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

КУРСОВАЯ РАБОТА

по дисциплине «Программирование»

Тема: «Программная реализация электронной картотеки»

Студент гр. 9305 _____ Николаенко К.Н.

Преподаватель _____ Перязева Ю.В.

Санкт-Петербург

2020

ЗАДАНИЕ

НА КУРСОВУЮ РАБОТУ

Студент Николаенко К.Н.

Группа 9305

Тема работы : программная реализация электронной картотеки.

Исходные данные:
csvфайл, содержащий электронную картотеку либо он пуст.

Содержание пояснительной записки:
«Введение», «Электронная картотека» , «Программная реализация», «Приложения», «Заключение», «Список использованных источников».

Предполагаемый объем пояснительной записки:
64 страницы.

Дата выдачи задания: 01.04.2020

Дата сдачи реферата: 27.05.2020

Дата защиты реферата: 30.05.2020

Студент _____ Николаенко К.Н.

Преподаватель _____ Перязева Ю.В.

АННОТАЦИЯ

В данной курсовой работе рассматривается электронная картотека с функционалом, а именно добавления, удаления, поиска, сортировки и редактирования. Основной задачей является написание программы, реализующей данную картотеку с возможностью считывания из файла и сохранения обработанной информации в него же. В содержание работы входят: описание картотеки, описание функций, примеры работы программы и блок-схемы, а также ссылка на гитхаб с самим кодом.

SUMMARY

In this course work, we consider an electronic card file with functionality, namely, adding, deleting, searching, sorting and editing. The main task is to write a program that implements this card file with the ability to read from the file and save the processed information in it. The content of the work includes: a description of the card library, a description of functions, examples of the program and flowcharts, as well as a link to the github with the code itself.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель работы

Написать программу, позволяющую работай с картотекой на определенную тематику.

Задачи

- Изучить поставленную задачу.
- Придумать пути её решения.
- отобразить алгоритм работы картотеки в виде блок-схемы.
- Написать программную реализацию картотеки.
- Проанализировать полученные результаты.

1. ЭЛЕКТРОННАЯ КАРТОТЕКА

1.1. Тематика и структуры

Электронная картотека основана на теме футбольных клубов.

Поля таблицы картотеки:
1)Название клуба
2)Страна клуба
3)Вероятность прохода в Лигу Чемпионов
4)Количество побед за последние 10 матчей
5)Количество ничьих за последние 10 матчей

Сумма побед и ничьих для одного клуба должна быть не больше 10.

Структура данных:

```
typedef struct fut
{
   char *name;
   char *country;
   float probability;
   float statistics[2];
} fut;
```

statistics[0] и statistics[1] соответственно количество побед и количество ничьих.

1.2. Функционал картотеки

Возможности картотеки:

1)Справка

2)Добавление карточки

3)Удаление

4)Поиск

5)Редактирование

6)Сортировка

7)Сохранение в файл

8)Выход

2. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1. Описание решения

В электронной картотеке в качестве хранения информации были использованы списки.

Добавление осуществляется либо вначало, либо в конец.

Удаление возможно по номеру карточки либо по полям картотеки с заданием какого-либо значения.

Вывод картотеки осуществленом двумя способами: с начала и с конца.

Сложный поиск работает по принципу нужно ли включать данное поле в поиск, если да, то вводится значение, иначе поле игнорируется.

Редактирование работает для одной карточки и определенно выбранного поля.

Сортировка возможна по любому из пяти полей картотеки.

Сохранение просто печатает текущий список в файл, если его нет то в файл печатается пробел.

Программа не позволяет одинаковым элементам появляется в картотеке.

2.2. Описание структур

Описание головы списка

Имя поля	**Тип**	**Назначение**
N	int	количество узлов
first	node	адрес на первый элемент списка
last	node	адрес на последний элемент списка

Описание узла

Имя поля	**Тип**	**Назначение**
id	int	Порядковый номер
baza	fut	элемент , содержащий структуру данных
next	node	адрес на следующий узел
prev	node	адрес не предыдущий узел

Описание структуры данных

Имя поля	**Тип**	**Назначение**
name	char	Название футбольного клуба
country	char	Название страны, в котором находится данный клуб
probability	float	вероятность выхода в Лигу Чемпионов
statistics	float	массив из 2 элементов(1 - количетсво побед, 2 - количетсво ничьих)

2.3. Описание функций меню

1. main
Описание:
Вызов функций считывания из файла.

Прототип:
int main()

Примеры вызова:
main()

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная переменная	q	head	голова списка

Возвращаемое значение: 0

2.menu

Описание:
меню выбора действий в картотеке. Основные функции вызываются отсюда.

Прототип:
void menu(head *q)

Примеры вызова:
menu(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	f	integer	выбор
Локальная	g	integer	выбор
Формальный аргумент	q	head	голова списка
Локальная	k	integer	для корректного ввода
Локальная	s	char	Информативная строка.
Локальная	d	integer	вспомогательная переменная для функции «delay»

Возвращаемое значение: отсутствует

3.delay

Описание:
Задерживает консоль перед очищением, чтобы пользователь видел результат своих действий, елси он есть.

Прототип:
void delay()

Примеры вызова:
delay()

Описание переменных: отсутствуют

Возвращаемое значение: отсутствует

4.title

Описание:
Печатает надпись SPORT над основным меню.

Прототип:
void title()

Примеры вызова:
title()

Описание переменных: отсутствуют

Возвращаемое значение: отсутствует

5.spravka

Описание:
Краткая справка о работе

Прототип:
void spravka()

Примеры вызова:
spravka()

Описание переменных: отсутствует

Возвращаемое значение: отсутствует

2.4. Описание функций списка

1.free_head

Описание:
Очищает голову списка.

Прототип:
void free_head(head *q)

Примеры вызова:
free_head(q)

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

2.free_node

Описание:
Очищение узла и структуры данных.

Прототип:
void free_node(node *temp)

Примеры вызова:
free_node(temp)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	temp	node	узел списка

Возвращаемое значение: отсутствует

3.free_list

Описание:
Очищает весь список используя функции «free_node» и «free_head».

Прототип:
void free_list(head *q)

Примеры вызова:
free_list(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	q	head	голова списка
Локальная	temp	node	узел списка

Возвращаемое значение: отсутствует

4.malloc_node

Описание:
Выделяет память для структуры данных внутри узла.

Прототип:
void malloc_node(node *temp)

Примеры вызова:
malloc_node(temp)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	temp	node	узел списка

Возвращаемое значение: отсутствует

5.create_head

Описание:
Создаёт голову списка.

Прототип:
head *create_head()

Примеры вызова:
create_head()

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	p	head	голова списка

Возвращаемое значение: p

6.create_node

Описание:
Создаёт узел.

Прототип:
node *create_node(head *q)

Примеры вызова:
create_node(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: temp

7.add_last

Описание:
Добавляет созданный элемент в конец списка.

Прототип:
void add_last(head *q)

Примеры вызова:
add_last(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

8.add_first

Описание:
Добавляет созданный элемент в начало списка.

Прототип:
node add_first(head *q)

Примеры вызова:
add_first(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

9.delete_first

Описание:
Удаление первого элемента списка.

Прототип:
void delete_first(head *q)

Примеры вызова:
delete_first(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

10.delete_node

Описание:
Удаление элемента списка.

Прототип:
void delete_node(node *temp, head *q)

Примеры вызова:
delete_node(temp, q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

11.copy_node

Описание:
Копирует информацию узла в другой.

Прототип:
node copy_node(node *p, node *temp)

Примеры вызова:
copy_node(p, temp)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	temp	node	узел списка
Формальный аргумент	p	node	узел списка

Возвращаемое значение: temp

12.copy_list

Описание:
Копирование всего списка.

Прототип:
void copy_list(head *q1, head *q)

Примеры вызова:
copy_list(q1, q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	temp	node	узел списка q
Формальный аргумент	q	head	голова списка
Локальная	temp1	node	узел списка q1
Формальный аргумент	q1	head	голова нового списка

Возвращаемое значение: отсутствует

13.transfer_node

Описание:
Перемещение одного узла перед другим.

Прототип:
void transfer_node(node *p, node *temp, head *q)

Примеры вызова:
transfer_node(p, temp, q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	p3	node	узел списка
Формальный аргумент	q	head	голова списка
Формальный аргумент	p	node	Узел списка, перед каким нужно вставить
Формальный аргумент	temp	node	Узел списка, который нужно переместить
Локальная	p1	node	узел списка, который переместят перез узлом «p»

Возвращаемое значение: отсутствует

2.5. Описание функций ввода и вывода

1.fill_list

Описание:
Считывание информации из файла и забивания его в массив структуры. Пока строка не совпадет с предыдущей.

Прототип:
void fill_list(head *q)

Примеры вызова:
fill_list(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	message	char	одна из строк файла
Локальная	str	char	одна из строк файла
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

2.split

Описание:
Функция разделяет полученную строку из файла и вбивает эти значения в новый узел.

Прототип:
void split(char *s, head *q)

Примеры вызова:
split(s, q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	mes	char	одна из строк файла
Локальная	s	char	одна выделенная часть строки mes
Формальный аргумент	q	head	голова списка
Локальная	temp	node	узел списка
Локальная	n	integer	разделяющая переменная

Возвращаемое значение: отсутствует

3.charToInt

Описание:
Переводит из char в integer.

Прототип:
int charToInt(char numeric)

Примеры вызова:
charToInt(s[i])

Описание переменных:

Вид переменной	Имя переменной	Тип	Назначение
Формальный аргумент	numeric	char	символ

Возвращаемое значение: numeric - 48

4.save_file

Описание:
Печатает в файл весь список.

Прототип:
void save_file(head *q)

Примеры вызова:
save_file(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	t	integer	количество узлов
Локальная	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

5. enterFromKeyboard

Описание:
Ввод с клавиатуры значения полей добавленного элемента.

Прототип:
void enterFromKeyboard (head *q)

Примеры вызова:
enterFromKeyboard (q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	k	integer	флаг для корректного ввода
Локальная	s1	char	информационная строка
Формальный аргумент	q	head	голова списка
Локальная	f	integer	введенное число
Локальная	temp	node	узел списка
Локальная	s	char	информационная строка

Возвращаемое значение: отсутствует

6. output_list_reverse

Описание:
Вывод списка с конца.

Прототип:
void output_list_reverse (head *q)

Примеры вызова:
output_list_reverse (q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	temp	node	узел списка
Локальная	i	integer	индекс узла
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

7. output_list

Описание:
Вывод списка с начала.

Прототип:
void output_list (head *q)

Примеры вызова:
output_list (q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	i	integer	индекс узла
Локальная	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

8. output_node

Описание:
Вывод узла.

Прототип:
void *output_node (node *temp, int i)

Примеры вызова:
output_node (temp, i)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	temp	node	узел списка
Формальный аргумент	i	integer	индекс узла

Возвращаемое значение: отсутствует

9.inform

Описание:
Информация о столбцах картотеки.

Прототип:
void inform()

Примеры вызова:
inform()

Описание переменных: отсутствуют

Возвращаемое значение: отсутствует

10.input

Описание:
Функция для корректного ввода чисел.

Прототип:
int input(char *s)

Примеры вызова:
input(s)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	i	integer	индекс
Локальная	u	integer	вспомогательная переменная некорректного ввода
Формальный аргумент	s1	char	информационная строка
Локальная	f	integer	число

Возвращаемое значение: f

2.6. Описание функций поиска

1. search_match

Описание:
функция поиска совпадений является связующей и вызывает остальные.

Прототип:
void search_match (head *q)

Примеры вызова:
search_match (q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s5	char	информационная строка
Локальная	s4	char	информационная строка
Формальный аргумент	q	head	голова списка
Локальная	q1	head	новый скопированный список
Локальная	s1	char	информационная строка
Локальная	s2	char	информационная строка
Локальная	s3	char	информационная строка

Возвращаемое значение: отсутствует

2. choose

Описание:
Выбор: «включать поле или же нет»

Прототип:
void choose (head *q1, char *s, int n)

Примеры вызова:
choose (q1, s, n)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	n	integer	определитель функции
Формальный аргумент	s	char	информационная строка
Формальный аргумент	q1	head	голова нового списка
Локальная	u	integer	вспомогательная переменная для корректного ввода
Локальная	f	integer	число

Возвращаемое значение: отсутствует

3. deleteCards

Описание:
Функция для выбора поля либо с помощью id удаления.

Прототип:
void deleteCards(head *q)

Примеры вызова:
deleteCards(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s	char	информационная строка
Локальная	g	integer	число
Формальный аргумент	q	head	голова списка
Локальная	k	integer	вспомогательная функция для корректного ввода

Возвращаемое значение: отсутствует

4. input_draws

Описание:
Функция ввода количество ничьих, она же является связующей.

Прототип:
void input_draws (head *q, int n)

Примеры вызова:
input_draws (q, n)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s	char	информационная строка
Формальный аргумент	n	integer	определитель функции
Формальный аргумент	q	head	голова списка
Локальная	l	integer	вспомогательная переменная для корректного ввода
Локальная	kod	integer	число

Возвращаемое значение: отсутствует

4. input_wins

Описание:
Функция ввода количество побед, она же является связующей.

Прототип:
void input_wins (head *q, int n)

Примеры вызова:
input_wins (q, n)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s	char	информационная строка
Формальный аргумент	n	integer	определитель функции
Формальный аргумент	q	head	голова списка
Локальная	l	integer	вспомогательная переменная для корректного ввода
Локальная	kod	integer	число

Возвращаемое значение: отсутствует

6. input_probability

Описание:
Функция ввода значение вероятности, она же является связующей.

Прототип:
void input_probability (head *q, int n)

Примеры вызова:
input_probability (q, n)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s	char	информационная строка
Формальный аргумент	n	integer	определитель функции
Формальный аргумент	q	head	голова списка
Локальная	l	integer	переменная для корректного ввода
Локальная	en	integer	верхняя граница вероятности в процентах
Локальная	beg	integer	нижняя граница вероятности в процентах
Локальная	e	float	верхняя граница вероятности
Локальная	b	float	нижняя граница вероятности
Локальная	f	integer	число

Возвращаемое значение: отсутствует

7. input_country

Описание:
функция ввода названия страны, она же является связующей.

Прототип:
void input_country (head *q, int n)

Примеры вызова:
input_country (q, n)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s	char	строка
Формальный аргумент	n	integer	определитель функции
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

8. input_name

Описание:
функция ввода название клуба, она же является связующей.

Прототип:
void input_name (head *q, int n)

Примеры вызова:
input_name (q, n)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s	char	строка
Формальный аргумент	n	integer	определитель функции
Формальный аргумент	q	head	голова списка

Возвращаемое значение: отсутствует

9. search_mega

Описание:
Общая функция для сложного поиска и удаления по полям.

Прототип:
void search_mega (head *q, char *s, int n, float b, float e, int f)

Примеры вызова:
search_mega (q, s, n, 1, 1, kod)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	n	integer	одна из строк файла
Формальный аргумент	s	char	одна из строк файла
Формальный аргумент	q	head	голова списка
Формальный аргумент	e	float	верхняя граница вероятности
Формальный аргумент	b	float	нижняя граница вероятности
Формальный аргумент	f	integer	количество побед либо ничьих
Локальная	num	integer	количество удаленных элементов
Локальная	p	node	предыдущий узел списка
Локальная	temp	node	узел списка

Возвращаемое значение: отсутствует

10. delete_matches

Описание:
функция удаляет нужный элемент, отделена для удобства чтения.

Прототип:
node *delete_matches (head *q, node *temp, node *p)

Примеры вызова:
delete_matches (q, temp, p)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Формальный аргумент	p	node	уредыдущий узел списка
Формальный аргумент	temp	node	узел списка
Формальный аргумент	q	head	голова списка

Возвращаемое значение: temp

11. delete_by_number

Описание:
Удаление карточек по номеру.

Прототип:
void delete_by_number (head *q)

Примеры вызова:
delete_by_number (q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	s	char	информационная строка
Локальная	l	integer	вспомогательная переменная
Формальный аргумент	q	head	голова списка
Локальная	f	integer	число
Локальная	i	integer	индекс
Локальная	temp	node	узел списка

Возвращаемое значение: отсутствует

2.7. Описание функций редактирования

1. edit_card

Описание:
Функция выбора карточки и редактирования.

Прототип:
void edit_card (head *q)

Примеры вызова:
edit_card (q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	temp	node	узел списка
Локальная	str	char	информационная строка
Локальная	s9	char	информационная строка
Формальный аргумент	q	head	голова списка
Локальная	f	integer	число
Локальная	g	integer	число
Локальная	k	integer	переменная для корректного ввода
Локальная	i	integer	индекс

Возвращаемое значение: отсутствует

2. edit_wins

Описание:
Функция для изменения поля количества побед, вынесена отдельно для удобства чтения.

Прототип:
void edit_wins(node *temp)

Примеры вызова:
edit_wins(temp)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	l	integer	переменная для корректного ввода
Локальная	s2	char	информационная строка
Формальный аргумент	temp	node	узел списка
Локальная	f	integer	число

Возвращаемое значение: отсутствует

3. edit_draws

Описание:
Функция для изменения поля количества побед, вынесена отдельно для удобства чтения.

Прототип:
void edit_draws(node *temp)

Примеры вызова:
edit_draws(temp)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	l	integer	переменная для корректного ввода
Локальная	s	char	информационная строка
Формальный аргумент	temp	node	узел списка
Локальная	f	integer	число

Возвращаемое значение: отсутствует

2.8. Описание функций сортировки

1.sort_cards

Описание:
Выбор поля сортировки.

Прототип:
void sort_cards(head *q)

Примеры вызова:
sort_cards(q)

Описание переменных:

Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	g	integer	число
Локальная	s	char	информационная строка
Формальный аргумент	q	head	голова списка
Локальная	k	integer	переменная для корректного ввода

Возвращаемое значение: отсутствует

2.sortMega

Описание:
Общая функция сортировки по каждому полю.

Прототип:
void sortMega (head *q, int n)

Примеры вызова:
sortMega (q, n)

Описание переменных:

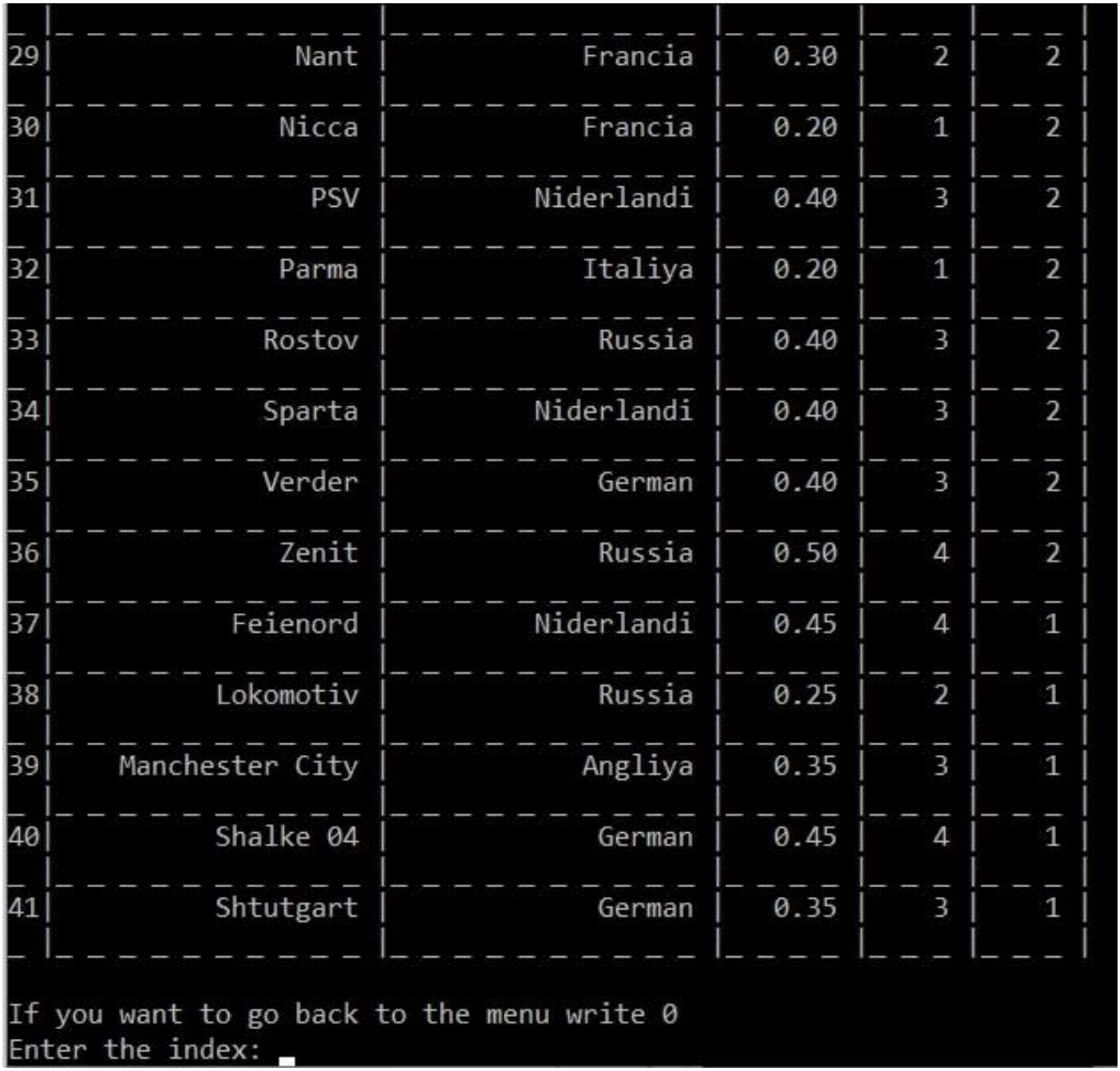
Вид переменной	**Имя переменной**	**Тип**	**Назначение**
Локальная	p1	node	Узел с которым сравнивается с temp
Формальный аргумент	n	integer	Определитель поля
Формальный аргумент	q	head	голова списка
Локальная	temp	node	Текущий узел
Локальная	k	integer	Флаг для выхода

Возвращаемое значение: отсутствует

2.9. Пример работы программы

Основное меню

Удаление по номеру



ЗАКЛЮЧЕНИЕ

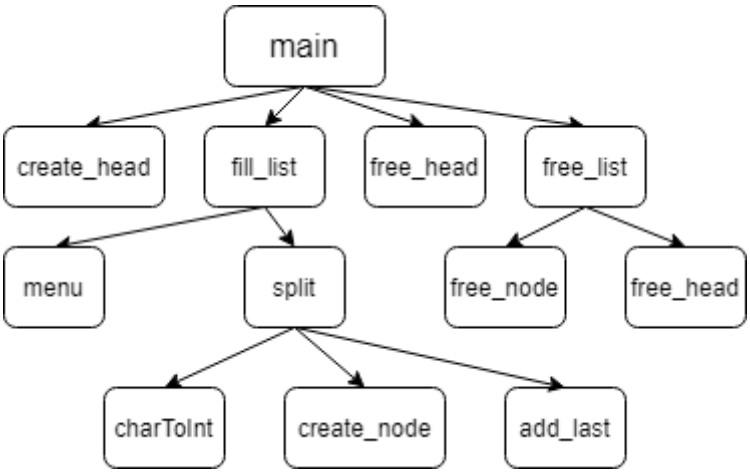
В ходе курсовой работы была реализована электронная картотека на языке программирования Си. Для её реализации были изучены: принципы работы с двусвязными линейными списками, программная разработка, реализация и отладка конечной программы. Для программной реализации потребовались полученные практические знания синтаксиса и правила написания кода на языке Си , в частностиф. Была проведена работа со списками , расфайловкой кода и указателями. По итогу я пришел к выводу, что данная работа была проделана не впустую, я смог узнать что-то новое, а также закрепить свои старые знания и создать что-то интересное.

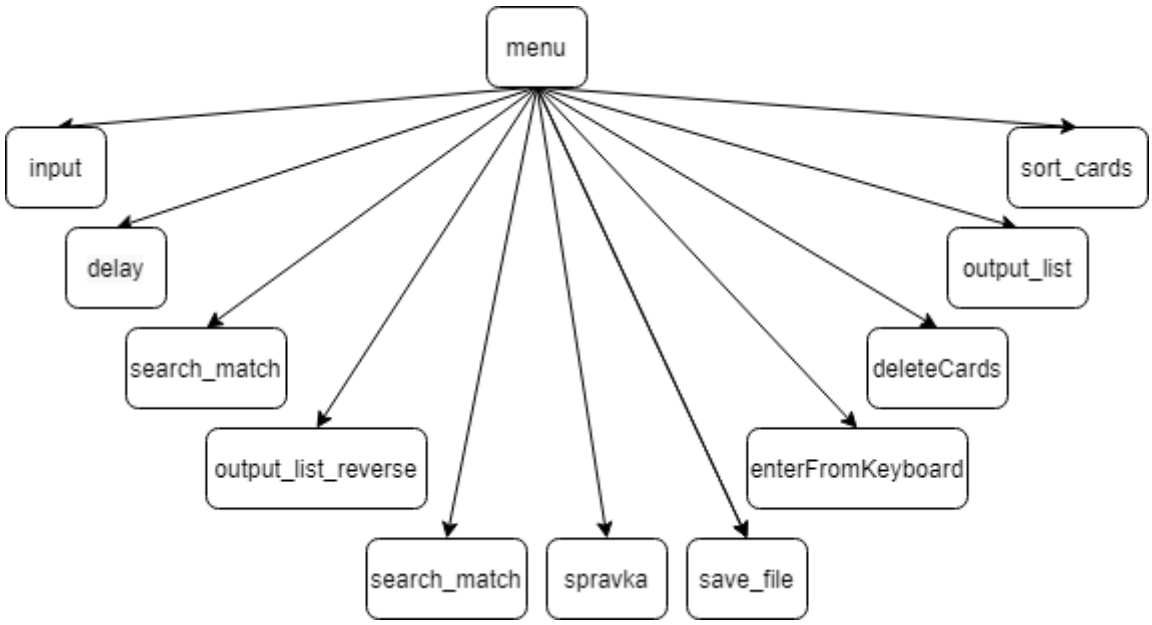
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

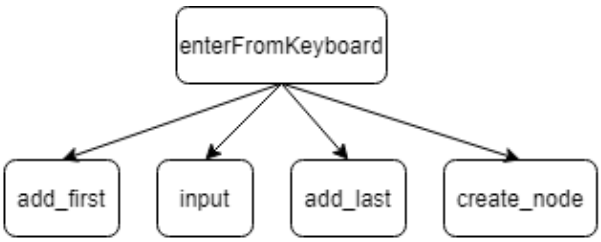
1) Двусвязный список в си
HTTPS://PROG-CPP.RU/DATA-DLS/

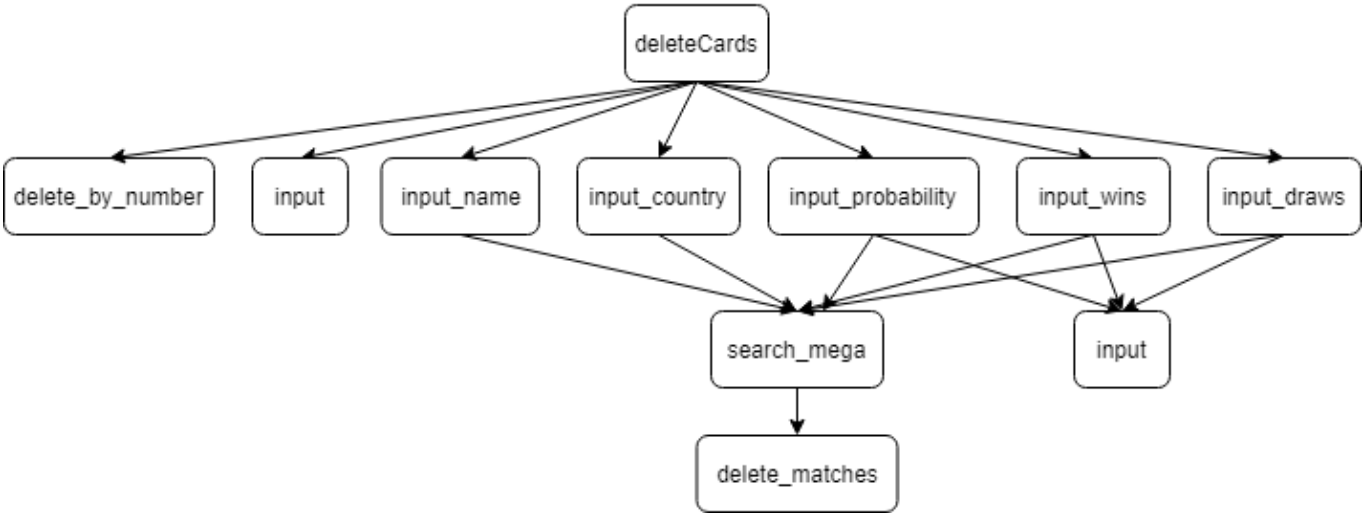
ПРИЛОЖЕНИЯ

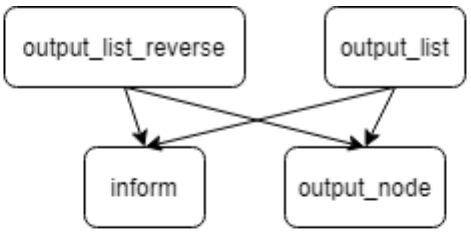
A. Схема вызова функций

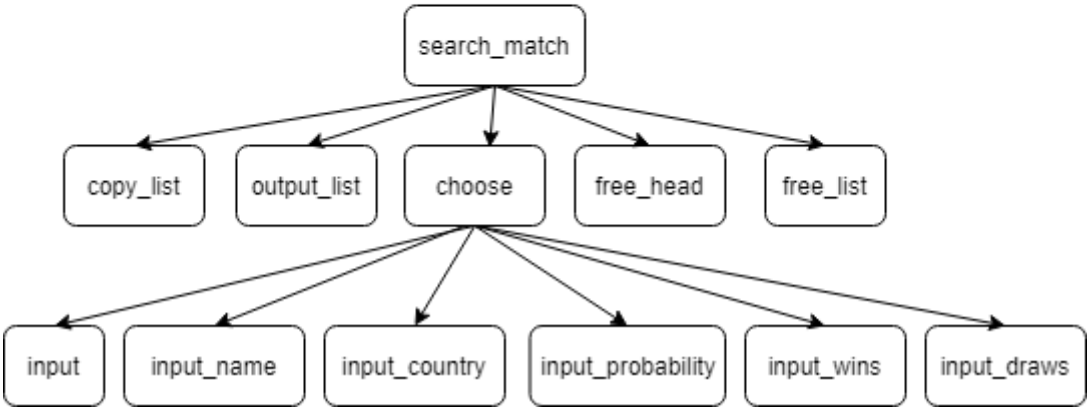


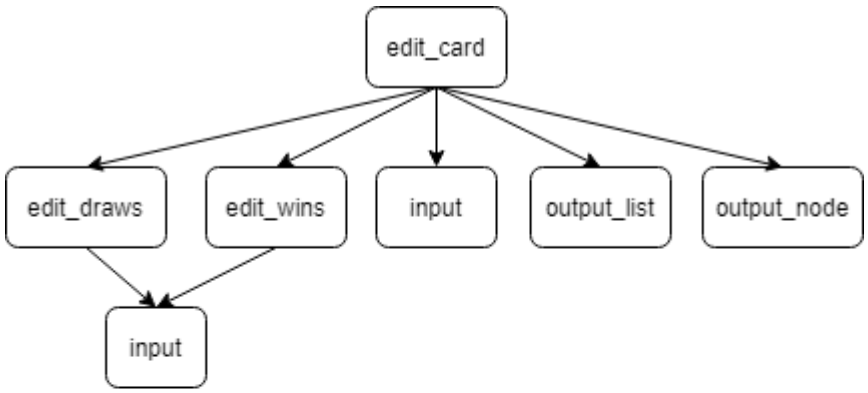

menu
input
delay
search_match
output_list_reverse
search_match
spravka
save_file
enterFromKeyboard
deleteCards
output_list
sort_cards

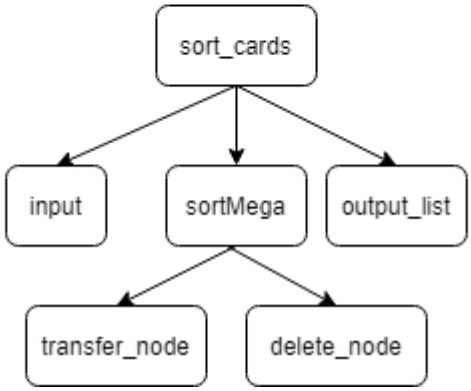

enterFromKeyboard
add_first
input
add_last
create_node


deleteCards
delete_by_number
input
input_name
input_country
input_probability
input_wins
input_draws
search_mega
input
delete_matches


output_list_reverse
output_list
inform
output_node

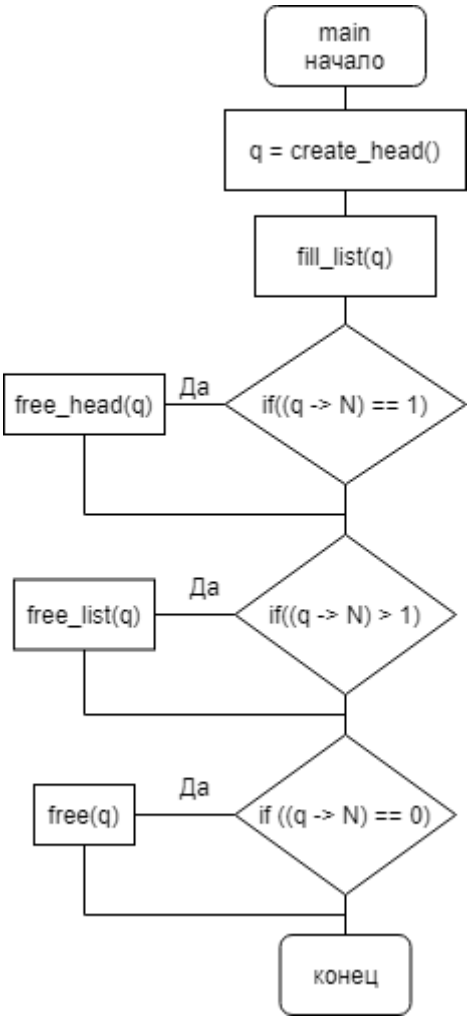

search_match
copy_list
output_list
choose
free_head
free_list
input
input_name
input_country
input_probability
input_wins
input_draws


edit_card
edit_draws
edit_wins
input
output_list
output_node
input

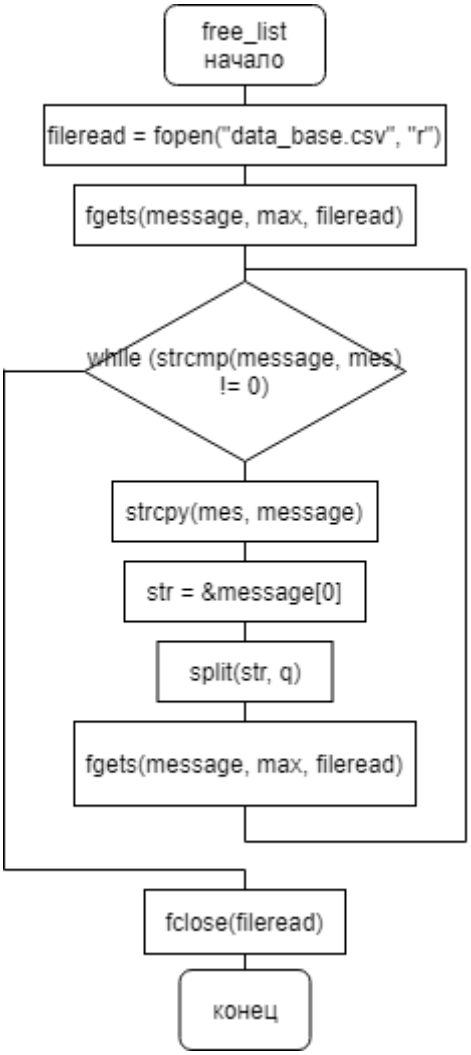

sort_cards
input
sortMega
output_list
transfer_node
delete_node

В. Схемы функций

main



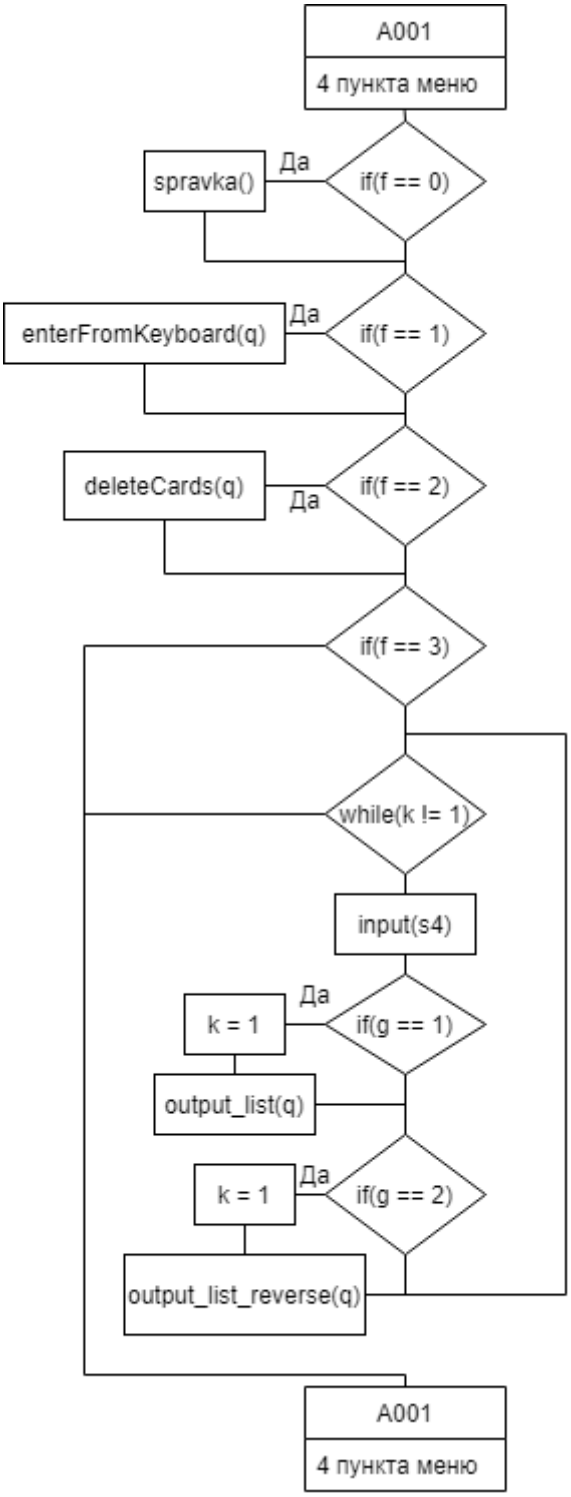
fill_list


free_list
начало
fileread = fopen("data_base.csv", "r")
fgets(message, max, fileread)
while (strcmp(message, mes)
!= 0)
strcpy(mes, message)
str = &message[0]
split(str, q)
fgets(message, max, fileread)
fclose(fileread)
конец

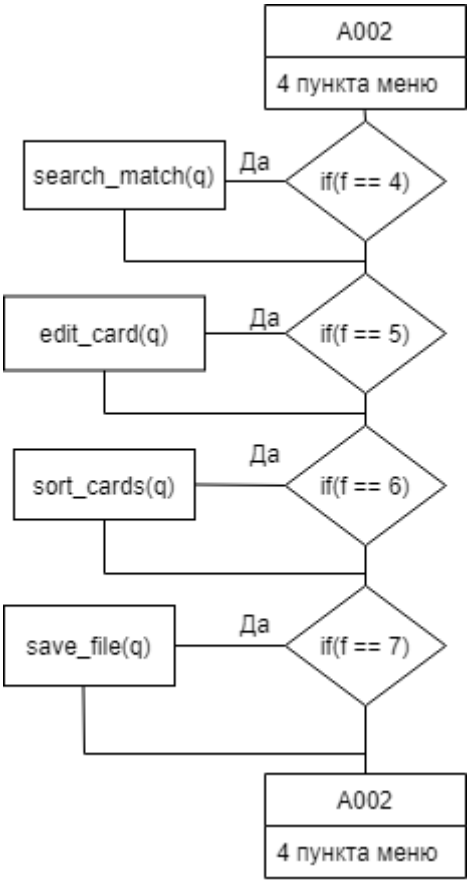
menu


menu
начало
f = 9
while(f != 8)
A001
4 пункта меню
A002
4 пункта меню
конец

A001


A001
4 пункта меню
if(f == 0)
Да
spravka()
if(f == 1)
Да
enterFromKeyboard(q)
if(f == 2)
Да
deleteCards(q)
if(f == 3)
while(k != 1)
input(s4)
if(g == 1)
Да
k = 1
output_list(q)
if(g == 2)
Да
k = 1
output_list_reverse(q)
A001
4 пункта меню

A002


A002
4 пункта меню
search_match(q)
Да
if(f == 4)
edit_card(q)
Да
if(f == 5)
sort_cards(q)
Да
if(f == 6)
save_file(q)
Да
if(f == 7)
A002
4 пункта меню

C. Текст программы

Ссылка на гитхаб:
https://github.com/NikolaenkoKonstantin/Programm_lab/tree/master/Kursovaya_2semestr