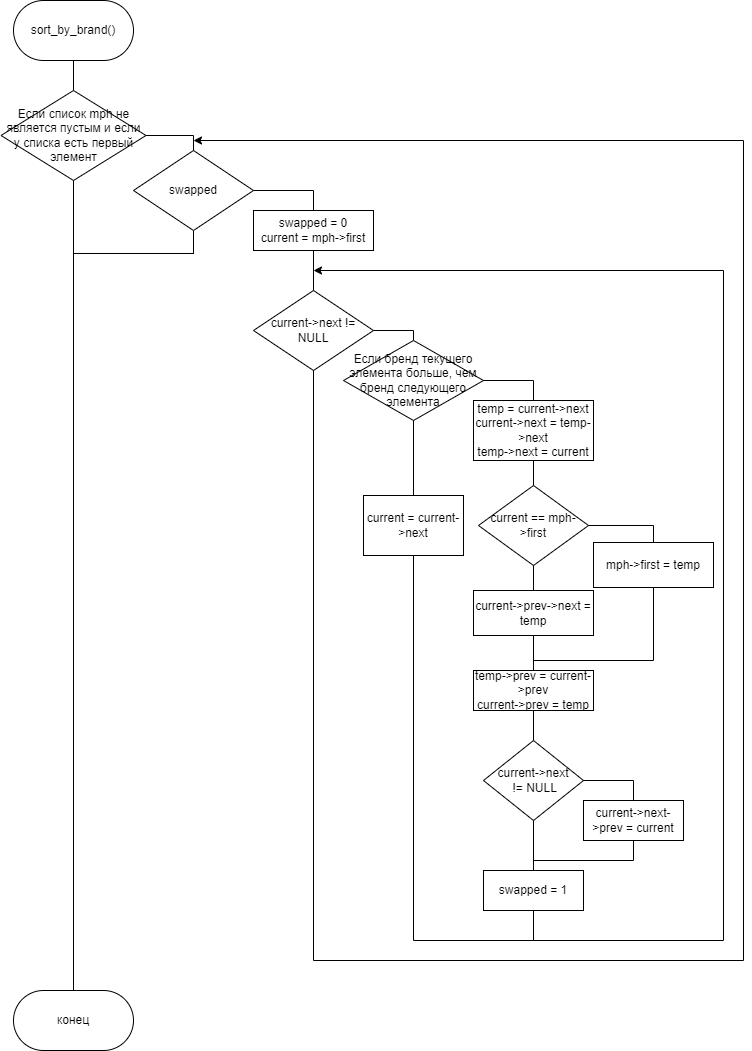
****

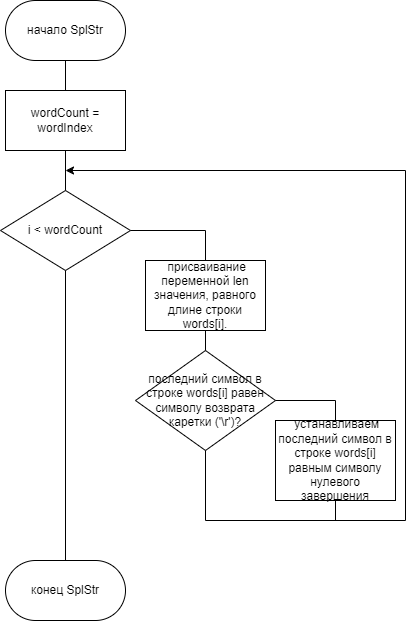
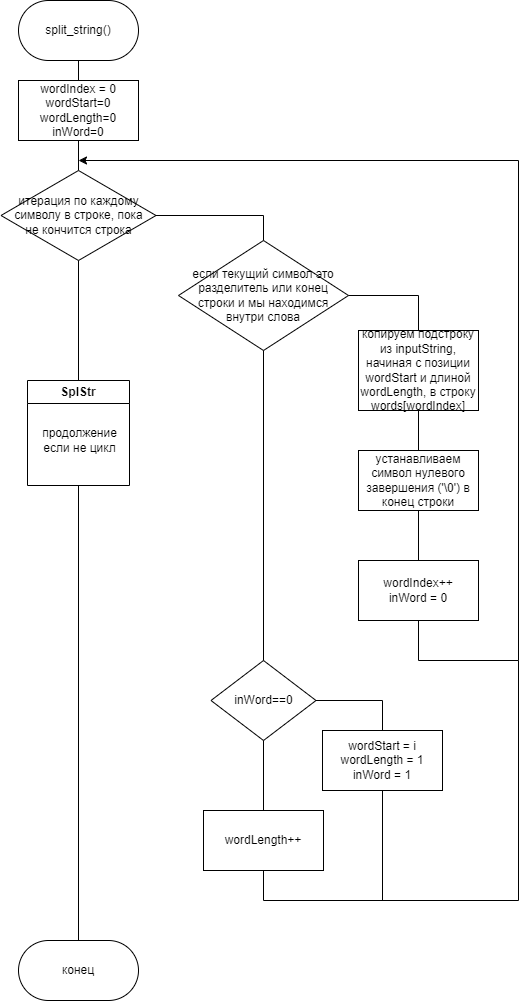
****

****

****

****

****

****

**Текст программы:**

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <string.h>**

**#include <ctype.h>**

**#define maxlen 128**

**/\* Structure ------------------------------------------------------------------- \*/**

**struct Record; /\* Define structure of Record \*/**

**struct Record\_head; /\* Define structure of Record List Head\*/**

**struct model; /\* Define structure of model \*/**

**struct model\_head; /\* Define structure of model List Head \*/**

**typedef struct Record MOV; /\* Define Structured Type of Record \*/**

**typedef struct Record\_head MHD; /\* Define Structured Type of Record List Head\*/**

**typedef struct model DIR; /\* Define Structured Type of model \*/**

**typedef struct model\_head DHD; /\* Define Structured Type of model List Head\*/**

**/\* Node and List --------------------------------------------------------------- \*/**

**MHD \*make\_Record\_head(); /\* Record Head Initialization \*/**

**DHD \*make\_model\_head(); /\* model Head Initialization \*/**

**DIR \*find\_model(DHD \*dph, int id); /\* Get pointer to Node in model List by ID \*/**

**MOV \*create\_Record(int id\_mov, char \*Record\_brand, int id, int Record\_year, int Record\_price, float Record\_enginePower, float Record\_mileage, int watch\_seats[3], DHD \*dph); /\* Record Node Initialization \*/**

**DIR \*create\_model(int id, char \*Record\_model); /\* model Node Initialization \*/**

**void add\_Record(MHD \*ph, MOV \*new\_node, MOV \*current\_node); /\* Add new Record Node to List \*/**

**void add\_model(DHD \*dph, DIR \*new\_node, DIR \*current\_node); /\* Add new model Node to List \*/**

**void delete\_Record(MHD \*mph, MOV \*current\_Record); /\* Delete selected Record Node \*/**

**void delete\_model(DHD \*dph, MHD \*mph, DIR \*current\_model); /\* Delete selected model Node \*/**

**void save\_database\_changes(char \*Record\_filename, char \*model\_filename, MHD \*mph, DHD \*dph, char sep);**

**/\* Interface --------------------------------------------------------------------\*/**

**void menu(); /\* Output main menu \*/**

**void print\_head(int n); /\* Output title of sheet (0 - for Record LIST, 1 - for model LIST) \*/**

**void output\_Record(MHD \*mph, int n, int reverse); /\* Output list with MOV structure (n for PRINT HEAD) \*/**

**void output\_model(DHD \*dph, int n, int reverse); /\* Output list with DIR structure (n for PRINT HEAD) \*/**

**void clear\_screen(); /\* Clear the console \*/**

**void search\_menu(MHD \*mph, DHD \*dph);**

**/\* Sort, Form and other Stuff \*/**

**void split\_string(char \*inputString, char \*\*words, int \*wordCount, char delimiter); /\* Split string by separator \*/**

**void add\_Record\_to\_list(char \*filebrand, MHD \*mph, char sep, DHD \*dph); /\* Adding the Record data of file to list \*/**

**void add\_model\_to\_list(char \*filebrand, DHD \*dph, char sep); /\* Adding the model data of file to list \*/**

**void add\_Record\_to\_file(char \*filebrand, MHD \*mph, char sep, DHD \*dph);/\*write to file\*/**

**int add\_Model\_to\_file(char \*filemodel, DHD \*dph);/\*add a new model to the list or select an existing one\*/**

**void delete\_Record\_by\_ID(MHD \*mph, int id);/\*removal from database by ID\*/**

**void search\_by\_model(MHD \*mph, DHD \*dph);/\*\* Search records by model name \*/**

**void search\_by\_brand(MHD \*mph, DHD \*dph);/\*\* Search records by brand name \*/**

**void Sort\_menu(MHD \*mph, DHD \*dph);/\*\* Display sorting menu and perform sorting based on user selection \*/**

**void sort\_by\_year(MHD \*mph);/\*\* Sort records by year \*/**

**void sort\_by\_brand(MHD \*mph);/\*\* Sort records by brand name \*/**

**void sort\_by\_model(MHD \*mph);/\*\* Sort records by model name \*/**

**void edit\_menu(MHD \*mph, DHD \*dph);**

**void edit\_Record(MHD \*mph, DHD \*dph);**

**void edit\_model(DHD \*dph);**

**void print\_unique\_brands(MHD \*mph);**

**void add\_new\_model\_to\_list(DHD \*dph);**

**void delete\_model\_by\_ID(DHD \*dph, MHD \*mph, int id);**

**/\* Main Program \*/**

**/\*\*/**

**int main() {/\*block complete\*/**

**char save\_choice;**

**char sep=';';**

**int option;**

**int id\_to\_delete,model\_id\_to\_delete;**

**DHD \*dph;**

**MHD \*mph;**

**dph=make\_model\_head();**

**mph=make\_Record\_head();**

**add\_model\_to\_list("struct-data2-win.txt",dph,sep);**

**add\_Record\_to\_list("struct-data-win.txt",mph,sep,dph);**

**do{**

**menu();**

**scanf("%i", &option);**

**switch(option){**

**case 0:{**

**puts("\nYour selection is EXIT");**

**printf("Do you want to save the changes to the database before exiting? (Y/N): ");**

**scanf(" %c", &save\_choice);**

**if (toupper(save\_choice) == 'Y') {**

**save\_database\_changes("struct-data-win.txt", "struct-data2-win.txt", mph, dph, sep);**

**printf("Changes saved successfully.\n");**

**} else {**

**printf("Changes were not saved.\n");**

**}**

**exit(0);**

**}**

**case 1:{**

**puts("\nYour selection is SHOW THE Record DATA\n");**

**getchar();**

**output\_Record(mph,0,0);**

**break;**

**}**

**case 2:{**

**puts("\nYour selection is SHOW THE model DATA\n");**

**getchar();**

**output\_model(dph,1,0);**

**break;**

**}**

**case 3: {**

**puts("\nYour selection is ADD DATA TO FILE\n");**

**getchar();**

**add\_Record\_to\_file("struct-data-win.txt", mph, sep, dph);**

**break;**

**}**

**case 4:{**

**puts("\nYour selection is DELETE RECORD BY ID\n");**

**output\_Record(mph,0,0);**

**printf("Enter the ID of the record you want to delete:");**

**scanf("%d", &id\_to\_delete);**

**delete\_Record\_by\_ID(mph, id\_to\_delete);**

**printf("\nUpdated database:\n");**

**output\_Record(mph,0,0);**

**getchar();**

**break;**

**}**

**case 5: {**

**puts("\nYour selection is SEARCH MENU\n");**

**search\_menu(mph, dph);**

**getchar();**

**break;**

**}**

**case 6: {**

**puts("\nYour selection is SORT MENU\n");**

**Sort\_menu(mph, dph);**

**break;**

**}**

**case 7: {**

**puts("\nYour selection is EDIT DATA\n");**

**getchar();**

**edit\_menu(mph, dph);**

**break;**

**}**

**case 8:{**

**puts("\nYour selection is SHOW THE DIRECTORY\n");**

**getchar();**

**puts("This program is a file cabinet. Performs the functions presented in the menu.\nAuto have been selected as the subject area.\nDesignations: brand- the name of the auto,\nmodel - the model of the car, year - the year of the care,\nprice - the price of the car, enginePower - enginePower of the car,\nmileage - mileage of the car,\nseats - the seats of the car.");**

**break;**

**}**

**case 9: {**

**puts("\nYour selection is ADD NEW MODEL TO LIST\n");**

**getchar();**

**add\_new\_model\_to\_list(dph);**

**break;**

**}**

**case 10: {**

**puts("\nYour selection is REMOVE MODEL FROM LIST\n");**

**output\_model(dph, 1, 0);**

**printf("Enter the ID of the model you want to delete: ");**

**scanf("%d", &model\_id\_to\_delete);**

**delete\_model\_by\_ID(dph, mph, model\_id\_to\_delete);**

**break;**

**}**

**default:{**

**puts("\nIncorrect key");**

**getchar();**

**}**

**}**

**puts("\nPress ENTER to continue");**

**getchar();**

**clear\_screen();**

**} while (option!=0);**

**return 0;**

**}**

**/\* Functions and their description ---------------------------------------------- \*/**

**struct Record{**

**int id;**

**char \*brand;**

**DIR \*model;**

**int year;**

**int price;**

**float enginePower;**

**float mileage;**

**int seats[3];**

**struct Record \*prev;**

**struct Record \*next;**

**};**

**struct Record\_head{**

**struct Record \*first;**

**struct Record \*last;**

**};**

**struct model{**

**int id; /\* ID of the model \*/**

**char \*brand; /\* model brand \*/**

**struct model \*next; /\* Link to previous node \*/**

**struct model \*prev; /\* Link to next node \*/**

**};**

**struct model\_head{**

**struct model \*first;**

**struct model \*last;**

**};**

**typedef struct {**

**int id;**

**char brand[100];**

**} UniqueBrand;**

**MHD \*make\_Record\_head(){/\*block complete\*/**

**MHD \*ph=NULL; /\* Define and init Head \*/**

**ph=(MHD\*)malloc(sizeof(MHD));**

**ph->first=NULL;**

**ph->last=NULL;**

**return ph;**

**}**

**DHD \*make\_model\_head(){/\*block complete\*/**

**DHD \*ph=NULL; /\* Define and init Head \*/**

**ph=(DHD\*)malloc(sizeof(DHD));**

**ph->first=NULL;**

**ph->last=NULL;**

**return ph;**

**}**

**DIR \*find\_model(DHD \*dph, int id){/\*block complete\*/**

**DIR \*current = NULL;**

**int flag=0;**

**current = dph->first;**

**while(current!=NULL && flag==0){**

**if(current->id==id){**

**flag=1;**

**} else current=current->next;**

**}**

**return current;**

**}**

**MOV \*create\_Record(int id\_mov, char \*Record\_brand, int id, int Record\_year, int Record\_price, float Record\_enginePower, float Record\_mileage, int watch\_seats[3], DHD \*dph){/\*block complete\*/**

**MOV \*new\_Record = NULL; /\* Pointer to new node \*/**

**DIR \*Record\_model = NULL; /\* Pointer to model of Record \*/**

**char \*brand = NULL;**

**new\_Record = (MOV\*)malloc(sizeof(MOV));**

**brand = (char\*)malloc((strlen(Record\_brand) + 1) \* sizeof(char));**

**Record\_model = (DIR \*)malloc(sizeof(DIR));**

**Record\_model=find\_model(dph, id);**

**if (new\_Record && brand) { /\* Data is not empty \*/**

**new\_Record->id = id\_mov;**

**new\_Record->brand = brand;**

**new\_Record->model = Record\_model;**

**new\_Record->year = Record\_year;**

**new\_Record->price = Record\_price;**

**new\_Record->enginePower = Record\_enginePower;**

**new\_Record->mileage = Record\_mileage;**

**memcpy(brand, Record\_brand, strlen(Record\_brand) + 1);**

**memcpy(new\_Record->seats, watch\_seats, sizeof(new\_Record->seats));**

**new\_Record->next = NULL;**

**new\_Record->prev = NULL;**

**}**

**return new\_Record; /\* Return adress of node \*/**

**}**

**DIR \*create\_model(int id, char \*Record\_model){/\*block complete\*/**

**DIR \*new\_model = NULL; /\* Pointer to new node \*/**

**char \*model = NULL;**

**new\_model = (DIR\*)malloc(sizeof(DIR));**

**model = (char\*)malloc((strlen(Record\_model) + 1) \* sizeof(char));**

**if (new\_model && model){ /\* Data is not empty \*/**

**new\_model->id = id;**

**new\_model->brand = model;**

**memcpy(model, Record\_model, strlen(Record\_model) + 1);**

**new\_model->prev = NULL;**

**new\_model->next = NULL;**

**model[strlen(Record\_model)] = '\0';**

**}**

**return new\_model; /\* return adress of node \*/**

**}**

**void add\_Record(MHD \*ph, MOV \*new\_node, MOV \*current\_node){/\*block complete\*/**

**if (ph && new\_node) {**

**if (current\_node == NULL) { /\* Add first node of list \*/**

**ph->first = new\_node;**

**ph->last = new\_node;**

**new\_node->prev = NULL;**

**new\_node->next = NULL;**

**} else {**

**current\_node->next = new\_node;**

**new\_node->prev = current\_node;**

**new\_node->next = NULL;**

**ph->last = new\_node;**

**}**

**}**

**}**

**void add\_model(DHD \*dph, DIR \*new\_node, DIR \*current\_node){/\*block complete\*/**

**if (dph && new\_node) {**

**if (current\_node == NULL) { /\* Add first node of list \*/**

**dph->first = new\_node;**

**dph->last = new\_node;**

**new\_node->prev = NULL;**

**new\_node->next = NULL;**

**} else {**

**current\_node->next = new\_node;**

**new\_node->prev = current\_node;**

**new\_node->next = NULL;**

**dph->last = new\_node;**

**}**

**}**

**}**

**void delete\_Record(MHD \*mph, MOV \*current\_Record) {/\*block complete\*/**

**if (current\_Record == mph->first) { /\* If deleted node is the first in the list \*/**

**mph->first = current\_Record->next;**

**if (mph->first) {**

**mph->first->prev = NULL;**

**} else { /\* If deleted node is alone in the list \*/**

**mph->last = NULL;**

**}**

**} else if (current\_Record == mph->last) { /\* If deleted node is the last in the list \*/**

**mph->last = current\_Record->prev;**

**if (mph->last) {**

**mph->last->next = NULL;**

**} else { /\* If deleted node is alone in the list \*/**

**mph->first = NULL;**

**}**

**} else { /\* If deleted node not first or last of the list \*/**

**current\_Record->prev->next = current\_Record->next;**

**current\_Record->next->prev = current\_Record->prev;**

**}**

**current\_Record->next = NULL;**

**current\_Record->prev = NULL;**

**free(current\_Record);**

**}**

**void delete\_model(DHD \*dph, MHD \*mph, DIR \*current\_model) {/\*block complete\*/**

**MOV \*current\_Record = mph->first;**

**MOV \*next\_Record = NULL;**

**while (current\_Record != NULL) {**

**next\_Record = current\_Record->next;**

**if (current\_Record->model == current\_model) {**

**delete\_Record(mph, current\_Record);**

**}**

**current\_Record = next\_Record;**

**}**

**if (current\_model == dph->first) {**

**dph->first = current\_model->next;**

**if (dph->first) {**

**dph->first->prev = NULL;**

**} else {**

**dph->last = NULL;**

**}**

**} else if (current\_model== dph->last) {**

**dph->last = current\_model->prev;**

**if (dph->last) {**

**dph->last->next = NULL;**

**} else {**

**dph->first = NULL;**

**}**

**} else {**

**current\_model->prev->next = current\_model->next;**

**current\_model->next->prev = current\_model->prev;**

**}**

**current\_model->next = NULL;**

**current\_model->prev = NULL;**

**free(current\_model);**

**}**

**void menu(){**

**puts("Choose the option");**

**puts("0 - for EXIT program");**

**puts("1 - for show the Record data");**

**puts("2 - for show the model data");**

**puts("3 - add data to database");**

**puts("4 - delete data from the database");**

**puts("5 - search menu");**

**puts("6 - sort menu");**

**puts("7 - edit data menu");**

**puts("8 - SHOW directory");**

**puts("9 - add new model to list");**

**puts("10 - remove model from list");**

**printf("Enter the option: ");**

**}**

**void print\_head(int n){**

**if(n==0){**

**printf("| %2s | %25s | %25s | %4s | %7s | %4s | %5s | %7s |\n","ID","brand","model","Year","Price","enginePower","mileage","Seats");**

**printf("+----+---------------------------+---------------------------+------+---------+-------------+---------+---------+\n");**

**} else {**

**printf("| %2s | %25s |\n","ID","model");**

**printf("+----+---------------------------+\n");**

**}**

**}**

**void output\_Record(MHD \*mph, int n, int reverse) {/\*block complete\*/**

**MOV \*current;**

**if (reverse)**

**current = mph->last;**

**else**

**current = mph->first;**

**print\_head(n);**

**while (current != NULL) {**

**if (current->model != NULL) {**

**printf("| %2d | %25s | %25s | %4d | %7d | %11.1f | %7.1f | %2d %2d %d |\n", current->id, current->brand, current->model->brand, current->year, current->price, current->enginePower, current->mileage, current->seats[0], current->seats[1], current->seats[2]);**

**}**

**if (reverse)**

**current = current->prev;**

**else**

**current = current->next;**

**}**

**}**

**void output\_model(DHD \*dph, int n, int reverse) {/\*block complete\*/**

**DIR \*current;**

**if (reverse)**

**current = dph->last;**

**else**

**current = dph->first;**

**print\_head(n);**

**while (current != NULL) {**

**printf("| %2d | %25s |\n", current->id, current->brand);**

**if (reverse)**

**current = current->prev;**

**else**

**current = current->next;**

**}**

**}**

**void clear\_screen(){**

**#if defined(\_WIN32) || defined(\_WIN64)**

**system("cls");**

**#else**

**system("clear");**

**#endif**

**}**

**void split\_string(char \*inputString, char \*\*words, int \*wordCount, char delimiter) {/\*block complete\*/**

**int wordIndex = 0, wordStart=0, wordLength=0, inWord=0, i;**

**for (i = 0; i <= strlen(inputString); i++) {**

**if ((inputString[i] == delimiter || inputString[i] == '\0')&&(inWord==1)) {**

**words[wordIndex] = (char \*)malloc(wordLength + 1);**

**strncpy(words[wordIndex], inputString + wordStart, wordLength);**

**words[wordIndex][wordLength] = '\0';**

**wordIndex++;**

**inWord = 0;**

**} else {**

**if (inWord==0) {**

**wordStart = i;**

**wordLength = 1;**

**inWord = 1;**

**} else {**

**wordLength++;**

**}**

**}**

**}**

**\*wordCount = wordIndex;**

**for (i = 0; i < \*wordCount; i++) {**

**int len = strlen(words[i]);**

**if (words[i][len - 1] == '\r') {**

**words[i][len - 1] = '\0';**

**}**

**}**

**}**

**void add\_Record\_to\_list(char \*filebrand, MHD \*mph, char sep, DHD \*dph){/\*block complete\*/**

**char line[maxlen], \*words[10];**

**int wordCount, seats[3], i;**

**MOV \*new\_Record;**

**FILE \*file = fopen(filebrand, "r");**

**if (file == NULL) printf("Error opening file.\n");**

**while (fgets(line, sizeof(line), file)){**

**line[strcspn(line, "\n")] = 0;**

**split\_string(line, words, &wordCount, sep);**

**if (wordCount == 10){**

**int id\_mov = atoi(words[0]);**

**char \*brand = words[1];**

**int id = atoi(words[2]);**

**int year = atoi(words[3]);**

**int price = atoi(words[4]);**

**float enginePower = atof(words[5]);**

**float mileage = atof(words[6]);**

**seats[0] = atoi(words[7]);**

**seats[1] = atoi(words[8]);**

**seats[2] = atoi(words[9]);**

**new\_Record = create\_Record(id\_mov, brand, id, year, price, enginePower, mileage, seats, dph);**

**add\_Record(mph, new\_Record, mph->last);**

**}**

**else printf("Invalid number of attributes in line: %s\n", line);**

**for (i = 0; i < wordCount; i++){**

**free(words[i]);**

**}**

**}**

**fclose(file);**

**}**

**void add\_model\_to\_list(char \*filebrand, DHD \*dph, char sep){/\*block complete\*/**

**char line[maxlen], \*words[2];**

**int wordCount, i;**

**DIR \*new\_model;**

**FILE \*file = fopen(filebrand, "r");**

**if (file == NULL) printf("Error opening file.\n");**

**while (fgets(line, sizeof(line), file)){**

**line[strcspn(line, "\n")] = 0;**

**split\_string(line, words, &wordCount, sep);**

**if (wordCount == 2){**

**int id = atoi(words[0]);**

**char \*brand = words[1];**

**new\_model = create\_model(id, brand);**

**add\_model(dph, new\_model, dph->last);**

**}**

**else printf("Invalid number of attributes in line: %s\n", line);**

**for (i = 0; i < wordCount; i++){**

**free(words[i]);**

**}**

**}**

**fclose(file);**

**}**

**void add\_Record\_to\_file(char \*filebrand, MHD \*mph, char sep, DHD \*dph){/\*block complete\*/**

**int id\_mov=1, id, year, price, seats[3];**

**float enginePower, mileage;**

**char brand[100];**

**FILE \*file = fopen(filebrand, "a");**

**if (file == NULL) {**

**printf("Error opening file.\n");**

**}**

**if (mph->last != NULL) {**

**id\_mov = mph->last->id + 1;**

**}**

**printf("Enter brand: ");**

**scanf("%s", brand);**

**/\*add a new model to the list or select an existing one\*/**

**printf("Model ID which are in the database:\n");**

**output\_model(dph,1,0);**

**printf("Enter 0 to add a new model or select existing model ID: ");**

**scanf("%d", &id);**

**if(id==0){**

**id = add\_Model\_to\_file("struct-data2-win.txt", dph);**

**}**

**/\*------------------------------------------------------------------\*/**

**printf("Enter year: ");**

**scanf("%d", &year);**

**printf("Enter price: ");**

**scanf("%d", &price);**

**printf("Enter engine power: ");**

**scanf("%f", &enginePower);**

**printf("Enter mileage: ");**

**scanf("%f", &mileage);**

**printf("Enter seats (3 integers separated by spaces): ");**

**scanf("%d %d %d", &seats[0], &seats[1], &seats[2]);**

**/\*write to file\*/**

**fprintf(file, "%d;%s;%d;%d;%d;%.3f;%.3f;%d;%d;%d\n", id\_mov, brand, id, year, price, enginePower, mileage, seats[0], seats[1], seats[2]);**

**fclose(file);**

**printf("Data added successfully.\n");**

**}**

**int add\_Model\_to\_file(char \*filemodel, DHD \*dph) {/\*block complete\*/**

**int id;**

**char model[100];**

**FILE \*file = fopen(filemodel, "a");**

**if (file == NULL) {**

**printf("Error opening file.\n");**

**}**

**printf("Enter model name: ");**

**scanf("%s", model);**

**if (dph->last != NULL) {**

**id = dph->last->id + 1;**

**} else {**

**id = 1;**

**}**

**fprintf(file, "%d;%s\n", id, model);**

**fclose(file);**

**printf("Model added successfully.\n");**

**return id;**

**}**

**void delete\_model\_by\_ID(DHD \*dph, MHD \*mph, int id) {/\*block complete\*/**

**DIR \*current\_model = dph->first;**

**while (current\_model != NULL) {**

**if (current\_model->id == id) {**

**delete\_model(dph, mph, current\_model);**

**printf("Model with ID '%d' deleted successfully.\n", id);**

**}**

**current\_model = current\_model->next;**

**}**

**printf("Model with ID '%d' not found in the list.\n", id);**

**}**

**void delete\_Record\_by\_ID(MHD \*mph, int id) {/\*block complete\*/**

**MOV \*current\_Record = mph->first;**

**if (mph->first == NULL) {**

**printf("Database is empty. No records to delete.\n");**

**} else {**

**while (current\_Record != NULL) {**

**if (current\_Record->id == id) {**

**MOV \*next\_Record = current\_Record->next;**

**delete\_Record(mph, current\_Record);**

**current\_Record = next\_Record;**

**} else {**

**current\_Record = current\_Record->next;**

**}**

**}**

**}**

**}**

**void search\_menu(MHD \*mph, DHD \*dph) {**

**int search\_option;**

**puts("Select an option to search:");**

**puts("1 - Search by model");**

**puts("2 - Search by brand");**

**printf("Enter option: ");**

**scanf("%i", &search\_option);**

**getchar();**

**switch(search\_option) {**

**case 1:**

**puts("\nYour selection is SEARCH DATA BY MODEL\n");**

**output\_model(dph, 1, 0);**

**search\_by\_model(mph, dph);**

**break;**

**case 2:**

**puts("\nYour selection is SEARCH DATA BY BRAND\n");**

**print\_unique\_brands(mph);**

**search\_by\_brand(mph, dph);**

**break;**

**default:**

**puts("Invalid option");**

**}**

**}**

**void search\_by\_model(MHD \*mph, DHD \*dph) {/\*block complete\*/**

**int found = 0;**

**MOV \*current\_Record = mph->first;**

**int model\_id;**

**printf("Enter the model ID to search: ");**

**scanf("%d", &model\_id);**

**getchar();**

**while (current\_Record != NULL) {**

**if (current\_Record->model->id == model\_id) {**

**if (!found) {**

**printf("\nRecords with model ID '%d':\n", model\_id);**

**print\_head(0);**

**}**

**found = 1;**

**printf("| %2d | %25s | %25s | %4d | %7d | %11.1f | %7.1f | %2d %2d %d |\n", current\_Record->id, current\_Record->brand, current\_Record->model->brand, current\_Record->year, current\_Record->price, current\_Record->enginePower, current\_Record->mileage, current\_Record->seats[0], current\_Record->seats[1], current\_Record->seats[2]);**

**}**

**current\_Record = current\_Record->next;**

**}**

**if (!found) {**

**printf("No records found with model ID '%d'.\n", model\_id);**

**}**

**}**

**void search\_by\_brand(MHD \*mph, DHD \*dph) {/\*block complete\*/**

**int brand\_id=0;**

**DIR \*brand = find\_model(dph, brand\_id);**

**MOV \*current\_Record = mph->first;**

**int found = 0;**

**char brand\_name[100];**

**printf("Enter the brand ID to search: ");**

**scanf("%d", &brand\_id);**

**if (brand == NULL) {**

**printf("Brand with ID '%d' not found.\n", brand\_id);**

**}**

**strcpy(brand\_name, brand->brand);**

**while (current\_Record != NULL) {**

**if (current\_Record->model->id == brand\_id) {**

**if (!found) {**

**printf("\nRecords with brand '%s':\n", brand\_name);**

**print\_head(0);**

**}**

**found = 1;**

**printf("| %2d | %25s | %25s | %4d | %7d | %11.1f | %7.1f | %2d %2d %d |\n", current\_Record->id, current\_Record->brand, current\_Record->model->brand, current\_Record->year, current\_Record->price, current\_Record->enginePower, current\_Record->mileage, current\_Record->seats[0], current\_Record->seats[1], current\_Record->seats[2]);**

**}**

**current\_Record = current\_Record->next;**

**}**

**if (!found) {**

**printf("No records found with brand '%s' (ID: %d).\n", brand\_name, brand\_id);**

**}**

**}**

**void Sort\_menu(MHD \*mph, DHD \*dph) {**

**int search\_option;**

**puts("Select an option to search:");**

**puts("1 - Sort by model");**

**puts("2 - Sort by brand");**

**puts("3 - Sort by year");**

**printf("enter option: ");**

**scanf("%i", &search\_option);**

**getchar();**

**switch(search\_option) {**

**case 1:**

**puts("\nYour selection is SORT DATA BY MODEL\n");**

**sort\_by\_model(mph);**

**break;**

**case 2:**

**puts("\nYour selection is SORT DATA BY BRAND\n");**

**sort\_by\_brand(mph);**

**break;**

**case 3:**

**puts("\nYour selection is SORT DATA BY YEAR\n");**

**sort\_by\_year(mph);**

**output\_Record(mph,0,0);**

**break;**

**default:**

**puts("Invalid option");**

**}**

**}**

**void sort\_by\_year(MHD \*mph) {/\*block complete\*/**

**MOV \*current, \*temp;**

**int swapped = 0;**

**if (mph != NULL && mph->first != NULL) {**

**do {**

**swapped = 0;**

**current = mph->first;**

**while (current->next != NULL) {**

**if (current->year > current->next->year) {**

**temp = current->next;**

**current->next = temp->next;**

**temp->next = current;**

**if (current == mph->first) {**

**mph->first = temp;**

**} else {**

**current->prev->next = temp;**

**}**

**temp->prev = current->prev;**

**current->prev = temp;**

**if (current->next != NULL) {**

**current->next->prev = current;**

**}**

**swapped = 1;**

**} else {**

**current = current->next;**

**}**

**}**

**} while (swapped);**

**}**

**}**

**void sort\_by\_brand(MHD \*mph) {/\*block complete\*/**

**MOV \*current, \*temp;**

**int swapped = 0;**

**if (mph != NULL && mph->first != NULL) {**

**do {**

**swapped = 0;**

**current = mph->first;**

**while (current->next != NULL) {**

**if (strcmp(current->brand, current->next->brand) > 0) {**

**temp = current->next;**

**current->next = temp->next;**

**temp->next = current;**

**if (current == mph->first) {**

**mph->first = temp;**

**} else {**

**current->prev->next = temp;**

**}**

**temp->prev = current->prev;**

**current->prev = temp;**

**if (current->next != NULL) {**

**current->next->prev = current;**

**}**

**swapped = 1;**

**} else {**

**current = current->next;**

**}**

**}**

**} while (swapped);**

**}**

**}**

**void sort\_by\_model(MHD \*mph) {/\*block complete\*/**

**MOV \*current, \*temp;**

**int swapped;**

**if (mph != NULL && mph->first != NULL) {**

**do {**

**swapped = 0;**

**current = mph->first;**

**while (current->next != NULL) {**

**if (strcmp(current->model->brand, current->next->model->brand) > 0) {**

**temp = current->next;**

**current->next = temp->next;**

**temp->next = current;**

**if (current == mph->first) {**

**mph->first = temp;**

**} else {**

**current->prev->next = temp;**

**}**

**temp->prev = current->prev;**

**current->prev = temp;**

**if (current->next != NULL) {**

**current->next->prev = current;**

**}**

**swapped = 1;**

**} else {**

**current = current->next;**

**}**

**}**

**} while (swapped);**

**}**

**}**

**void edit\_menu(MHD \*mph, DHD \*dph) {**

**int edit\_option,model\_id\_to\_delete;**

**puts("Select an option to edit:");**

**puts("1 - Edit Record");**

**puts("2 - Edit model");**

**puts("3 - delete model");**

**printf("Enter option: ");**

**scanf("%i", &edit\_option);**

**getchar();**

**switch (edit\_option) {**

**case 1:**

**edit\_Record(mph, dph);**

**break;**

**case 2:**

**edit\_model(dph);**

**break;**

**case 3:**

**output\_model(dph,1,0);**

**printf("Enter the ID of the model you want to delete: ");**

**scanf("%d", &model\_id\_to\_delete);**

**break;**

**default:**

**puts("Invalid option");**

**}**

**}**

**void add\_new\_model\_to\_list(DHD \*dph) {/\*block complete\*/**

**char model[100];**

**DIR \*new\_model;**

**int id;**

**printf("Enter model name: ");**

**scanf("%s", model);**

**if (dph->last != NULL) {**

**id = dph->last->id + 1;**

**} else {**

**id = 1;**

**}**

**new\_model = create\_model(id, model);**

**add\_model(dph, new\_model, dph->last);**

**output\_model(dph,1,0);**

**printf("the addition was successful");**

**getchar();**

**}**

**void edit\_Record(MHD \*mph, DHD \*dph) {/\*block complete\*/**

**char new\_brand[100];**

**int new\_model\_id, id\_to\_edit, edit\_line\_option,record\_found = 0;**

**MOV \*current\_Record = mph->first;**

**output\_Record(mph, 0, 0);**

**printf("Enter the ID of the record you want to edit: ");**

**scanf("%d", &id\_to\_edit);**

**getchar();**

**while (current\_Record != NULL) {**

**if (current\_Record->id == id\_to\_edit) {**

**record\_found = 1;**

**puts("Select an option to edit line:");**

**puts("1 - Edit brand");**

**puts("2 - Edit model");**

**puts("3 - Edit year");**

**puts("4 - Edit price");**

**puts("5 - Edit engine power");**

**puts("6 - Edit mileage");**

**puts("7 - Edit seats");**

**puts("8 - Edit all line");**

**printf("Enter option: ");**

**scanf("%i", &edit\_line\_option);**

**getchar();**

**switch (edit\_line\_option) {**

**case 1:**

**printf("Enter new brand: ");**

**fgets(new\_brand, 100, stdin);**

**new\_brand[strcspn(new\_brand, "\n")] = 0;**

**strcpy(current\_Record->brand, new\_brand);**

**break;**

**case 2:**

**output\_model(dph, 1, 0);**

**printf("Enter new model ID: ");**

**scanf("%d", &new\_model\_id);**

**getchar();**

**current\_Record->model = find\_model(dph, new\_model\_id);**

**break;**

**case 3:**

**printf("Enter new year: ");**

**scanf("%d", &(current\_Record->year));**

**getchar();**

**break;**

**case 4:**

**printf("Enter new price: ");**

**scanf("%d", &(current\_Record->price));**

**getchar();**

**break;**

**case 5:**

**printf("Enter new engine power: ");**

**scanf("%f", &(current\_Record->enginePower));**

**getchar();**

**break;**

**case 6:**

**printf("Enter new mileage: ");**

**scanf("%f", &(current\_Record->mileage));**

**getchar();**

**break;**

**case 7:**

**printf("Enter new seats (3 integers separated by spaces): ");**

**scanf("%d %d %d", &(current\_Record->seats[0]), &(current\_Record->seats[1]), &(current\_Record->seats[2]));**

**getchar();**

**break;**

**case 8:**

**printf("Enter new brand: ");**

**fgets(new\_brand, 100, stdin);**

**new\_brand[strcspn(new\_brand, "\n")] = 0;**

**strcpy(current\_Record->brand, new\_brand);**

**output\_model(dph, 1, 0);**

**printf("Enter new model ID: ");**

**scanf("%d", &new\_model\_id);**

**getchar();**

**current\_Record->model = find\_model(dph, new\_model\_id);**

**printf("Enter new year: ");**

**scanf("%d", &(current\_Record->year));**

**getchar();**

**printf("Enter new price: ");**

**scanf("%d", &(current\_Record->price));**

**getchar();**

**printf("Enter new engine power: ");**

**scanf("%f", &(current\_Record->enginePower));**

**getchar();**

**printf("Enter new mileage: ");**

**scanf("%f", &(current\_Record->mileage));**

**getchar();**

**printf("Enter new seats (3 integers separated by spaces): ");**

**scanf("%d %d %d", &(current\_Record->seats[0]), &(current\_Record->seats[1]), &(current\_Record->seats[2]));**

**getchar();**

**break;**

**default:**

**puts("Invalid option");**

**}**

**printf("Record updated successfully.\n");**

**}**

**current\_Record = current\_Record->next;**

**}**

**if (!record\_found) {**

**printf("Record with ID %d not found.\n", id\_to\_edit);**

**}**

**}**

**void edit\_model(DHD \*dph) {/\*block complete\*/**

**char new\_brand[100];**

**DIR \*current\_model = dph->first;**

**int id\_to\_edit, model\_found = 0;**

**output\_model(dph,1,0);**

**printf("Enter the ID of the model you want to edit: ");**

**scanf("%d", &id\_to\_edit);**

**getchar();**

**while (current\_model != NULL) {**

**if (current\_model->id == id\_to\_edit) {**

**model\_found = 1;**

**printf("Enter new model name: ");**

**fgets(current\_model->brand, 100, stdin);**

**current\_model->brand[strcspn(current\_model->brand, "\n")] = 0;**

**printf("Model updated successfully.\n");**

**}**

**current\_model = current\_model->next;**

**}**

**if (!model\_found) {**

**printf("Model with ID %d not found.\n", id\_to\_edit);**

**}**

**}**

**void save\_database\_changes(char \*Record\_filename, char \*model\_filename, MHD \*mph, DHD \*dph, char sep) {/\*block complete\*/**

**FILE \*Record\_file = fopen(Record\_filename, "w");**

**FILE \*model\_file = fopen(model\_filename, "w");**

**MOV \*current\_Record = mph->first;**

**DIR \*current\_model = dph->first;**

**if (Record\_file == NULL || model\_file == NULL) {**

**printf("Error opening files for writing.\n");**

**}**

**while (current\_Record != NULL) {**

**fprintf(Record\_file, "%d;%s;%d;%d;%d;%.3f;%.3f;%d;%d;%d\n",**

**current\_Record->id, current\_Record->brand, current\_Record->model->id,**

**current\_Record->year, current\_Record->price, current\_Record->enginePower,**

**current\_Record->mileage, current\_Record->seats[0], current\_Record->seats[1],**

**current\_Record->seats[2]);**

**current\_Record = current\_Record->next;**

**}**

**while (current\_model != NULL) {**

**fprintf(model\_file, "%d%c%s\n", current\_model->id, sep, current\_model->brand);**

**current\_model = current\_model->next;**

**}**

**fclose(Record\_file);**

**fclose(model\_file);**

**printf("Changes saved successfully.\n");**

**}**

**void print\_unique\_brands(MHD \*mph) {/\*block complete\*/**

**MOV \*current\_Record = mph->first;**

**char unique\_brands[maxlen][100];**

**int unique\_ids[maxlen];**

**int unique\_count = 0, i;**

**while (current\_Record != NULL) {**

**int found = 0;**

**for (i = 0; i < unique\_count; ++i) {**

**if (strcmp(current\_Record->brand, unique\_brands[i]) == 0) {**

**found = 1;**

**}**

**}**

**if (!found) {**

**strcpy(unique\_brands[unique\_count], current\_Record->brand);**

**unique\_ids[unique\_count] = current\_Record->model->id;**

**++unique\_count;**

**}**

**current\_Record = current\_Record->next;**

**}**

**printf("+----+-----------+\n");**

**printf("| ID | Brand |\n");**

**printf("+----+-----------+\n");**

**for (i = 0; i < unique\_count; ++i) {**

**printf("|%3d |%11s|\n", unique\_ids[i], unique\_brands[i]);**

**}**

**printf("+----+-----------+\n");**

**}**

**Заключение:**

Разработка электронной картотеки представляет собой важную задачу, позволяющую эффективно управлять и обрабатывать данные в различных предметных областях. В ходе выполнения данного задания была разработана программа для создания и управления электронной картотекой.

Программа включает в себя ряд функций, таких как добавление, редактирование, удаление, поиск и сортировка данных, что обеспечивает широкие возможности для работы с картотекой. Пользователю предоставляется удобное меню для выбора нужных действий, а данные могут быть многократно обработаны в процессе работы программы. Важным аспектом реализации программы является структурирование кода на отдельные функции, что обеспечивает легкость сопровождения и расширения программы в будущем.

Таким образом, разработка электронной картотеки с использованием предложенного подхода позволяет эффективно управлять и обрабатывать большие объемы данных, что делает её полезным инструментом в различных областях деятельности.

#### Использование заголовочных файлов стандартной библиотеки

1. **stdio.h**:
   * Использован для работы с функциями ввода-вывода, такими как **printf()**, **scanf()**, **puts()**, **getchar()**, и т.д., что позволяет взаимодействовать с пользователем через консоль.
2. **stdlib.h**:
   * Использован для управления динамической памятью через функции **malloc()**, **calloc()**, и **free()**.
   * Также использован для завершения программы с помощью функции **exit()**.
3. **string.h**:
   * Использован для работы с функциями обработки строк, такими как **strcpy()**, **strcat()**, **strcmp()**, и т.д., что позволяет манипулировать строками в программе.
4. **ctype.h**:
   * Использован для работы с функцией **toupper()**, которая позволяет преобразовать символ в верхний регистр, что необходимо для обработки пользовательского ввода (например, при запросе о сохранении изменений перед выходом).

### Выводы о достижении цели работы

Цель работы по разработке электронной картотеки была успешно достигнута. В ходе выполнения задачи была создана программа, которая позволяет:

* Добавлять, редактировать и удалять записи и модели.
* Искать записи по различным критериям.
* Сортировать записи по выбранным параметрам.
* Сохранять изменения в базу данных и загружать данные из файлов.

Программа предоставляет пользователю удобный интерфейс для выполнения всех необходимых операций с данными, что делает её эффективным инструментом для управления электронной картотекой.

Начало формы