САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра кафедра вычислительной техники

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: «Программная реализация электронной картотеки»

Студент гр. 9305 _____ Епифанцев Е.В.

Преподаватель _____ Перязева Ю.В.

Санкт-Петербург
2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ (КУРСОВОЙ ПРОЕКТ)

Студент Епифанцев Е.В.

Группа 9305

Тема работы: Программная реализация электронной картотеки.

Исходные данные:
Данные первоначально считываются из файла, в процессе работы данные вводятся с клавиатуры.

Содержание пояснительной записки:
Перечисляются требуемые разделы пояснительной записки (обязательны разделы «Содержание», «Введение», «Электронная картотека», «Программная реализация», «Описание функций», «Приложение А. Схема вызовов функций» , «Приложение Б. Блок-схемы функций», «Приложение В. Исходный код программы», «Список использованных источников»)

Предполагаемый объем пояснительной записки:
Не менее 60 страниц.

Дата выдачи задания: 01.04.2000

Дата сдачи реферата: 23.05.2020

Дата защиты реферата: 19.05.2000

Студент(ка) _____ Епифанцев Е.В.

Преподаватель _____ Перязева Ю.В.

АННОТАЦИЯ

Курсовая работа состоит из теоретического описания методов работы программы и самой программной реализации электронной картотеки. В первой части первого раздела описывается работы электронной картотеки. В первой части второго раздела описываются методы программной реализации. Во второй части второго раздела описывается структура проекта. В третьем разделе описываются функции для работы с файлом, меню и двусвязным списком. В четвертом разделе приведены примеры работы программы.

SUMMARY

Coursework consists of theoretical descriptions. The first part of the first section describes the work of an electronic file cabinet. The first part of the second section describes methods of software implementation. Project structure. The third section describes the functions for working with menus, menus, and doubly linked lists. The fourth section provides examples of the program.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью данной работы является программная реализация электронной картотеки, которая будет хранится на диске и с которой можно будет проводить 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 картотеки на диске

В процессе обработки картотека должная храниться в памяти компьютера в виде списков и массивов структур, связанных указателями.

В качестве предметной области были выбраны машины.

Для программной реализации была выбрана динамическая структура данных — двусвязый список.

1. ЭЛЕКТРОННАЯ КАРТОТЕКА

1.1. Описание работы

Картотека представляет собой таблицу с данными (1 столбец — название автомобиля, 2 — название компании производителя, 3 — год производства, 4 — цена, 5 — вес, 6 — пробег, 7 — минимально достигаемая скорость за 5 секунд, 8 — максимально достигаемая скорость за 5 секунд).

Для картотеки определены следующие операции: 1) Вывод картотеки на экран; 2) Добавление новой записи в таблицу; 3) Удаление записи из таблицы; 4) Поиск записей в таблице; 5) Сортировка записей; 6) Редактирование записей; 7) Обмен местами двух записей; 8) Сохранение картотеки.

Добавление записей происходит в то места, куда захочет пользователь, если картотека пустая, то запись добавится в начало таблицы.

Поиск записей организован не только по конкретным полям, но и по нескольким сразу.

Сортировка работает по желанию пользователя, например, по убыванию определенного поля.

Удаляются записи из таблицы, исходя, из номеров введенных пользователем.

Обмен двух записей работает только в том случае, если картотека не пустая, иначе программа предупреждает пользователя о том, что картотека пустая.

Редактируются только те записи и только те поля, которые выбирает пользователь.

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. Постановка задачи и описание способов ее решения.

Имеется электронная картотека, которая хранится в файле cart.csv, каждое поле записи в ней разделено точкой с запятой.

Требуется реализовать следующие функции для работы с картотекой:

- 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

В качестве предметной области были выбраны машины. Каждая запись в картотеки представляет собой следующие поля:

- Название автомобиля
- Название компании — производителя
- Год производства
- Цена
- Вес
- Пробег
- Максимальная скорость, достигаемая машиной за 5 сек.
- Минимальная скорость, достигаемая машиной за 5 сек.

В качестве структуры данных для хранения записей в картотеке был выбран двусвязный список.

В начале работы программы, открывается файл, где находится картотека, затем происходит формирование таблицы уже имеющимися в файле записями, если же файл оказывается пуст, список остается пустым.

Затем на экран пользователя выводится меню, которое содержит все реализованные функции для работы со списком.

Затем в зависимости от выбора пользователя в меню вызываются соответствующие функции и происходит работа программы.

Каждая манипуляции со списком затрагивает только «облачную» часть картотеки, то есть измененная картотека не будет сохранена, после выполнения какой — либо функции. Для того, чтобы пользователю сохранить измененную картотеку достаточно выбрать 8 пункт в меню и картотека будет сохранена.

В программе реализована функция safe_scanf, которая контролирует вводимые пользователем данные. Если пользователь вводит строки в места, куда предполагались числа, например, safe_scanf предупреждает пользователя об этом и выводит ошибку.

Для добавления записей в таблицу была реализована функция add_new_card, для удаления — delete_card, для поиска и редактирования — search_card и edit_card — соответственно, для сортировки sort_alpha и sort_number, для вывода картотеки — print_list.

2.2. Описание структуры проекта

Точка входа в программу (функция main) находится в файле main.c, картотека находится в корне директории в файле cart.csv, в директории includesH находятся прототипы функции, созданных для работы с файлами, строками и списком. В директории includesC описаны сами функции, прототипы которых определены в includesH.

Имя модуля	Назначение
declaration	Содержит созданные структуры.
file	Содержит функции для работы с файлами и строками.
list	Содержит функции для работы со списком
menu	Содержит функции печати и взаимодействия с меню
secondary_table	Содержит функции для работы с второстепенной таблицей

2.3. Описание структур данных

Структура узла списка

Имя поля	Тип	Назначение
name	char*	Название автомобиля
company	char*	Название компании, производящей данный автомобиль
year	int	Год производства автомобиля
price	int	Цен автомобиляа
weight	float	Вес автомобиля
mileage	float	Пробег автомобиля
speed	int[1]	Максимальная и минимальная скорость автомобиля

Структура головы списка

Имя поля	Тип	Назначение
count	char*	Количество элементов в списке
first	carNode*	Указатель на первый элемент в списке
last	carNode*	Указатель на последний элемент в списке

Структура узла списка

Имя поля	Тип	Назначение
id	char*	Количество элементов в списке
car	car*	Указатель на данные элемента в списке
next	carNode*	Указатель на следующий узел в списке
prev	carNode*	Указатель на предыдущий узел в списке

Структура сложного запроса

Имя поля	**Тип**	**Назначение**
is_good	char	Информация о том, ялвяется ли автомобиль хорошим
name	char*	Название автомобиля
company	char*	Компания автомобиля
low_year	int	Нижняя граница поиска по полю year
max_year	int	Верхняя граница поиска по полю year
low_price	int	Нижняя граница поиска по полю price
max_price	int	Верхняя граница поиска по полю price
low_speed_max	int	Нижняя граница поиска по полю speed[1]
max_speed_max	int	Верхняя граница поиска по полю speed[1]
low_speed_min	int	Нижняя граница поиска по полю speed[0]
max_speed_min	int	Верхняя граница поиска по полю speed[0]

3. ОПИСАНИЕ ФУНКЦИЙ

3.1. Функции для работы с файлом

1. Функция read_file

Описание:

Функция считывает данные из файла в список с помощью функций add_frist и fill_from_file.

Прототип:

void read_file (FILE *fp , carHead *head)

Пример вызова:

read_file(fp, carHead);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	fp	FILE*	Указатель на читаемый файл
Локальная	tmp	carNode*	Вспомогательная переменная
Локальная	cur_node	carNode*	Указатель на последний считанный узел списка
Локальная	buf	char	Массив, состоящий из символов считанной строки
Локальная	flag	int	Флаг
Локальная	id	int	Id вставляемого элемента
Локальная	arr	Char **	Массив указателей на

			строки

Возвращаемое значение:

Нет

2. Функция split

Описание:

Функция разбивает считанную из файла строку на массив указателей на строки

Прототип:

char **split(char **text, char *string, char *sep)

Пример вызова:

arr = split(arr,id)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	text	char**	Указатель на массив указателей на строки
Параметр	string	char*	Указатель на считанную строку
Параметр	sep	char*	Хранение символа разделителя
Локальная	count	int	Количество слов в строке
Локальная	i	int	Переменная счетчик
Локальная	k	int	Переменная-счетчик
Локальная	flag	int	Флаг
Локальная	count_to_clear	int	Количество успешно считанных строк

Локальная	res	Char*	Указатель на сформированную строку

Возвращаемое значение:

Массив указателей на строки

3. Функция clear_array

Описание: Функция очищает память, выделенную под массив указателей на строки при считывании файла.

Прототип:

void clear_array(char **arr, int cout)

Пример вызова:

arr = split(arr,id)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	arr	char**	Указатель на массив указателей на строки
Параметр	count	int	Количество строк, которые нужно очитсить
Локальная	i	int	Переменная - счетчик

Возвращаемое значение:

Нет

4. Функция arr_to_node

Описание:

Функция получает указатель на массив подстрок arr и значение поля id текущего элемента, считанного из файла. В теле функции происходит 12

выделение памяти под новый узел списка res, а так же выделение памяти под его поле data с последующим его заполнением из массива подстрок arr. Так же в теле функции заново вызывается функция split для того, чтобы разбить подстроку с максимальной и минимальной скоростью (последний элемент массива arr) разбить на подстроки. Так же функция устанавливает значения указателей res→next и res→prev = NULL.

Прототип:

carNode *arr_to_node(char *arr, int id)

Пример вызова:

tmp = arr_to_node(arr,id)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	res	carNode*	Указатель на новый элемент в списке
Параметр	arr	char**	Указатель на массив подстрок
Параметр	id	int	Id текущего элемента

Возвращаемое значение:

Функция возвращает указатель на новый узел списка (res).

5. Функция save_file

Описание:

Функция выделяет сохраняет список в исходный файл.

Прототип:

void save_file(FILE *fp, carHead *head);

Пример вызова:

save_file(fp, head);

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная	tmp	carNode*	Временная переменная
Локальная	i	int	Переменная-флаг
Параметр	fp	FILE *	Указатель на сохраняемый файл
Параметр	head	carHead*	Указатель на голову списка

Возвращаемое значение:

Нет

6. Функция get_node

Описание:

Функция выделяет память под новый узел списка и выполняет считывание его информационных полей с клавиатуры и установку следующего и предыдущего элемента за считываемым в NULL.

Прототип:

carNode *get_node()

Пример вызова:

node = get_node()

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная	new_node	carNode*	Указатель на новый узел в списке

Возвращаемое значение:

Возвращает указатель на новый узел списка, значение информационных полей которого было введено с клавиатуры.

7. Функция fill_from_file

Описание:

Функция получает указатель на голову списка, указатель на текущий элемент списка и указатель на узел списка, который нужно вставить после текущего. В теле функции происходит установка значения указателя cur_node→ next = node и установка указателя node→ prev = cur_node (т. е. предыдущий элемент в считанном с клавиатуры узле указывает на считанный до него узел в списке). Так же происходит установка указателя на последний элемент в списке на текущий элемент и увеличение счетчика узлов в списке.

Прототип:

void fill_from_file(carHead *head, carNode *cur_node, carNode *node)

Пример вызова:

fill_from_file(head,cur_node,tmp)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	CarHead *	Указатель на голову списка
Параметр	cur_node	carNode*	Указатель на текущий узел списка
Параметр	node	carNode*	Указатель на новый узел списка

Возвращаемое значение:

Нет

8. Функция safe_scanf

Описание:

Функция организует проверку введенных значений пользователем

Прототип:

int safe_scanf()

Пример вызова:

choice = safe_scanf()

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная	choоce	int	Корректные данные введенные пользователем
Локальная	str	Char [100]	Буфер для считывания данных из терминала

Возвращаемое значение:

Корректные данные введенные пользователем

9. Функция safe_scanf_f

Описание:

Функция организует проверку введенных значений пользователем (для значений типа float)

Прототип:

int safe_scanf_f()

Пример вызова:

choice = safe_scanf_f()

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение

Локальная	choоce	float	Корректные данные введенные пользователем
Локальная	str	Char [100]	Буфер для считывания данных из терминала

Возвращаемое значение:

Корректные данные введенные пользователем

3.2. Функции ддя работы с меню

1. Функция menu

Описание:

Функция организует печать меню на экран и считывание действий пользователя

Прототип:

void menu(carHead *head, FILE *fp)

Пример вызова:

menu(head , fp)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная	choice	int	Хранение выбора пользователя
Локальная	menu_func	(*menu_func[9])	Массив указателей на функции для работы с меню
Параметр	head	CarHead*	Указатель на голову списка
Параметр	fp	File*	Указатель на считанный файл

Возвращаемое значение:

Нет.

2. Функция sub_menu_search

Описание:

Функция организует печать под-меню на экран и считывание действий пользователя

Прототип:

void sub_menu_search(carHead *head)

Пример вызова:

ыги_menu_search(head , fp)

Описание переменных:

Нет

Возвращаемое значение:

Нет.

3. Функция sub_menu_sort

Описание:

Функция организует печать под-меню сортировки на экран и считывание действий пользователя

Прототип:

void sub_menu_sort(carHead *head)

Пример вызова:

menu(head , fp)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная	choice	short	Хранение выбора пользователя

Параметр	head	CarHead*	Указатель на голову списка

Возвращаемое значение:

Нет.

4. Функция sub_menu_edit

Описание:

Функция организует печать под-меню редактирования карточек на экран и считывание действий пользователя

Прототип:

void sub_menu_edit(carHead *head)

Пример вызова:

sub_menu_edit(head , fp)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная	choice	short	Хранение выбора пользователя
Локальная	id	int	Хранение id карточки, которую нужно будет заменить
Параметр	head	CarHead*	Указатель на голову списка

Возвращаемое значение:

Нет.

5. Функция print_reference

Описание:

Функция печатает справку на экран пользователя

Прототип:

void print_reference

Пример вызова:

print_reference

Описание переменных:

Нет

Возвращаемое значение:

Нет.

6. Функция print_secondary_table

Описание:

Функция организует печать второй таблицы (информация о качестве автомобиля) на экран пользователя.

Прототип:

void print_secondary_table(carHead *head)

Пример вызова:

sub_menu_edit(head , fp)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Локальная	choоce	int	Хранение выбора пользователя
Локальная	i	int	Переменная — флаг
Локальная	is_good	char	Хранение информации о том, качественен ли автомобиль
Локальная	tmp	carNode*	Временная переменная

Параметр	head	carHead*	Указатель на голову списка

Возвращаемое значение:

Нет.

3.3. Функции ддя работы со списком

1. Функция add_first

Описание:

Функция организует добавление нового узла в начало списка.

Прототип:

void add_first(carHead *head, carNode *node)

Пример вызова:

add_first(head, cur_node)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	node	carNode*	Указатель на новый узел в списке

Возвращаемое значение:

Нет.

1. Функция add_first

Описание:

Функция организует добавление нового узла в начало списка.

Прототип:

void add_first(carHead *head, carNode *node)

Пример вызова:

add_first(head, cur_node)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	node	carNode*	Указатель на новый узел в списке

Возвращаемое значение:

Нет.

2. Функции print_list, print_cur_list

Описание:

Функции организуют вывод списка на экран пользователя. В случае вызова фугкции print_cur_list пользователю не предоставляется возможность вернуться в главное меню

Прототип:

void print_list(carHead *head)

void print_cur_list(carHead *head)

Пример вызова:

print_list(head)

print_cur_list(head)

Описание переменных:

Вид переменной	Имя	Тип	Назначение

	переменной		
Параметр	head	carHead*	Указатель на голову списка
Локальная	tmp	carNode*	Вспомогательная переменная для прохода по всему списку

Возвращаемое значение:

Нет.

3. Функция add_new_card

Описание:

Функция организует добавление нового узла в начало списка или на место, выбранное пользователем.

Прототип:

void add_new_card(carHead *head)

Пример вызова:

add_new_card(head)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Параметр	head	carHead*	Указатель на голову списка
Локальная	new_node	carNode*	Указатель на новый узел в списке
Локальная	uid	int	Хранение id элемента, после которого пользователь хочет вставить новую запись
Локальная	tmp	carNode*	Вспомогательная

			переменная для прохода по всему списку

Возвращаемое значение:

Нет.

4. Функция delete_card

Описание:

Функция организует удаление элемента из списка

Прототип:

void delete_card(carHead *head)

Пример вызова:

delete_card(head)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Локальная	number	int	Хранение номера элемента, который пользователь хочет удалить
Локальная	tmp	carNode*	Вспомогательная переменная для прохода по всему списку

Возвращаемое значение:

Нет.

4. Функция delete_card_data

Описание:

Функция организует очищение памяти, выделенной под поля удаляемого узла.

Прототип:

void delete_card_data(carNode *node)

Пример вызова:

delete_card(node)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	node	carNode*	Указатель на удаляемый узел

Возвращаемое значение:

Нет

5. Функции inc_id и dec_id

Описание:

Функция inc_id выполняет увеличение на 1 id элементов в списке, начиная с передаваемого элемента. Функция dec_id выполняет уменьшение на 1 id элементов в списке, начиная с передаваемого элемента.

Прототип:

void inc_id(carHead *head , carNode *cur_node)

void dec_id(carHead *head , carNode *cur_node)

Пример вызова:

dec_id(head,tmp)

inc_id(head,tmp)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	cur_node	CarNode*	Указатель на элемент в списке, с которого будет производиться уменьшение или увеличение id элементов в списке

Возвращаемое значение:

Нет

6. Функция print_node

Описание:

Функция организует печать отдельной записи на экран пользователя

Прототип:

void print_node(carNode *node)

Пример вызова:

print_node(node)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	node	CarNode*	Указатель на элемент в списке, который будет напечатан

Возвращаемое значение:

Нет

7. Функция print_best_cars

Описание:

Функция организует печать качественных автомобилей на экран пользователя.

Прототип:

void print_best_cars(carHead *head)

Пример вызова:

print_best_cars(head)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Локальная	tmp	CarNode*	Вспомогательная переменная
Локальная	i	int	Переменная - счетчик

Возвращаемое значение:

Нет

8. Функция search_card

Описание:

Функция организует поиск записей по одному или нескольким полям и вывод на экран найденных записей.

Прототип:

void search_card(carHead *head)

Пример вызова:

search_card(head)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Локальная	tmp	CarNode*	Вспомогательная переменная
Локальная	i	int	Переменная - счетчик
Локальная	count	int	Количество найденных записей
Локальная	user_query	query*	Указатель на пользовательский запрос
Локальная	str	char[]	Буфер для ввода значений строковых полей

Возвращаемое значение:

Нет

9. Функция compare_card_and_query

Описание:

Функция организует сравнение полей записи с полями введенными пользователем для поиска

Прототип:

int compare_car_adn_query(carNode *node, query *user_query)

Пример вызова:

search_card(head)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	node	carNode*	Указатель на сравниваемую запись
Локальная	Res,name.company,year,price,max_speed,min_speed	int	Начальные значения для поиска
Локальная	res	int	Переменная - флаг
Параметр	user_query	query*	Указатель на пользовательский запрос

Возвращаемое значение:

1 — запись подходит, 0 — запись не подходит

10. Функция edit_card

Описание:

Функция организует вывод редактируемой записи на экран и вызов функции edit_filed, которая редактирует поля найденной записи.

Прототип:

void edit_card(carHead *head, int id, int key)

Пример вызова:

edit_card(head, id, key)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	id	int	Id записи, выбранной пользователем для редактирования
Параметр	key	int	Номер поля записи, которое будет отредактировано
Локальная	tmp	carNode*	Временная переменная

Возвращаемое значение:

Нет

11. Функция edit_field

Описание:

Функция редактирует выбранные поля записи

Прототип:

void edit_field(carHead *head, int id, int key)

Пример вызова:

edit_field(head, id, key)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	id	int	Id записи, выбранной пользователем для редактирования
Параметр	key	int	Номер поля записи, которое будет отредактировано
Локальная	new_data	char[]	Переменная для новых данных строкового типа
Локальная	new_data_int	int	Переменная для новых данных типа int
Локальная	new_data_float	float	Переменная для новых данных типа float

Возвращаемое значение:

Нет

12. Функция sort_card

Описание:

Функция редактирует выбранные поля записи

Прототип:

void edit_field(carHead *head, int id, int key)

Пример вызова:

edit_field(head, id, key)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	key	int	Номер поля записи, по которому будет проводиться сортировка
Локальная	sort_func	Void (*sort_func[2])	Массив указателей на функции для сортировки

Возвращаемое значение:

Нет

13. Функции sort_number и sort_alpha

Описание:

Функции сортируют списки по числовым и строковым полям соответственно.

Прототип:

void sort_alpha(carHead *head, int key)

void sort_number(carHead *head, int key)

Пример вызова:

sort_alpha(head, key)

sort_number(head, key)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	key	int	Номер поля записи, по которому будет проводиться сортировка
Локальная	choice	short	Хранение выбора пользователя
Локальная	i	int	Переменная - счетчик
Локальная	j	int	Переменная - счетчик
Локальная	cur	carNode*	Указатель на текущий узел в списке
Локальная	indx	carNode*	Указатель на следующий узел в списке

Возвращаемое значение:

Нет

14. Функции get_data_int и get_data_str

Описание:

Функции получают значения полей целого и строкового типа соответственно.

Прототип:

void get_data_int(car *data, int key)

void get_data_str(car *data, int key)

Пример вызова:

price = get_data_int(node->data, key)

name = get_data_str(node->data, key)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	data	car*	Указатель на данные из записи
Параметр	key	int	Номер поля, из которого будут браться данные

Возвращаемое значение:

Значение полей в текущей записи.

15. Функция customize_search

Описание:

Функция позволяет задать параметры для поиска записей в таблице

Прототип:

query *customize_search(carHead *head)

Пример вызова:

query = customize_search(head)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Локальная	key	int	Номер поля, из которого будут браться данные
Локальная	param1	int	Хранение искомого параметра типа int
Локальная	param2	int	Хранение искомого параметра типа int
Локальная	param3	char	Хранение искомого параметра типа char
Локальная	choice	short	Хранение выбора пользователя
Локальная	user_query	query*	Указатель на запрос пользователя
Локальная	data	char[]	Хранение введенных пользователем строковых полей

Возвращаемое значение:

Составной пользовательский запрос (user_query)

16. Функция init_query

Описание:

Функция инициализации структуры пользовательского запроса.

Прототип:

query *init_query()

Пример вызова:

user_query = init_query()

Описание переменных:

Нет

Возвращаемое значение:

Возвращает инициализированную структуру пользовательского запроса.

15. Функция clear_query

Описание:

Функция очищает память, выделенную под пользовательский запрос.

Прототип:

void clear_query(query *query)

Пример вызова:

clear_query(user_query)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	query	query*	Указатель на структуру запроса, которую нужно очистить

Возвращаемое значение:

Нет

16. Функция print_query

Описание:

Функция выводит на экран параметры пользовательского запроса для поиска

Прототип:

void print_query(query *query)

Пример вызова:

print_query(cur_query)

Описание переменных:

Нет

Возвращаемое значение:

Нет

17. Функция check_query_values

Описание:

Функция проверяет пользовательские параметры для поиска на корректность

Прототип:

int check_query_values(int val1, int val2)

Пример вызова:

is_error = check_query_values(val1, val2)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	val1	int	Первый параметр для проверки
Параметр	val2	int	Второй параметр для проверки
Локальная	is_error	int	Переменная - флаг

Возвращаемое значение:

1 — ошибка есть, 0 — ошибки нет

18. Функция swap_datas

Описание:

Функция swap_datas производит непосредственный обмен полей двух записей

Прототип:

void swap_datas(car **data1, car **data2)

Пример вызова:

swap_datas(&node1→data, &node2 → data)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	data1	car**	Указатель на данные первой записи
Параметр	data2	car**	Указатель на данные второй записи

Возвращаемое значение:

Нет

19. Функция swap_cards

Описание:

Функция swap_cards проверяет возможность обмена двух записей местами, а так же получает номера записей, которые нужно обменять.

Прототип:

void swap_cards(carHead *head)

Пример вызова:

swap_cards(head)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Локальная	id_1	int	Хранение id первого элемента
Локальная	id_2	int	Хранение id второго элемента
Локальная	node_1	carNode*	Указатель на узел списка с id = id_1
Локальная	node_2	carNode*	Указатель на узел списка с id = id_2

Возвращаемое значение:

Нет

19. Функция find_node

Описание:

Функция находит в списке узел с переданным ей id

Прототип:

carNode *find_node(carHead *ehad, int id)

Пример вызова:

node = find_node(head, id)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Параметр	id	int	Хранение id искомого узла
Локальная	tmp	CarNode*	Временная переменная

Возвращаемое значение:

tmp — найденный узел

20. Функция init_head

Описание:

Функция выделяет память под голову списка и устанавливает значение поля count = 0 и значения полей first и last в NULL.

Прототип:

carHead *init_head()

Пример вызова:

ph = init_head(ph)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	ph	carHead*	Указатель на голову списка

Возвращаемое значение:

head — инициализированная голова списка

21. Функция clear_list

Описание:

Функция очищает память, выделенную под список и его голову.

Прототип:

void clear_list(carHead *head)

Пример вызова:

clear_list(head)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Параметр	head	carHead*	Указатель на голову списка
Локальная	tmp	CarNode *	Временная переменная
Локальная	tmp1	CarNode *	Временная переменная

Возвращаемое значение:

Нет

4. ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

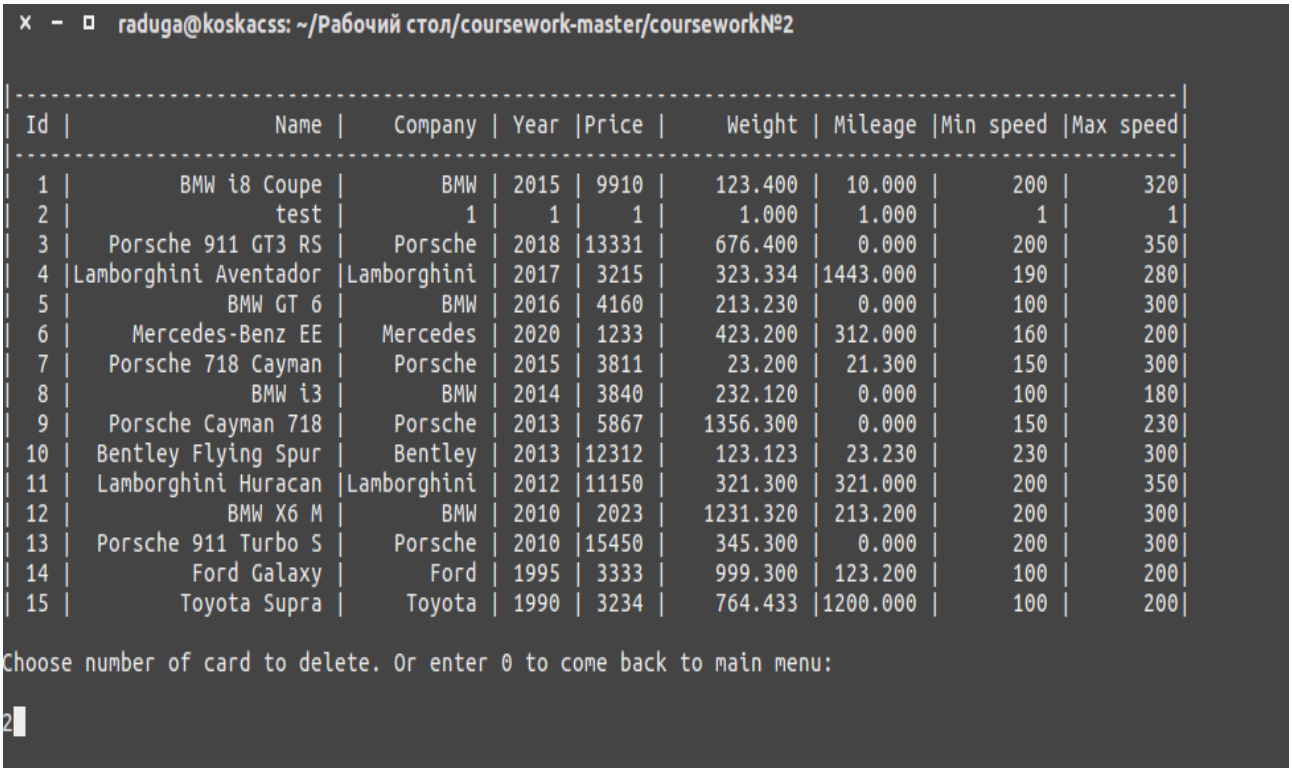
Добавление записи:

```
raduga@koskacss: ~/Рабочий стол/coursework-master/courseworkNo2
Fill information about new card:

Enter the name(Max length is 512 chars!):
test
Enter the company(Max length is 512 chars!):
1
Enter the year of production:
1
Enter the price:
1
Enter the weight:
1
Enter the mileage
1
Enter min and max speed
1
1
Enter the id after which you want to insert new card:
1
```

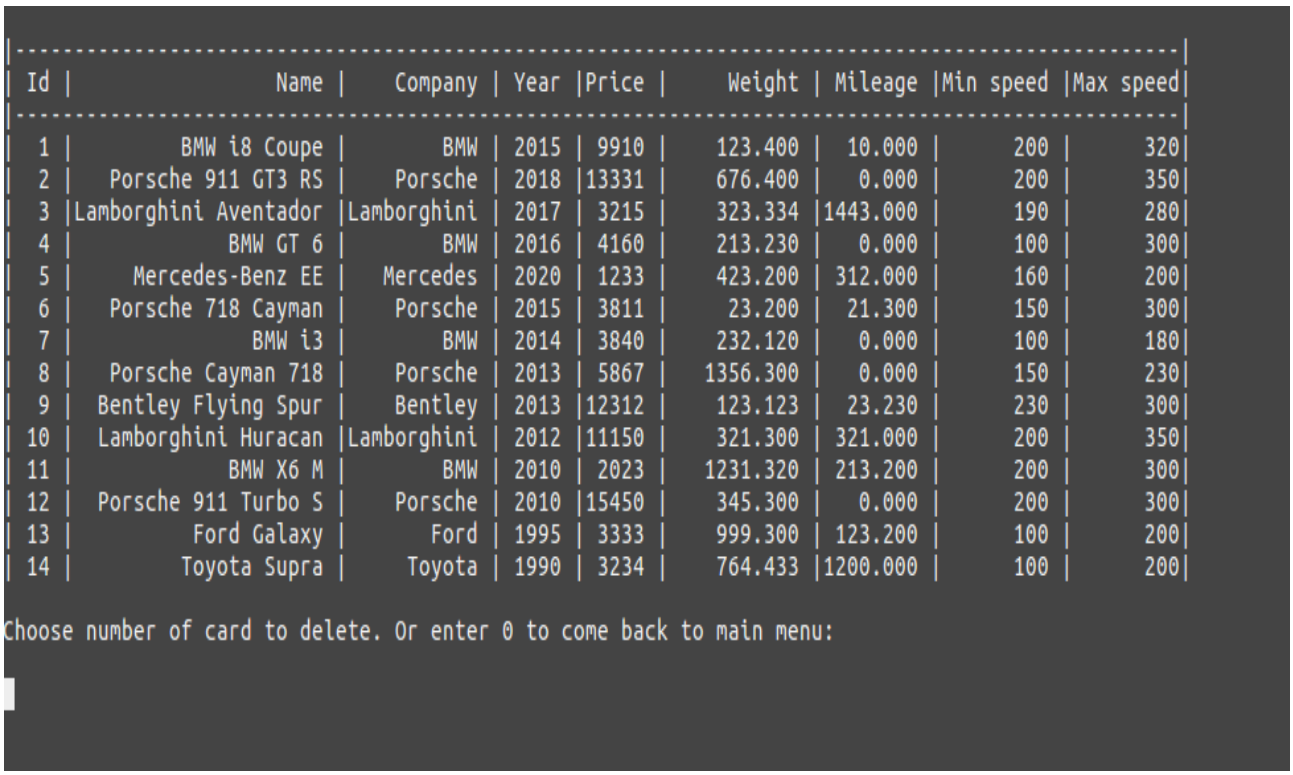
Картотека после добавленной записи:

```
raduga@koskacss: ~/Рабочий стол/coursework-master/courseworkNo2

|-----------------------------------------------------------------------------------------------|
| Id |                Name |    Company | Year |Price |     Weight | Mileage |Min speed |Max speed|
|-----------------------------------------------------------------------------------------------|
|  1 |         BMW i8 Coupe |        BMW | 2015 | 9910 |    123.400 |  10.000 |      200 |      320|
|  2 |                 test |          1 |    1 |    1 |      1.000 |   1.000 |        1 |        1|
|  3 |   Porsche 911 GT3 RS |    Porsche | 2018 |13331 |    676.400 |   0.000 |      200 |      350|
|  4 |Lamborghini Aventador |Lamborghini | 2017 | 3215 |    323.334 |1443.000 |      190 |      280|
|  5 |             BMW GT 6 |        BMW | 2016 | 4160 |    213.230 |   0.000 |      100 |      300|
|  6 |     Mercedes-Benz EE |   Mercedes | 2020 | 1233 |    423.200 | 312.000 |      160 |      200|
|  7 |   Porsche 718 Cayman |    Porsche | 2015 | 3811 |     23.200 |  21.300 |      150 |      300|
|  8 |               BMW i3 |        BMW | 2014 | 3840 |    232.120 |   0.000 |      100 |      180|
|  9 |   Porsche Cayman 718 |    Porsche | 2013 | 5867 |   1356.300 |   0.000 |      150 |      230|
| 10 |  Bentley Flying Spur |    Bentley | 2013 |12312 |    123.123 |  23.230 |      230 |      300|
| 11 |  Lamborghini Huracan |Lamborghini | 2012 |11150 |    321.300 | 321.000 |      200 |      350|
| 12 |             BMW X6 M |        BMW | 2010 | 2023 |   1231.320 | 213.200 |      200 |      300|
| 13 |  Porsche 911 Turbo S |    Porsche | 2010 |15450 |    345.300 |   0.000 |      200 |      300|
| 14 |         Ford Galaxy |       Ford | 1995 | 3333 |    999.300 | 123.200 |      100 |      200|
| 15 |        Toyota Supra |     Toyota | 1990 | 3234 |    764.433 |1200.000 |      100 |      200|

Press any key to come back to main menu
```

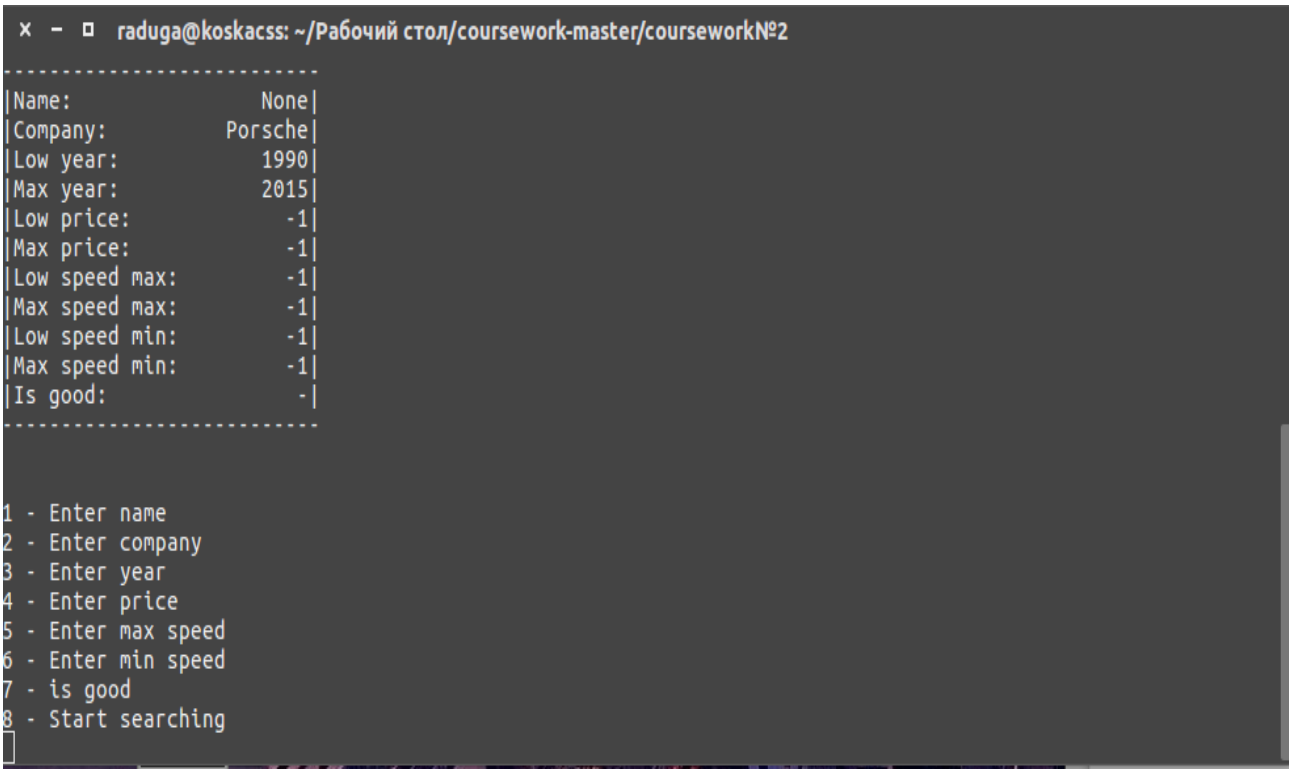
Удаление записи:



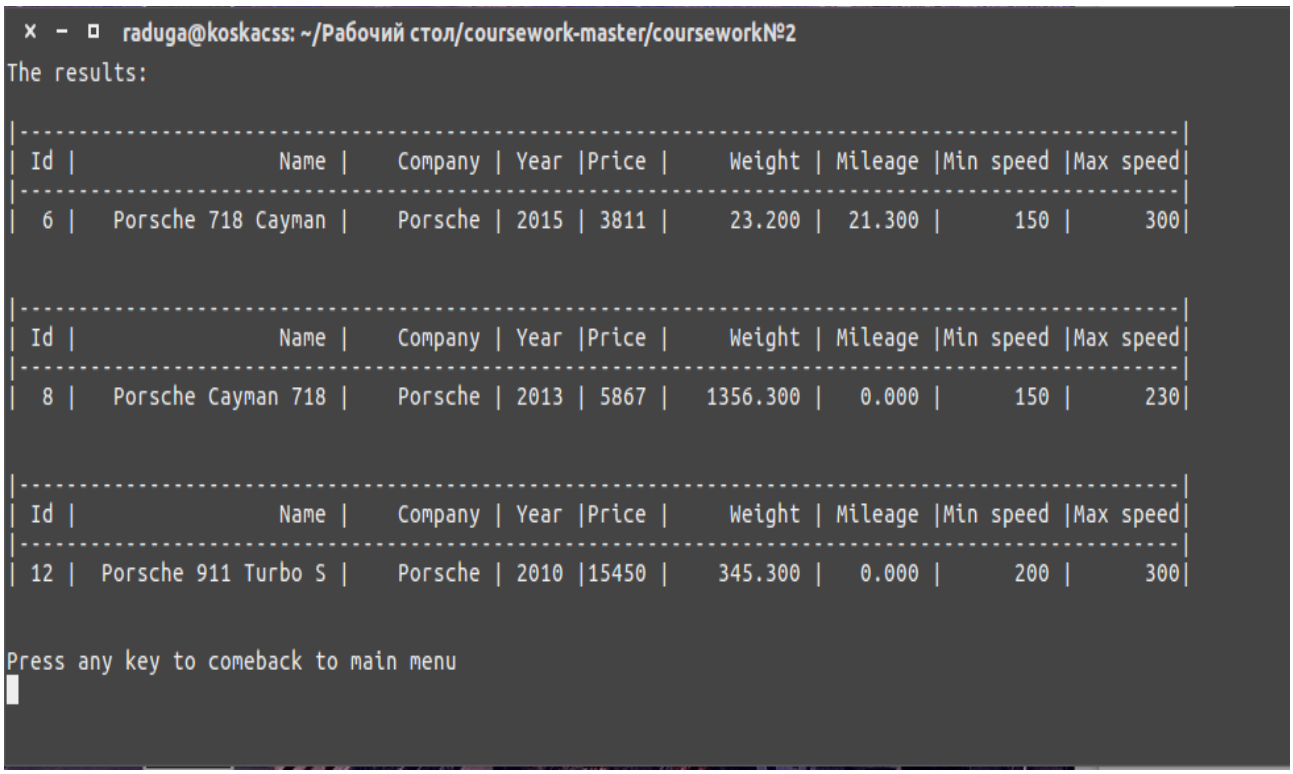
Картотека после добавленной записи:



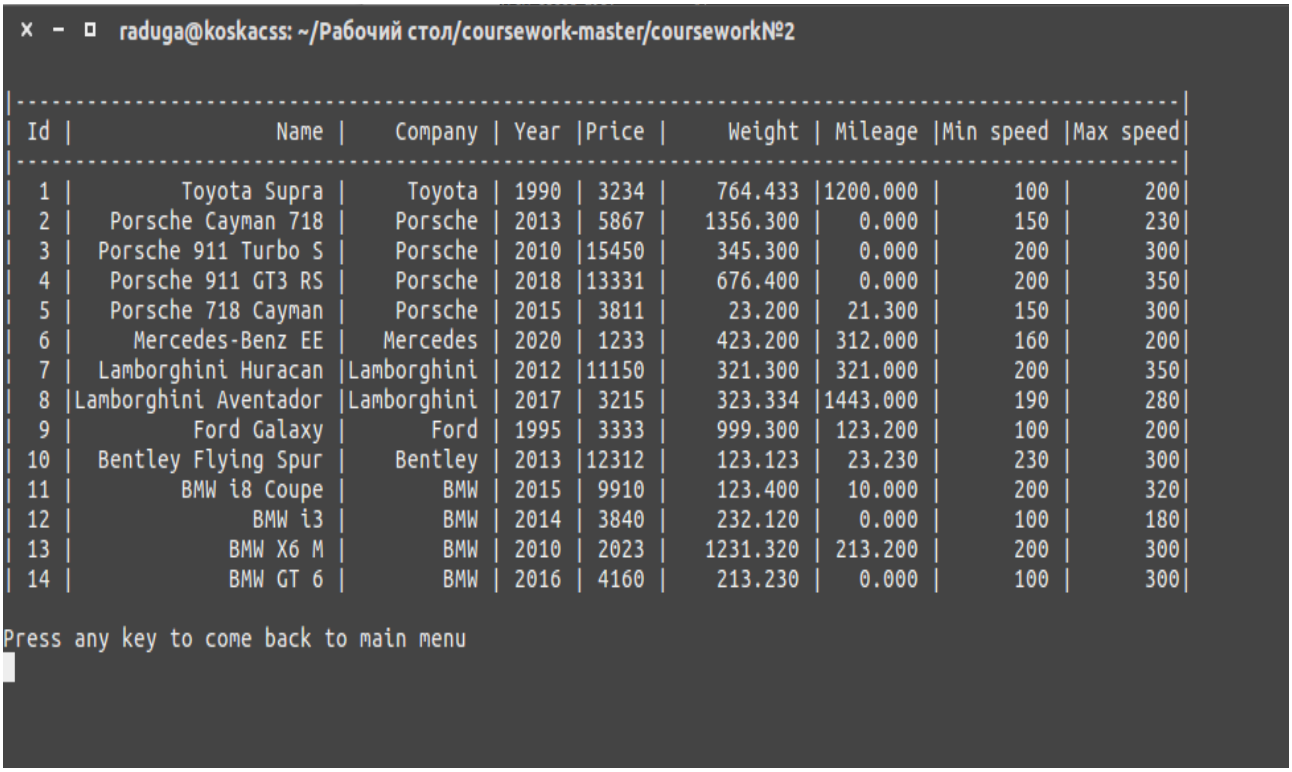
Поиск записей:



Найденные записи:



Отсортированная по убыванию первого поля картотека:



Редактирование записей:

```
raduga@koskacss: ~/Рабочий стол/coursework-master/courseworkNo2
1

Enter the field to edit:

1 - Name
2 - Company
3 - Year of production
4 - Price
5 - Weight
6 - Mileage
7 - Min speed
8 - Max speed
0 - Go back to main menu
1
You are going to update this card:


|-----------------------------------------------------------------------------------------------|
| Id |                Name |    Company | Year |Price |     Weight | Mileage |Min speed |Max speed|
|-----------------------------------------------------------------------------------------------|
|  1 |        Toyota Supra |     Toyota | 1990 | 3234 |    764.433 |1200.000 |      100 |      200|

Enter new name (Max length 21 symbols):
Test
```

Картотека после редактирования:

```
raduga@koskacss: ~/Рабочий стол/coursework-master/courseworkNo2

|-----------------------------------------------------------------------------------------------|
| Id |                Name |    Company | Year |Price |     Weight | Mileage |Min speed |Max speed|
|-----------------------------------------------------------------------------------------------|
|  1 |                Test |     Toyota | 1990 | 3234 |    764.433 |1200.000 |      100 |      200|
|  2 |   Porsche Cayman 718 |    Porsche | 2013 | 5867 |   1356.300 |   0.000 |      150 |      230|
|  3 |  Porsche 911 Turbo S |    Porsche | 2010 |15450 |    345.300 |   0.000 |      200 |      300|
|  4 |   Porsche 911 GT3 RS |    Porsche | 2018 |13331 |    676.400 |   0.000 |      200 |      350|
|  5 |   Porsche 718 Cayman |    Porsche | 2015 | 3811 |     23.200 |  21.300 |      150 |      300|
|  6 |     Mercedes-Benz EE |   Mercedes | 2020 | 1233 |    423.200 | 312.000 |      160 |      200|
|  7 |  Lamborghini Huracan |Lamborghini | 2012 |11150 |    321.300 | 321.000 |      200 |      350|
|  8 |Lamborghini Aventador |Lamborghini | 2017 | 3215 |    323.334 |1443.000 |      190 |      280|
|  9 |          Ford Galaxy |       Ford | 1995 | 3333 |    999.300 | 123.200 |      100 |      200|
| 10 |  Bentley Flying Spur |    Bentley | 2013 |12312 |    123.123 |  23.230 |      230 |      300|
| 11 |         BMW i8 Coupe |        BMW | 2015 | 9910 |    123.400 |  10.000 |      200 |      320|
| 12 |              BMW i3 |        BMW | 2014 | 3840 |    232.120 |   0.000 |      100 |      180|
| 13 |            BMW X6 M |        BMW | 2010 | 2023 |   1231.320 | 213.200 |      200 |      300|
| 14 |            BMW GT 6 |        BMW | 2016 | 4160 |    213.230 |   0.000 |      100 |      300|

Press any key to come back to main menu
```

Обмен двух записей местами:

```
x - □  raduga@koskacss: ~/Рабочий стол/coursework-master/courseworkNo2

|-------------------------------------------------------------------------------------------|
| Id |                Name |    Company | Year |Price |     Weight | Mileage |Min speed |Max speed|
|-------------------------------------------------------------------------------------------|
|  1 |                Test |     Toyota | 1990 | 3234 |    764.433 |1200.000 |      100 |      200|
|  2 |   Porsche Cayman 718 |    Porsche | 2013 | 5867 |   1356.300 |   0.000 |      150 |      230|
|  3 |  Porsche 911 Turbo S |    Porsche | 2010 |15450 |    345.300 |   0.000 |      200 |      300|
|  4 |   Porsche 911 GT3 RS |    Porsche | 2018 |13331 |    676.400 |   0.000 |      200 |      350|
|  5 |   Porsche 718 Cayman |    Porsche | 2015 | 3811 |     23.200 |  21.300 |      150 |      300|
|  6 |     Mercedes-Benz EE |   Mercedes | 2020 | 1233 |    423.200 | 312.000 |      160 |      200|
|  7 |  Lamborghini Huracan |Lamborghini | 2012 |11150 |    321.300 | 321.000 |      200 |      350|
|  8 |Lamborghini Aventador |Lamborghini | 2017 | 3215 |    323.334 |1443.000 |      190 |      280|
|  9 |          Ford Galaxy |       Ford | 1995 | 3333 |    999.300 | 123.200 |      100 |      200|
| 10 |  Bentley Flying Spur |    Bentley | 2013 |12312 |    123.123 |  23.230 |      230 |      300|
| 11 |         BMW i8 Coupe |        BMW | 2015 | 9910 |    123.400 |  10.000 |      200 |      320|
| 12 |              BMW i3 |        BMW | 2014 | 3840 |    232.120 |   0.000 |      100 |      180|
| 13 |            BMW X6 M |        BMW | 2010 | 2023 |   1231.320 | 213.200 |      200 |      300|
| 14 |            BMW GT 6 |        BMW | 2016 | 4160 |    213.230 |   0.000 |      100 |      300|

Enter the first id and then the second one
4 6
```

Картотека после обмена:

```
x - □  raduga@koskacss: ~/Рабочий стол/coursework-master/courseworkNo2

|-------------------------------------------------------------------------------------------|
| Id |                Name |    Company | Year |Price |     Weight | Mileage |Min speed |Max speed|
|-------------------------------------------------------------------------------------------|
|  1 |                Test |     Toyota | 1990 | 3234 |    764.433 |1200.000 |      100 |      200|
|  2 |   Porsche Cayman 718 |    Porsche | 2013 | 5867 |   1356.300 |   0.000 |      150 |      230|
|  3 |  Porsche 911 Turbo S |    Porsche | 2010 |15450 |    345.300 |   0.000 |      200 |      300|
|  4 |     Mercedes-Benz EE |   Mercedes | 2020 | 1233 |    423.200 | 312.000 |      160 |      200|
|  5 |   Porsche 718 Cayman |    Porsche | 2015 | 3811 |     23.200 |  21.300 |      150 |      300|
|  6 |   Porsche 911 GT3 RS |    Porsche | 2018 |13331 |    676.400 |   0.000 |      200 |      350|
|  7 |  Lamborghini Huracan |Lamborghini | 2012 |11150 |    321.300 | 321.000 |      200 |      350|
|  8 |Lamborghini Aventador |Lamborghini | 2017 | 3215 |    323.334 |1443.000 |      190 |      280|
|  9 |          Ford Galaxy |       Ford | 1995 | 3333 |    999.300 | 123.200 |      100 |      200|
| 10 |  Bentley Flying Spur |    Bentley | 2013 |12312 |    123.123 |  23.230 |      230 |      300|
| 11 |         BMW i8 Coupe |        BMW | 2015 | 9910 |    123.400 |  10.000 |      200 |      320|
| 12 |              BMW i3 |        BMW | 2014 | 3840 |    232.120 |   0.000 |      100 |      180|
| 13 |            BMW X6 M |        BMW | 2010 | 2023 |   1231.320 | 213.200 |      200 |      300|
| 14 |            BMW GT 6 |        BMW | 2016 | 4160 |    213.230 |   0.000 |      100 |      300|

Press any key to come back to main menu
```

ЗАКЛЮЧЕНИЕ

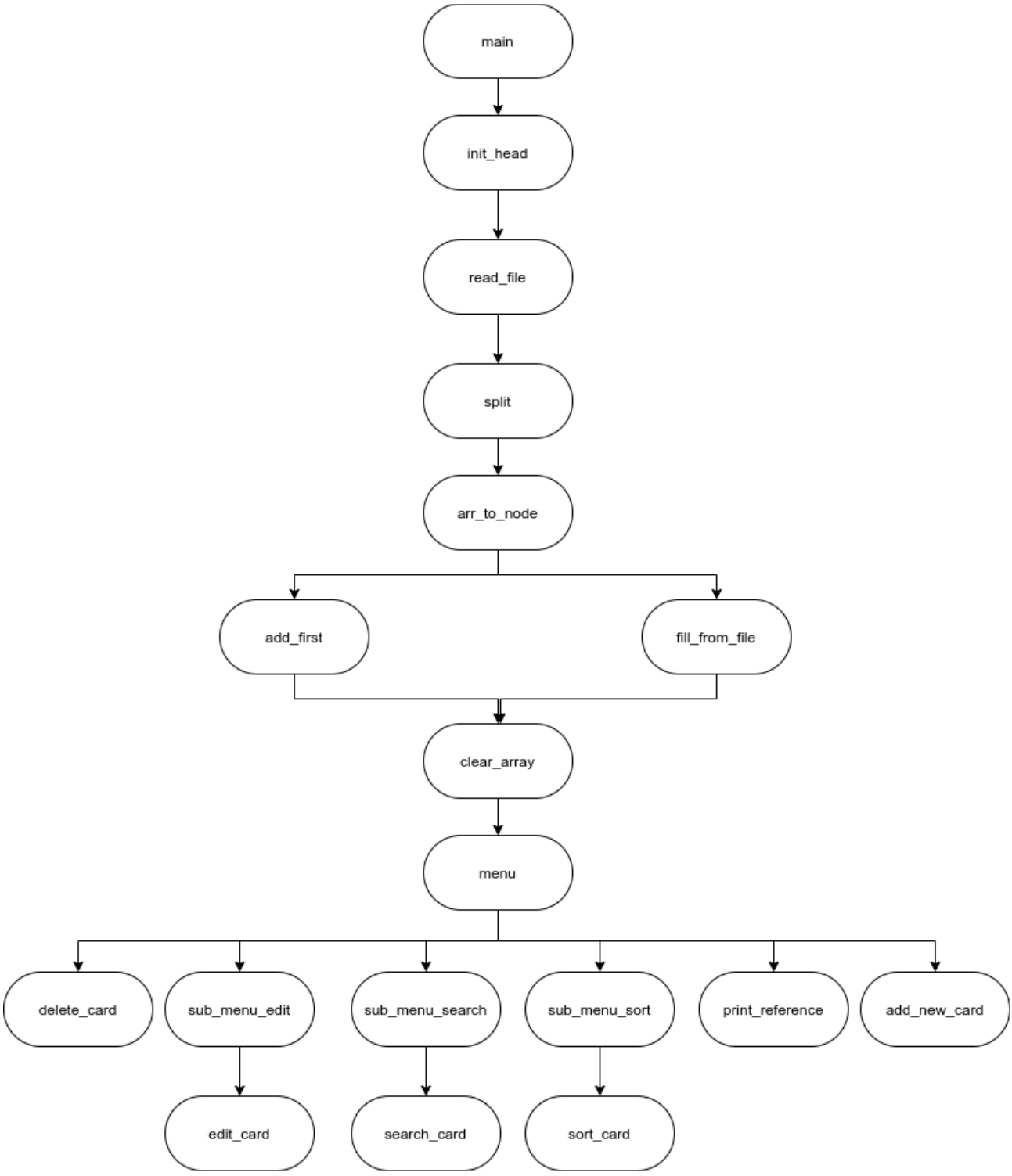
В ходе выполнения курсовой работы была разработана программа для хранения и обработки электронной картотеки, предметной областью которой являлись автомобили. Были реализованы функции поиска (в том числе с поддержкой сложных запросов), сортировки, печати, редактирования, добавления и удаления записей в электронной картотеке. В качестве динамической структуры данных был выбран двусвязный список. Программа была протестирована с помощью утилиты valgrind и никаких ошибок или утечек памяти обнаружено не было.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

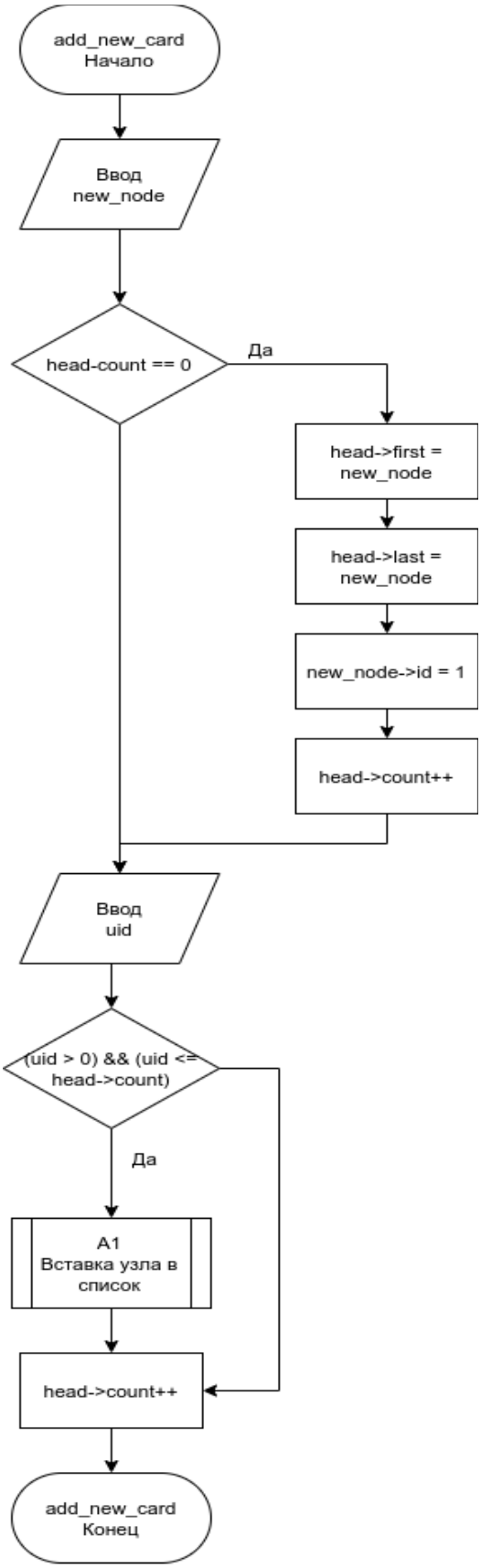
1. Примеры реализаций базовых функций для работы со связными списками // Prog-cpp // https://prog-cpp.ru/data-dls/ (дата обращения: 05.05.2020).

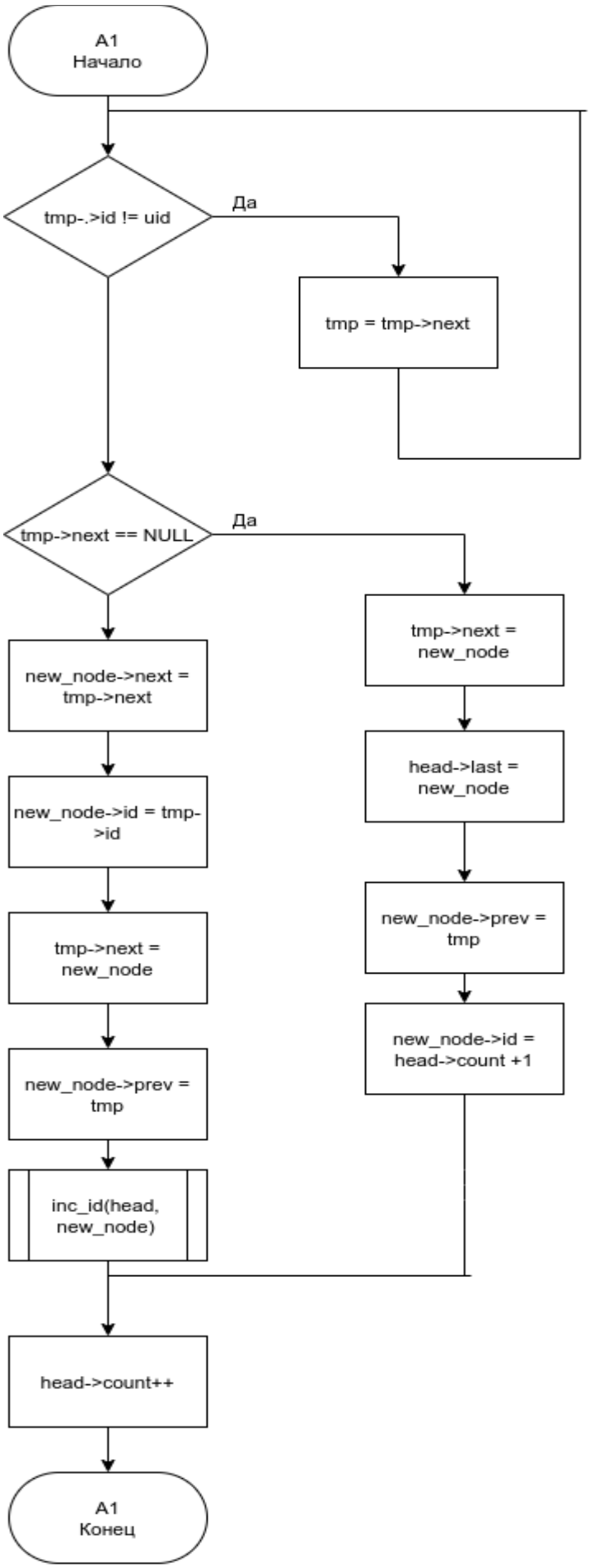
2. Описание связных списков // Wikipedia// https://ru.wikipedia.org/wiki (дата обращения: 01.05.2020).

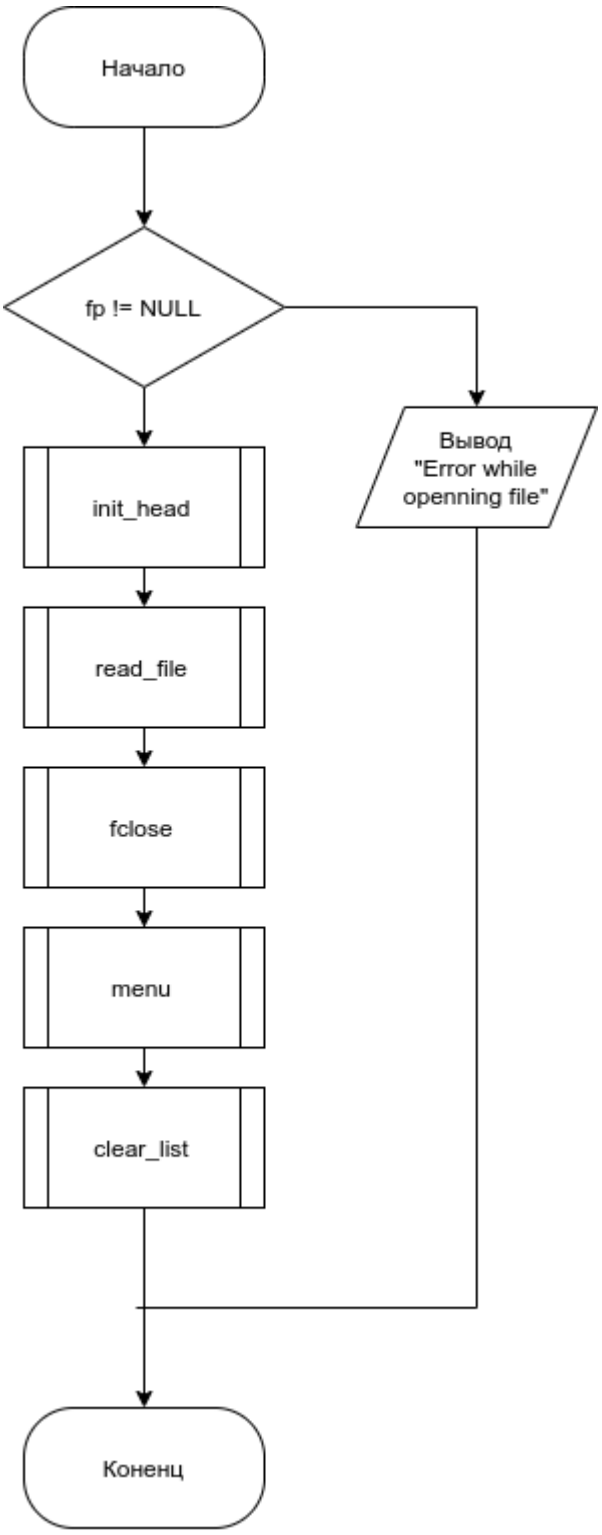
ПРИЛОЖЕНИЕ А
СХЕМА ВЫЗОВОВ ФУНКЦИЙ



ПРИЛОЖЕНИЕ Б
БЛОК-СХЕМЫ ФУНКЦИЙ




A1
Начало
tmp-.>id != uid
Да
tmp = tmp->next
tmp->next == NULL
Да
tmp->next = new_node
head->last = new_node
new_node->prev = tmp
new_node->id = head->count +1
new_node->next = tmp->next
new_node->id = tmp->id
tmp->next = new_node
new_node->prev = tmp
inc_id(head, new_node)
head->count++
A1
Конец


Начало
fp != NULL
Вывод
"Error while
openning file"
init_head
read_file
fclose
menu
clear_list
Коненц

ПРИЛОЖЕНИЕ В
ИСХОДНЫЙ КОД ПРОГРАММЫ

Ссылка на репозиторий GitHub: **https://github.com/koskacss/coursework**