Министерство науки и образования РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)
Факультет компьютерных технологий и информатики

Кафедра вычислительной техники

Пояснительная записка к курсовой работе на тему: “Электронная картотека”

по дисциплине “Программирование. Дополнительные главы”

Выполнил: студент гр. 4306 Табаков А.В.
Принял: к.т.н., доцент Сискович Т.И.

Санкт-Петербург
2015 г.

Оглавление

Цель

Получить практические навыки работы с электронной картотекой на языке программирования «C\C++».

1. 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 признаку;
- сортировку;
- вывод результата на экран и сохранение на диске.

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

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 списка.

Программа должна иметь дружественный интерфейс и обеспечивать устойчивую работу при случайном нажатии на клавишу.

В картотеке хранятся сведения о гитарах.

2. Уточнение задания

В программе должно быть использовано простейшее меню. Выполнение программы должно быть многократным по желанию пользователя. В программе должны быть функции добавления элементов в список, редактирование элементов списка, удаление элементов списка, формирование нового списка по выбранной выборке, сортировка, вывод на экран и в файл элементов списка. Все данные вводятся с клавиатуры.

Пункты меню:

0: Справка
1: Добавление карточек о гитарах
2: Редактирование карточек
3: Удаление карточек
4: Вывод картотеки
5: Поиск карточек по параметру
6: Сортировка картотеки по параметру
7: Выход

3. Описание структур

Для решения задач разработаны структуры:

```
typedef struct stWood
  {
   char* Deck;                    //Дерево корпуса
   char* Neck;                    //Дерево грифа
  } WOOD;

typedef struct stGuitars
  {
   char* Name;                     //Название гитары
   int Strings;                    //Кол-во струн
   int Year;                       //Год производства
   WOOD Wood;                      //Дерево
  } GUITARS;
```

```
typedef struct stList
  {
   GUITARS Guitars;                    //Структура с информационными полями
   struct stList *next;                //Следующий элемент
   struct stList *prev;                //Предыдущий элемент
  } sLIST;

typedef sLIST* LIST;                   //Указатель на элемент списка
```

4. Контрольные примеры

Контрольные примеры представлены на рисунках 1-3.

Исходные данные						**Результат**				
Марка	**Год производства**	**Кол-во струн**	**Дерево**		**Критерии поиска**	**Марка**	**Год производства**	**Кол-во струн**	**Дерево**	
			корпус	**гриф**	**Year**				**корпус**	**гриф**
Gibson	1964	6	Ольха	Кедр	1990	Gibson	1964	6	Ольха	Кедр
Fender	1983	6	Сосна	Клён		Fender	1983	6	Сосна	Клён
Dean	1991	7	Липа	Клён						

Рис. 1. Контрольные примеры (поиск)

Исходные данные						**Результат**				
Марка	**Год производства**	**Кол-во струн**	**Дерево**		**Критерии сортировки**	**Марка**	**Год производства**	**Кол-во струн**	**Дерево**	
			корпус	**гриф**					**корпус**	**гриф**
Gibson	1964	6	Ольха	Кедр	По названию	Dean	1991	7	Липа	Клён
Fender	1983	6	Сосна	Клён		Fender	1983	6	Сосна	Клён
Dean	1991	7	Липа	Клён		Gibson	1964	6	Ольха	Кедр

Рис. 2. Контрольные примеры (сортировка)

Исходные данные						**Результат**				
Марка	**Год производства**	**Кол-во струн**	**Дерево**		**Критерии редакции**	**Марка**	**Год производства**	**Кол-во струн**	**Дерево**	
			корпус	**гриф**	**Дерево грифа**				**корпус**	**гриф**
Gibson	1964	6	Ольха	Кедр	Липа	Gibson	1964	6	Ольха	Липа
Fender	1983	6	Сосна	Клён		Fender	1983	6	Сосна	Клён
Dean	1991	7	Липа	Клён	Махагон	Dean	1991	7	Липа	Махагон

Рис. 3. Контрольные примеры (редактирование)

5. Описание главной функции

Назначение: начальная точка выполнения алгоритма.

Описание переменных функции

Описание переменных представлено в Таблице 1.

Таблица 1. Описание переменных главной функции

Имя переменной	Тип	Назначение
list	LIST	Указатель на первый элемент исходного списка
newList	LIST	Указатель на первый элемент сформированного списка
tempList	LIST*	Указатель на адрес списка
Q	int	Переменная выбора меню

Краткое описание алгоритма

Начало программы.
Шаг №1. Вывод меню.
Шаг №2. Выбор пользователем пункта меню.
Шаг №3. Переход к пункту, выбранным пользователем.
0: Справка. Переход к шагу 1
1: Добавление карточек о гитарах. Переход к шагу 1
2: Редактирование карточек. Переход к шагу 1
3: Удаление карточек. Переход к шагу 1
4: Вывод картотеки. Переход к шагу 1
5: Поиск карточек по параметру. Переход к шагу 1
6: Сортировка карточек по параметру. Переход к шагу 1
7: Выход. Переход к шагу 4
Шаг №4. Конец программы.

Схема алгоритма главной функции

Схема алгоритма главной функции представлена на рисунке 4.

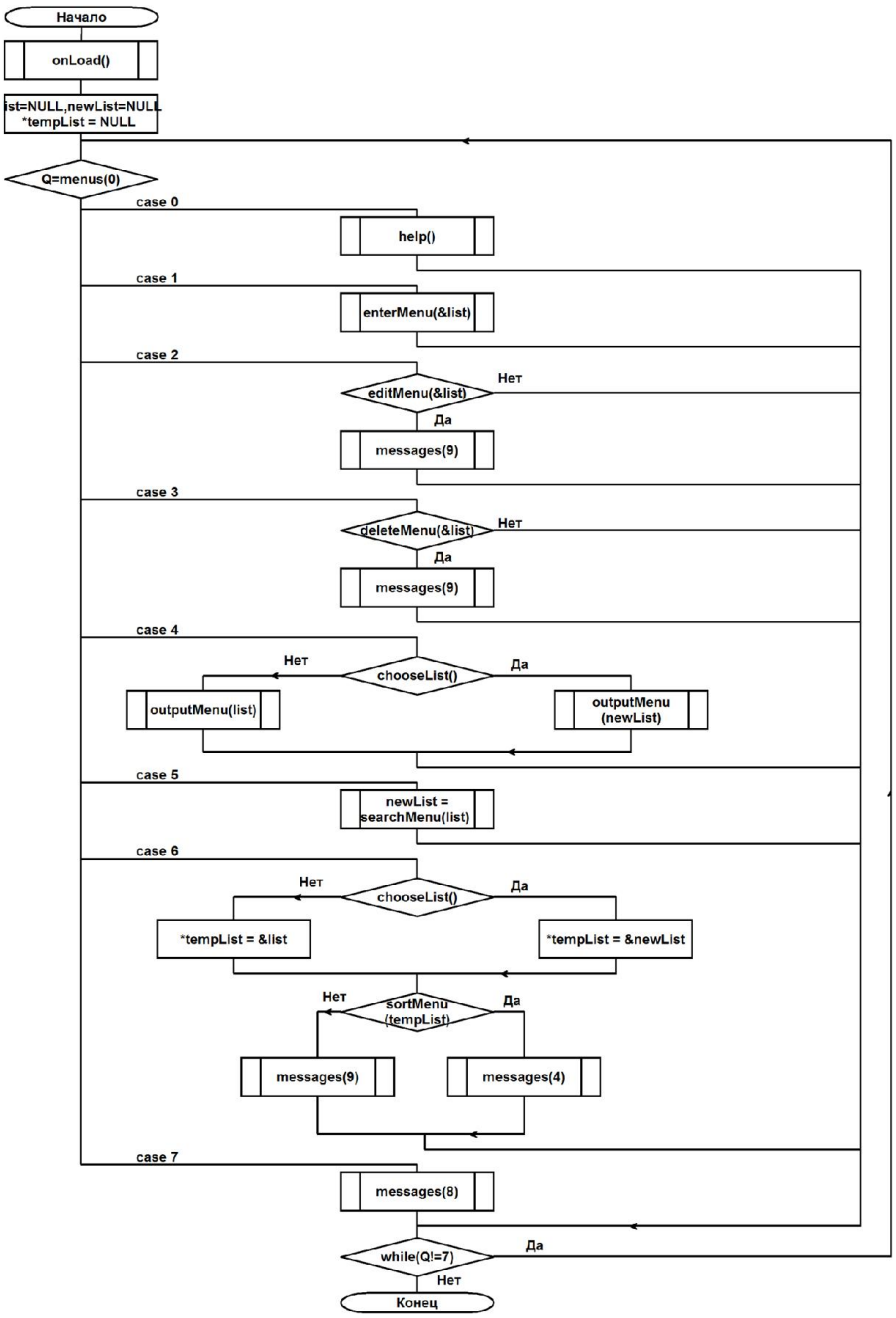

Рис. 4. Схема алгоритма главной функции

6. Описание функций

6.1 Описание функции onLoad

Назначение: вывод окна приветствия.
Прототип: int onLoad();
Пример вызова: onLoad();
Вызывающая функция: main.
Возвращаемое значение: 0 – выход из функции без ошибок.

6.2 Описание функции help

Назначение: вывод справки.
Прототип: int help();
Пример вызова: help();
Вызывающая функция: main.
Возвращаемое значение: 0 – выход из функции без ошибок.

6.3 Описание функции menus

Назначение: вывод разных меню программы.
Прототип: int menu(int Key); где –
Key – ключ для вывода разных меню.
Пример вызова: menu(0);
Вызывающая функция: main, enterMenu, edit, deleteMenu, sortMenu, searchMenu, outputMenu
Возвращаемое значение: 0 – выход из функции без ошибок.

6.4 Описание функции chooseList

Назначение: выбор списка.
Прототип: int chooseList();
Пример вызова: if(chooseList())
Вызывающая функция: main.
Вызываемая функция: enterNum.
Возвращаемое значение: ключ(0 – исходный список, 1 – сформированный список, 3 - бездействие)

6.5 Описание функции messages

Назначение: Функция используется для ввода сообщений пользователю.
Прототип: int messages(int Key); где –
Key – ключ для вывода разных сообщений.
Пример вызова: messages(1);
Вызывающая функция: main, enterMenu, deleteMenu , searchMenu, outputMenu, edit.
Возвращаемое значение: 0 – выход из функции без ошибок.
Сообщения:
messages(3): "Картотека удалена"
messages(4): "Сортировка успешно завершена "
"Для просмотра выберите пункт 'Вывод картотеки'"
messages(5): "Выборка из данных успешно сформирована"
"Для просмотра выберите пункт 'Вывод картотеки' -> 'Сформированной"
messages(6): "Выборка из данных не была сформирована"
"В исходных данных не нашлось таких результатов"
messages(8): "До новых встреч!"
messages(9): "Картотека пуста"
"Для выполнения этого действия, создайте картотеку, выбрав 1 пункт меню"
messages(10): "Элемент успешно удалён"
messages(11): "Элемент успешно добавлен"
"Для просмотра выберите пункт 'Вывод картотеки' -> 'Исходный список'"
messages(12): "Для того чтобы добавить элемент по позиции, необходимо наличие как минимум 2-х элементов"
messages(13): "Для того чтобы удалить элемент по позиции, необходимо наличие как минимум 2-х элементов"
messages(14): "Картотека успешно записана в файл"
messages(15): "Картотека успешно считана с файла и добавлена в исходную"
messages(17): "Файл с таким именем не существует"
"Измените имя файла!"

messages(18): "Картотека не записана. Возникли проблемы с файлом, обратитесь к
разработчику."
"Komdosh@gelezo2.ru"
"В теме укажите ‘Возникли проблемы с электронной картотекой’."
"В теле письма укажите, как появилась данная ошибка, и попытайтесь описать"
"алгоритм ваших действий."
messages(19): "Файл пустой"
messages(20): "Поле успешно изменено"
messages(777): "Возникла странная ошибка. Пожалуйста, напишите разработчику:"
"Komdosh@gelezo2.ru"
"В теме укажите ‘Возникли проблемы с электронной картотекой’."
"В теле письма укажите, как появилась данная ошибка, и попытайтесь описать"
"алгоритм ваших действий."

6.6 Описание функции enterMenu

Назначение: организация управления порядком вызова функций добавления элемента в список.
Прототип:int enterMenu(LIST* list); где –
*list – указатель на адрес первого элемента исходного списка.
Пример вызова: enterMenu(&list); где –
&list –адрес первого элемента исходного списка.
Вызывающая функция: main.
Вызываемая функция: menus, enterField, addNth, enterNum, messages, count.
Возвращаемое значение: 0 – выход из функции без ошибок.

Описание переменных

Описание переменных функции enterMenu представлено на рисунке 5.

Имя переменной	Тип	Назначение
Локальные переменные		
Q	int	Переменная выбора пункта меню
temp	LIST	Указатель на элемент списка для вставки
Формальные переменные		
list	LIST*	Указатель на адрес первого элемента исходного списка

Рис. 5. Описание переменных функции enterMenu

6.7 Описание функции addNth

Назначение: добавление n-го элемента.
Прототип: LIST addNth(LIST list, LIST temp, int n); где –
list – указатель на первый элемент исходного списка.
temp – указатель на элемент списка для вставки.
n – порядковый номер, куда вставить элемент.
Пример вызова: addNth(list, temp, 2); где –
list – указатель на первый элемент исходного списка.
temp – указатель на элемент списка для вставки.
2 – порядковый номер, куда вставить элемент.
Вызывающая функция: enterMenu, search, readFile.
Вызываемая функция: getElem.
Возвращаемое значение: указатель на первый элемент списка.

Описание переменных

Описание переменных функции addNth представлены на рисунке 6.

Имя переменной	Тип	Назначение
Формальные переменные		
list	LIST*	Указатель на адрес первого элемента исходного списка
temp	LIST	Указатель на элемента для вставки
n	int	Номер позиции

Рис. 6. Описание переменных функции addNth

6.8 Описание функции edit

Назначение: редактирование n-го элемента.
Прототип: int edit(LIST *list); где –
*list – указатель на адрес первого элемента исходного списка.
Пример вызова: edit (&list) где –
&list – адрес первого элемента исходного списка.
Вызывающая функция: main.
Вызываемая функция: getElem, enterNum, enterWord, messages.
Возвращаемое значение: 0 – выход из функции без ошибок.

Описание переменных

Описание переменных функции edit представлены на рисунке 7.

Имя переменной	Тип	Назначение
Q	int	Переменная выбора пункта меню
n	int	
Формальные переменные		
list	LIST*	Указатель на адрес первого элемента исходного списка

Рис. 7. Описание переменных функции edit

6.9 Описание функции enterField

Назначение: ввод информационных полей.
Прототип: LIST enterField();
Пример вызова: temp=enterField(); где –
temp - указатель на элемент списка.
Вызывающая функция: enterMenu.
Вызываемая функция: enterNum, enterWord.
Возвращаемое значение: указатель на элемент исходного списка.

Описание переменных

Описание переменных функции enterField представлено на рисунке 8.

Имя переменной	Тип	Назначение
Локальные переменные		
list	LIST	Элемент списка

Рис. 8. Описание переменных функции enterField

6.10 Описание функции enterWord

Назначение: ввод слова, длины 10.
Прототип: char* enterWord();
Пример вызова: tempstr=enterWord(); где –
tempstr - указатель на первый символ строки.
Вызывающая функция: enterField, fileName, edit.
Возвращаемое значение: указатель на первый символ строки.

Описание переменных

Описание переменных функции enterWord представлено на рисунке 9.

Имя переменной	Тип	Назначение
Локальные переменные		
temp	char*	Указатель на первый символ строки
str	char	Вспомогательная переменная, длины 10
i	unsigned int	Вспомогательная переменная для организации цикла

Рис. 9. Описание переменных функции enterField

6.11 Описание функции enterNum

Назначение: ввод чисел в заданном диапазоне.
Прототип: int enterNum(int first, int last); где –
first – начальное число.
last – конечное число.
Пример вызова: Q=enterNum(1, 7);
Вызывающая функция: menus, chooseList, enterMenu, edit, enterField, deleteMenu, searchMenu, fileName, outputMenu.
Возвращаемое значение: целое число.

Описание переменных

Описание переменных функции enterNum представлено на рисунке 10.

Имя переменной	Тип	Назначение
Локальные переменные		
num	int	Вспомогательная переменная
check_num	bool	Флаг является ли символ цифрой
check_all	bool	Флаг является ли строка числом
str	char*	Вспомогательная переменная
Формальные переменные		
first	int	Начальное число
last	int	Конечное число

Рис. 10. Описание переменных функции enterNum

6.12 Описание функции deleteMenu

Назначение: организация управления порядком вызова функций удаления элемента из списка.
Прототип:int deleteMenu(LIST *list); где –
*list – указатель на адрес первого элемента исходного списка.
Пример вызова: deleteMenu(&list); где –
&list – адрес первого элемента исходного списка.
Вызывающая функция: main.
Вызываемая функция: del, enterNum, messages, count.
Возвращаемое значение: 0 – выход из функции без ошибок.

Описание переменных

Описание переменных функции deleteMenu представлено на рисунке 11.

Имя переменной	Тип	Назначение
Локальные переменные		
Q	int	Переменная выбора пункта меню
Формальные переменные		
list	LIST*	Указатель на адрес первого элемента исходного списка

Рис. 11. Описание переменных функции deleteMenu

6.13 Описание функции del

Назначение: удаление n-го элемента.
Прототип: int del(LIST *list, int Key, int n); где –
*list – указатель на адрес первого элемента исходного списка.
Key – ключ какой элемент удалить (0 – начало, 1 – конец, 2 – n-ый, 3 – весь список).
n – номер элемента.
Пример вызова: if(del(&list, 2, 3)) где –
&list –адрес первого элемента исходного списка.
2 – ключ какой элемент удалить (2 – n-ый).
3 – номер элемента.
Вызывающая функция: deleteMenu, main.
Вызываемая функция: getElem.
Возвращаемое значение: 0 – выход из функции без ошибок, 1 – список пуст.

Описание переменных

Описание переменных функции del представлено на рисунке 12.

Имя переменной	Тип	Назначение
Локальные переменные		
temp	LIST	Указатель на элемент структуры
Формальные переменные		
list	LIST*	Указатель на адрес первого элемента списка
Key	Int	Ключ (0 – начало списка, 1 – конец списка,2 – n-ая позиция)
n	int	Номер элемента

Рис. 12. Описание переменных функции del

6.14 Описание функции sortMenu

Назначение: организация меню сортировки.
Прототип: int sortMenu(LIST list); где –
list – указатель на первый элемент списка.
Пример вызова: if(sortMenu(list)) где –
list – указатель на первый элемент списка.
Вызывающая функция: main.
Вызываемая функция: sort, menus.
Возвращаемое значение: 0 – выход из функции без ошибок, 1 – список пуст.

Описание переменных

Описание переменных функции sortMenu представлены на рисунке 13.

Имя переменной	Тип	Назначение
Локальные переменные		
Q	int	Переменная выбора пункта подменю
Формальные переменные		
list	LIST	Указатель на первый элемент исходного списка

Рис. 13. Описание переменных функции sortMenu

6.15 Описание функции sort

Назначение: сортировка списка.
Прототип:LIST sort(LIST list, int Key); где –
list – указатель на первый элемент списка.
Key – ключ выбора сортировки (1 – по названию, 2 – по количеству струн, 3 – по году производства, 4 – по дереву грифа, 5 – по дереву корпуса).
Пример вызова: list=sort(list, 1); где –
list – указатель на первый элемент списка.
1 - ключ выбора сортировки (1 – по названию)
Вызывающая функция: sortMenu.
Вызываемая функция: getElem.
Возвращаемое значение: указатель на первый элемент исходного списка

Описание переменных

Описание переменных функции sort представлено на рисунке 14.

Имя переменной	Тип	Назначение
Локальные переменные		
temp	LIST	Вспомогательная переменная
pFwd	LIST	Указатель на текущий элемент
pBwd	LIST	Указатель на предыдущий элемент
Sort	LIST	Указатель на первый элемент отсортированного списка
check	bool	Ключ сортировки (0 – производилась, 1 – не производилась)
Формальные переменные		
list	LIST	Указатель на первый элемент списка
Key	int	Ключ (1 – по году производства, 2 – по количеству струн)

Рис. 14. Описание переменных функции sort

6.16 Описание функции searchMenu

Назначение: организация меню обработки.
Прототип:LIST searchMenu (LIST list); где –
list – указатель на первый элемент списка.
Возвращаемое значение: указатель на первый элемент сформированного списка.
Пример вызова: New_list=searchMenu(list); где –
list – указатель на первый элемент исходного списка.
New_list – указатель на первый элемент сформированного списка.
Вызывающая функция: main.
Вызываемая функция: menus, search, enterNum, messages.

Описание переменных

Описание переменных функции searchMenu представлено на рисунке 15.

Имя переменной	Тип	Назначение
Локальные переменные		
New_list	LIST	Указатель на первый элемент сформированного списка
temp	int	Вспомогательная переменная
first	int	Нижняя граница числа
last	int	Верхняя граница числа
tempstr	char*	Указатель на первый символ строки
Q	int	Переменная выбора пункта подменю
Формальные переменные		
list	LIST	Указатель на первый элемент исходного списка

Рис. 15. Описание переменных функции searchMenu

6.17 Описание функции search

Назначение: поиск в списке.
Прототип: LIST search(LIST list , int Key, int num, char* str); где –
list – указатель на первый элемент исходного списка.
Key – ключ выбора сортировки (1 – по названию, 2 – по количеству струн, 3 – по году производства, 4 – по дереву грифа, 5 – по дереву корпуса).
num – число для сравнения.
str – строка для сравнения.
Пример вызова: New_List=search(list, 2, temp, tempstr); где –
list – указатель на первый элемент исходного списка.
2 – ключ выбора сортировки (2 – по количеству струн).
temp – число для сравнения.
tempstr – строка для сравнения.
New_list – указатель на первый элемент сформированного списка.
Вызывающая функция: searchMenu.
Вызываемая функция: addNth.
Возвращаемое значение: указатель на первый элемент сформированного списка.

Описание переменных

Описание переменных функции search представлено на рисунке 16.

Имя переменной	Тип	Назначение
Локальные переменные		
search	LIST	Указатель на первый элемент сформированного списка
temp	LIST	Вспомогательная переменная
Формальные переменные		
list	LIST	Указатель на первый элемент списка
str	char*	Указатель на первый символ строки
Key	Int	Ключ (1 – по году производства, 2 – по количеству струн)
num	Int	число для сравнения

Рис. 16. Описание переменных функции search

6.18 Описание функции fileName

Назначение: ввод имени файла.
Прототип: const char* fileName(int Key); где –
Key – ключ выбора подсказок (1 – для записи, 2 – для чтения)
Пример вызова: name=fileName(1); где –
1 – ключ выбора подсказок (1 – для записи)
Вызывающая функция: enterMenu, outputMenu.
Вызываемая функция: enterWord, enterNum.
Возвращаемое значение: имя файла.

Описание переменных

Описание переменных функции fileName представлены на рисунке 17.

Имя переменной	Тип	Назначение
Локальные переменные		
name	char	Указатель на первый символ вспомогательной переменной
Формальные переменные		
Key	int	Ключ(1 – для записи, 2 – для чтения)

Рис. 17. Описание переменных функции fileName

6.19 Описание функции recordFile

Назначение: запись в файл.
Прототип: int recordFile(LIST list, const char* name); где –
list – указатель на первый элемент списка.
name – имя файла.
Пример вызова: recordFile(list, name); где –
list – указатель на первый элемент списка.
name – имя файла.
Вызывающая функция: outputMenu.
Возвращаемое значение: 0 – выход из функции без ошибок, 1 – список пуст.

Описание переменных

Описание переменных функции recordFile представлено на рисунке 18.

Имя переменной	Тип	Назначение
Локальные переменные		
file	FILE*	Указатель на открытый файл
Формальные переменные		
List	LIST	Указатель на первый элемент
name	const char*	Указатель на первый символ строки имени файла

Рис. 18. Описание переменных функции recordFile

6.20 Описание функции readFile

Назначение: чтение из файла.
Прототип: LIST readFile(const char* name); где –
name – имя файла.
Пример вызова: list=readFile(name); где –
name – имя файла.
Вызывающая функция: enterMenu.
Вызываемая функция: addNth.
Возвращаемое значение: Указатель на первый элемент списка.

Описание переменных

Описание переменных функции readFile представлено на рисунке 19.

Имя переменной	Тип	Назначение
Локальные переменные		
name	char	Строка с именем
file	FILE*	Указатель на открытый файл
Формальные переменные		
name	const char*	Указатель на первый символ строки имени файла

Рис. 19. Описание переменных функции readFile

6.21 Описание функции outputMenu

Назначение: подменю вывода списка.
Прототип: int outputMenu(LIST list); где –
list – указатель на первый элемент списка.
Пример вызова: outputList (list); где –
list – указатель на первый элемент списка.
Вызывающая функция: main.
Вызываемая функция: messages, enterNum, recordFile, outputList.
Возвращаемое значение: 0 – выход из функции без ошибок.

6.22 Описание функции outputList

Назначение: вывод списка.
Прототип: int outputList(LIST list, int Key); где –
list – указатель на первый элемент исходного списка.
Key – ключ выбора вывода (0 – с начала, 1 – с конца).

Пример вызова: if(outputList(list, 0)) где –
list – указатель на первый элемент исходного списка.
0 – ключ выбора сортировки (0 – с начала).
Вызывающая функция: outputMenu, edit.
Вызываемая функция: messages, getElem.
Возвращаемое значение: 0 – выход из функции без ошибок, 1 – список пуст.

Описание переменных

Описание переменных функции outputList представлено на рисунке 20.

Имя переменной	Тип	Назначение
Формальные переменные		
list	LIST	Указатель на первый элемент исходного списка
Key	int	Переменная начала списка (0-с начала, 1-с конца)

Рис. 20. Описание переменных функции outputList

6.23 Описание функции getElem

Назначение: вернуть n-ый элемент.
Прототип:LIST getElem(LIST list, int Key, int n); где –
list – указатель на первый элемент списка.
Key – ключ какой элемент искать (0 – начало, 1 – конец, 2 – n-ый).
n – номер элемента.
Пример вызова: if(getElem(list, 0, 5)); где –
list – указатель на первый элемент списка.
Key – ключ какой элемент искать (0 – начало, 1 – конец, 2 – n-ый).
n – номер элемента.
Вызывающая функция: outputList, addNth, sort, del, edit
Возвращаемое значение: n-ый элемент списка.

Описание переменных

Описание переменных функции getElem представлено на рисунке 21.

Имя переменной	Тип	Назначение
Формальные переменные		
list	LIST	Указатель на первый элемент списка
Key	int	ключ (0 – начало, 1 – конец, 2 – n-ый)
n	int	номер элемента

Рис. 21. Описание переменных функции getElem

6.24 Описание функции count

Назначение: подсчёт количества элементов.
Прототип: int count(LIST list); где –
list – указатель на первый элемент списка.
Пример вызова: temp=count(list); где –
list – указатель на первый элемент списка.
temp – переменная целого типа.
Вызывающая функция: enterMenu, edit, deleteMenu, recordFile.
Возвращаемое значение: количество элементов в списке.

Описание переменных

Описание переменных функции count представлено на рисунке 22.

Имя переменной	Тип	Назначение
Локальные переменные		
Count	int	Количество элементов в списке
Формальные переменные		
list	LIST	Указатель на первый элемент списка

Рис. 22. Описание переменных функции count

7. Структура вызова функций

Структура вызова функций представлена на рисунке 23.

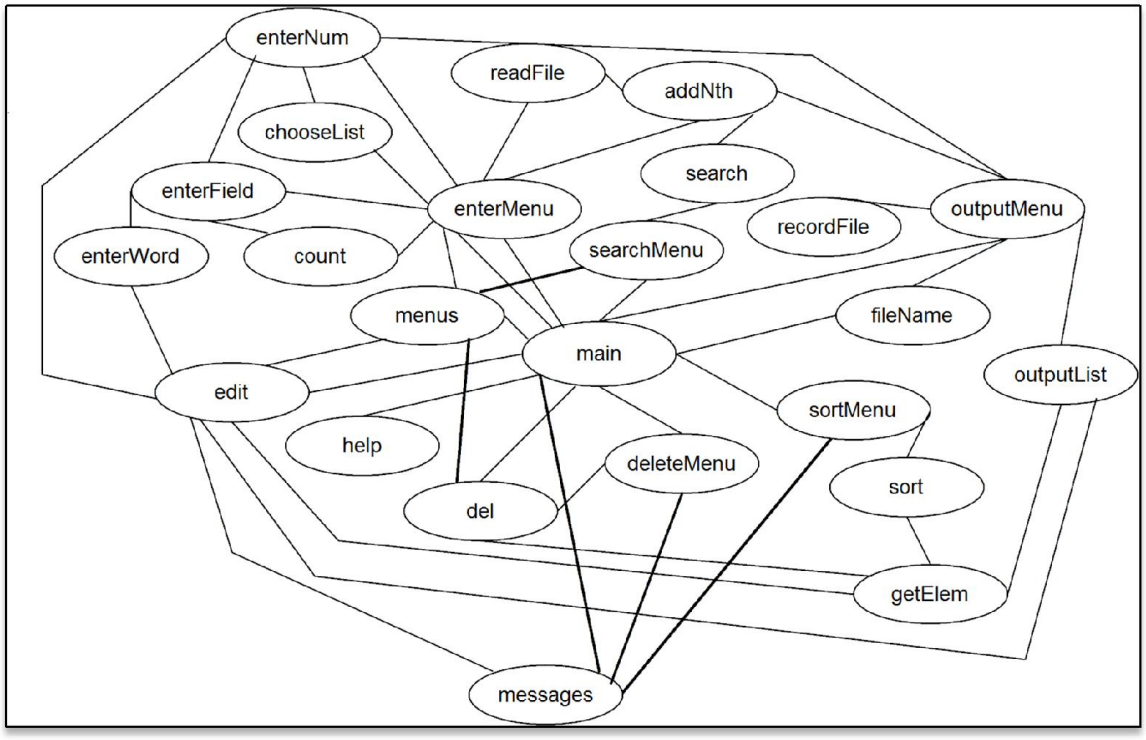

Рис. 23. Структура вызова функций

8. Текст программы с комментариями

```
#define _CRT_SECURE_NO_WARNINGS

#include <stdlib.h>
#include <stdio.h>
#include <locale.h>
#include <windows.h>
#include <conio.h>
#include <windows.h>

typedef struct stWood
  {
  char* Neck;                                          //Дерево грифа
  char* Deck;                                          //Дерево корпуса
  } WOOD;

typedef struct stGuitars
  {
  char* Name;                                          //Название гитары
  int Strings;                                         //Количество струн
  int Year;                                            //Год производства
  WOOD Wood;                                           //Дерево
  } GUITARS;

typedef struct stList
  {
  GUITARS Guitars;                                     //Структура с информационными полями
  struct stList *next;                                 //Следующий элемент
  struct stList *prev;                                 //Предыдущий элемент
  } sLIST;

typedef sLIST* LIST;                                   //Указатель на элемент списка

//------------------------------------------------------------------------------------------------
//-------------------------------------------------Прототипы функций------------------------------
//------------------------------------------------------------------------------------------------
int onLoad();                                            //Заставка
int help();                                              //Справка
```

```
int menus(int Key);                                          //Вывод меню
int chooseList();                                            //Выбор списка
int messages(int Key);                                       //Вывод сообщений
int enterMenu(LIST* list);                                   //Подменю ввода элементов
LIST addNth(LIST list, LIST temp, int n);                    //Добавление n-го элемента
int editMenu(LIST* list);                                    //Подменю редактирования элементов
LIST enterField();                                           //Ввод информационных полей
char* enterWord();                                           //Ввод слова до 10 символов
int enterNum(int first, int last);                           //Ввод целочисленных значений в
                                                             //диапазоне
int deleteMenu(LIST* list);                                  //Подменю удаления элементов
int del(LIST *list, int Key, int n);                         //Удаление n-го элемента
int sortMenu(LIST *list);                                    //Подменю сортировки
LIST sort(LIST list, int Key);                               //Сортировка
LIST searchMenu(LIST list);                                  //Подменю поиска
LIST search(LIST list, int Key, int num, char* str);         //Поиск
char* fileName(int Key);                                     //Ввод имени файла
int recordFile(LIST list, const char* name);                 //Запись в файл
int readFile(LIST *list, const char* name);                  //Чтение файла
int outputMenu(LIST list);                                   //Подменю вывода
int outputList(LIST list, int Key);                          //Вывод списка
LIST getElem(LIST list, int Key, int n);                     //Нахождение элементов
int count(LIST list);                                        //Подсчёт количества элементов
//-------------------------------------------------------------------------------------------------

int main()
  {
  system("mode con cols=80 lines=20");
  system("color 2");
  LIST list = NULL, newList = NULL, *tempList=NULL;
  int Q;
  system("chcp 1251");
  onLoad();
  do
    {
    switch (Q = menus(0))
      {
        case 0:
          help();
          break;
        case 1:
          enterMenu(&list);
          break;
        case 2:
          if (editMenu(&list))
            messages(9);
          break;
        case 3:
          if(deleteMenu(&list))
            messages(9);
          break;
        case 4:
          if (!list)
            {
            messages(9);
            break;
            }
          switch (chooseList())
            {
              case 1:
                outputMenu(list);
                break;
              case 2:
                outputMenu(newList);
                break;
              case 3:
                break;
            }
          break;
        case 5:
          del(&newList, 1, 0);
          newList = searchMenu(list);
          break;
```

```
        case 6:
          if (!list)
            {
            messages(9);
            break;
            }
          switch (chooseList())
            {
              case 1:
                tempList = &list;
                break;
              case 2:
                tempList = &newList;
                break;
            }
          if (tempList != NULL)
            {
            switch (sortMenu(tempList))
              {
                case 0:
                  messages(4);
                  break;
                case 1:
                  messages(9);
                  break;
                case 2:
                  break;
              }
            tempList = NULL;
            }
          break;
        case 7:
          messages(8);
          break;
      }
    }
    while (Q != 7);
    del(&list, 3, 0);
    del(&newList, 3, 0);
    return 0;
  }
//*****************************************************************************************
//Функция справка
int onLoad()
  {
  system("cls");
  puts("\t\t\t>>Электронная картотека гитар<<");
  puts("\t\t\t\t     _");
  puts("\t\t\t\t    / 7");
  puts("\t\t\t\t   /_(");
  puts("\t\t\t\t   |_|");
  puts("\t\t\t\t   |_|");
  puts("\t\t\t\t   |_|");
  puts("\t\t\t\t   |_| /\\");
  puts("\t\t\t\t /\\|=|/ /");
  puts("\t\t\t\t \\ |_| /");
  puts("\t\t\t\t  ) _  \\");
  puts("\t\t\t\t / |_|  \\");
  puts("\t\t\t\t/  -=-o /");
  puts("\t\t\t\t\\  /~\\_/");
  puts("\t\t\t\t \\/\n\n");
  Beep(657, 250);
  Beep(327, 150);
  Beep(327, 150);
  Beep(657, 250);
  Beep(619, 250);
  Beep(657, 250);
  Beep(520, 250);
  Beep(327, 150);
  Beep(327, 150);
  Beep(657, 250);
  Beep(619, 250);
  Beep(657, 250);
```

```
  Beep(491, 250);
  Beep(327, 150);
  Beep(327, 150);
  Beep(657, 250);
  Beep(619, 250);
  Beep(657, 250);
  Beep(739, 250);
  Beep(327, 150);
  Beep(327, 150);
  Beep(739, 250);
  Beep(657, 250);
  Beep(739, 250);
  Beep(784, 250);
  Beep(327, 150);
  Beep(327, 150);
  system("cls");
  puts("\n\n\t\t\t\tЭлектронная картотека");
  puts("\n\n Данная программа предназначена для создания картотеки и работы с ней.");
  puts(" Возможности:");
  puts("   *Создание картотеки с информацией о гитарах");
  puts("   *Добавление карточек в картотеку");
  puts("   *Редактирование карточек");
  puts("   *Сохранение картотеки в файл");
  puts("   *Чтение картотеки из файла");
  puts("   *Удаление карточек или/и списка");
  puts("   *Поиск по картотеке по выбранному параметру (формирование выборки)");
  puts("   *Сортировка картотеки по выбранному параметру");
  puts("   *Вывод картотеки на экран\n\n");
  system("pause");
  return 0;
  }
//*****************************************************************************
//Функция справка
int help()
  {
  int Q;
  do
    {
  system("cls");
  puts("\n\n\t\t\t\tЭлектронная картотека");
  puts("=============================================================================");
  puts("0 - Возможности");
  puts("1 - Как добавлять карточки?");
  puts("2 - Как редактировать карточки?");
  puts("3 - Как удалять карточки?");
  puts("4 - Как посмотреть карточки?");
  puts("5 - Как найти карточки?");
  puts("6 - Как отсортировать карточки?");
  puts("7 - Вернуться в главное меню");
  printf("Введите пункт меню - ");
    switch (Q = enterNum(0, 7))
      {
        case 0:
          system("cls");
          puts(" Возможности:");
          puts("   *Создание картотеки с информацией о гитарах");
          puts("   *Добавление карточек в картотеку");
          puts("   *Редактирование карточек");
          puts("   *Сохранение картотеки в файл");
          puts("   *Чтение картотеки из файла");
          puts("   *Удаление карточек или/и списка");
          puts("   *Поиск по картотеке по выбранному параметру (формирование выборки)");
          puts("   *Сортировка картотеки по выбранному параметру");
          puts("   *Вывод картотеки на экран\n\n");
          break;
        case 1:
          system("cls");
          puts("• Для входа в подменю добавления карточек необходимо выбрать 1-ый пункт меню.");
          puts("- Для добавления карточки на соответствующую позицию, выберите требуемый пункт
                подменю и следуйте подсказкам ввода.");
          puts("- Для добавления элементов из файла сохранённого на жёстком диске, выберите 4 - ый
                пункт подменю.");
          puts("- Для возврата в главное меню необходимо выбрать 5 - ый пункт подменю.");
```

```
          break;
        case 2:
          system("cls");
          puts("• Для входа в подменю редакции карточки необходимо выбрать 2-ой пункт меню.");
          puts("- Для того чтобы посмотреть под каким номером находится карточка выберите 1 - ый
                пункт подменю.");
          puts("- Для изменения соответствующего информационного поля n - ой карточки выберите
                требуемый пункт меню и следуйте подсказкам ввода.");
          puts("- Для возврата в главное меню выберите 7 - ой пункт меню.");
          break;
        case 3:
          system("cls");
          puts("• Для входа в подменю удаления карточки необходимо выбрать 3-ий пункт меню.");
          puts("- Для удаления карточки по соответствующей позиции, выберите требуемый пункт
                подменю.");
          puts("- Для удаления всей картотеки, выберите 4 - ый подменю.");
          puts("- Для возврата в главное меню выберите 5 - ый пункт подменю.");
          break;
        case 4:
          system("cls");
          puts("• Для входа в подменю вывода карточки списка необходимо выбрать 4-ый пункт меню.");
          puts("- Программа предложит выбрать, для какой картотеки выполнить вывод, введите «1» для
                вывода исходной или «2» для вывода сформированной.");
          puts("- Для вывода выбранной картотеки на экран выберите 1 - ый пункт подменю.");
          puts("- Для сохранения картотеки в файл выберите 2 - ой пункт подменю и следуйте
                подсказкам.");
          puts("- Для возврата в главное меню выберите 3 - ий пункт подменю.");
          break;
        case 5:
          system("cls");
          puts("• Для входа в подменю поиска карточек необходимо выбрать 5-ый пункт меню.");
          puts("- Для поиска карточек по параметру, выберите требуемый пункт подменю и следуйте
                подсказкам.");
          puts("- Для возврата в главное меню выберите 6 - ой пункт подменю.");
          break;
        case 6:
          system("cls");
          puts("• Для входа в подменю сортировки картотеки необходимо выбрать 6-ой пункт меню.");
          puts("- Программа предложит выбрать, для какой картотеки выполнить вывод, введите «1» для
                вывода исходной или «2» для вывода сформированной.");
          puts("- Для сортировки картотеки по параметру, выберите требуемый пункт подменю.");
          puts("- Для возврата в главное меню выберите 6 - ой пункт подменю.");
          break;
        case 7:
          return 0;
          break;
      }
     puts("\n\nЕсли возникли проблемы обращайтесь, пожалуйста, на электронную почту:");
     puts(" komdosh@gelezo2.ru\n");
     system("pause");
    }
  while (Q != 7);
  return 0;
  }
//*****************************************************************************************
//Функция меню
int menus(int Key)
  {
  system("cls");
  int first=0, last=0;
  switch (Key)
    {
      case 0:
        puts("Главное меню");
        puts("0 - Справка");
        puts("1 - Добавление карточек о гитарах");
        puts("2 - Редактирование карточек");
        puts("3 - Удаление карточек");
        puts("4 - Вывод картотеки");
        puts("5 - Поиск карточек по параметру");
        puts("6 - Сортировка карточек по параметру");
        puts("7 - Выход");
        first = 0;
```

```
        last = 7;
        break;
      case 1:
        puts("Меню добавления карточек");
        puts("1 - Добавить карточку в начало списка");
        puts("2 - Добавить карточку в конец списка");
        puts("3 - Добавить карточку на выбранную позицию");
        puts("4 - Прочитать картотеку с файла");
        puts("5 - Вернуться в главное меню");
        first = 1;
        last = 5;
        break;
      case 2:
        puts("Меню удаления карточки");
        puts("1 - Удалить первую карточку в картотеке");
        puts("2 - Удалить последнюю карточку в картотеке");
        puts("3 - Удалить карточку по её позиции");
        puts("4 - Удалить всю картотеку");
        puts("5 - Вернутся в главное меню");
        first = 1;
        last = 5;
        break;
      case 3:
        puts("Меню сортировки");
        puts("По какому пункту выполнить сортировку?");
        puts("1 - Название");
        puts("2 - Количество струн");
        puts("3 - Год производства");
        puts("4 - Дерево грифа");
        puts("5 - Дерево корпуса");
        puts("6 - Вернуться в главное меню");
        first = 1;
        last = 6;
        break;
      case 4:
        puts("Меню поиска");
        puts("По какому пункту сделать выборку?");
        puts("1 - Название");
        puts("2 - Количество струн");
        puts("3 - Год производства");
        puts("4 - Дерево грифа");
        puts("5 - Дерево корпуса");
        puts("6 - Вернуться в главное меню");
        first = 1;
        last = 6;
        break;
      case 5:
        puts("Меню вывода картотеки");
        puts("1 - Вывести картотеку на экран");
        puts("2 - Сохранить картотеку в файл");
        puts("3 - Вернуться в главное меню");
        first = 1;
        last = 3;
        break;
      case 6:
        puts("Меню редактирования карточек");
        puts("1 - Вывести картотеку на экран");
        puts("2 - Изменить марку n-ой карточки");
        puts("3 - Изменить количество струн n-ой карточки");
        puts("4 - Изменить год производства n-ой карточки");
        puts("5 - Изменить дерево грифа n-ой карточки");
        puts("6 - Изменить дерево корпуса n-ой карточки");
        puts("7 - Вернуться в главное меню");
        first = 1;
        last = 7;
        break;
    }
  printf("Введите номер пункта - ");
  return enterNum(first, last);
  }
//*****************************************************************************
//Функция выбора списка
int chooseList()
```

```
  {
  system("cls");
  puts("Для какой картотеки выполнить это действие?");
  puts("1 - Исходной");
  puts("2 - Сформированной");
  puts("3 - Вернуться в гланое меню");
  printf("Введите номер пункта (от %d до %d): ", 1, 3);
  return enterNum(1, 3);
  }
//*****************************************************************************
//Функция вывода сообщений пользователю
int messages(int Key)
  {
  system("cls");
  switch (Key)
    {
      case 3:
        puts("Картотека удалена");
        break;
      case 4:
        puts("Сортировка успешно завершена");
        puts("Для просмотра выберите пункт 'Вывод списка'");
        break;
      case 5:
        puts("Выборка из данных успешно сформирована");
        puts("Для просмотра выберите пункт 'Вывод' -> 'Сформированной'");
        break;
      case 6:
        puts("Выборка из данных не была сформирована");
        puts("В исходных данных не нашлось таких результатов");
        break;
      case 8:
        puts("До новых встреч!");
        break;
      case 9:
        puts("Картотека пуста");
        puts("Для выполнения этого действия, создайте картотеку, выбрав 1 пункт меню");
        break;
      case 10:
        puts("Элемент успешно удалён");
        break;
      case 11:
        puts("Элемент успешно добавлен");
        puts("Для просмотра выберите пункт 'Вывод' -> 'Исходной'");
        break;
      case 12:
        puts("Для того чтобы добавить элемент по позиции, необходимо наличие как минимум\n2-х
              элементов");
        break;
      case 13:
        puts("Для того чтобы удалить элемент по позиции, необходимо наличие как минимум\n2-х
              элементов");
        break;
      case 14:
        puts("Картотека успешно записана в файл");
        break;
      case 15:
        puts("Картотека успешно считана с файла и добавлена в исходную");
        break;
      case 17:
        puts("Файла с таким именем не существует");
        puts("Измените имя файла!");
        break;
      case 18:
        puts("Картотека не записана. Возникли проблемы с файлом, обратитесь к разработчику.");
        puts("Komdosh@gelezo2.ru");
        puts("В теме укажите 'Возникли проблемы с электронной картотекой'.");
        puts("В теле письма укажите, как появилась данная ошибка, и попытайтесь описать");
        puts("алгоритм ваших действий.");
        break;
      case 19:
        puts("Файл пустой");
        break;
```

```
      case 20:
        puts("Поле успешно изменено");
        break;
      case 777:
        puts("Возникла странная ошибка. Пожалуйста, напишите разработчику на электронную почту:");
        puts("Komdosh@gelezo2.ru");
        puts("В теме укажите 'Возникли проблемы с электронной картотекой'.");
        puts("В теле письма укажите, как появилась данная ошибка, и попытайтесь описать");
        puts("алгоритм ваших действий.");
        break;
    }
  system("pause");
  return 0;
  }
//*****************************************************************************************
//Функция подменю добавления элементов
int enterMenu(LIST* list)
  {
  LIST temp;
  int Q;
  do
    {
    switch (Q = menus(1))
      {
        case 1:
          temp = enterField();
          *list = addNth(*list, temp, 0);
          messages(11);
          break;
        case 2:
          if (*list)
            {
            temp = enterField();
            *list = addNth(*list, temp, count(*list));
            messages(11);
            }
          else
            messages(9);
          break;
        case 3:
          if (*list)
            if (count(*list)>1)
              {
              temp = enterField();
              printf("Введите номер позиции, куда вставить карточку (от %d до %d): ", 2,
                      (count(*list)>2) ? count(*list) - 1 : count(*list));
              *list = addNth(*list, temp, enterNum(2, count(*list)) - 1);
              messages(11);
              }
            else
              messages(12);
          else
            messages(9);
          break;
        case 4:
          switch (readFile(list, fileName(2)))
            {
              case 0:
                messages(15);
                break;
              case 1:
                messages(17);
                break;
              case 2:
                messages(19);
                break;
            }
          break;
      }
    }
   while (Q != 5);
   return 0;
  }
```

```
//*****************************************************************************************
//Функция добавления n-го
LIST addNth(LIST list, LIST temp, int n)
  {
  if (!n)
    {
    if (list)
      list->prev = temp;
    temp->next = list;
    temp->prev = NULL;
    return temp;
    }
  else
    {
    list = getElem(list, 2, n);
    if (list->next)
      {
      temp->next = list->next;
      temp->prev = list;
      list->next->prev = temp;
      }
    else
      {
      temp->next = NULL;
      temp->prev = list;
      }
    list->next = temp;
    return getElem(list, 0, 0);
    }
  }
//*****************************************************************************************
//Функция подменю редактирования элементов
int editMenu(LIST *list)
  {
  if (!*list)
    return 1;
  int Q;
    do
      {
      Q = menus(6);
      if (Q == 1)
        outputList(*list, 0);
      else
        {
        if (Q != 7)
          {
          printf("Введите номер элемента n - ");
          *list = getElem(*list, 2, enterNum(1, count(*list)));
          }
        }
      switch (Q)
        {
          case 2:
            printf("Введите марку гитары (кол-во символов от 1 до 10): ");
            (*list)->Guitars.Name = enterWord();
            break;
          case 3:
            printf("Введите количество струн (от %d до %d): ", 1, 20);
            (*list)->Guitars.Strings = enterNum(1, 20);
            break;
          case 4:
            printf("Введите год производства (от %d до %d): ", 1899, 2015);
            (*list)->Guitars.Year = enterNum(1899, 2015);
            break;
          case 5:
            printf("Введите название дерева грифа (кол-во символов от 1 до 10): ");
            (*list)->Guitars.Wood.Neck = enterWord();
            break;
          case 6:
            printf("Введите название дерева корпуса (кол-во символов от 1 до 10): ");
            (*list)->Guitars.Wood.Deck = enterWord();
            break;
        }
```

```
      if (Q != 7)
        messages(20);
      *list = getElem(*list, 0, 0);
      }
      while (Q != 7);
  return 0;
  }
//*****************************************************************************
//Функция ввода данных в поля
LIST enterField()
  {
  system("cls");
  LIST list = (LIST) malloc(sizeof(sLIST));
  printf("================================================================================");
  printf("Введите марку гитары\nКоличество символов от 1 до 10, в строке не может быть
         пробелов\nМарка:");
  list->Guitars.Name = enterWord();
  printf("================================================================================");
  printf("Введите количество струн (от %d до %d): ", 1, 20);
  list->Guitars.Strings = enterNum(1, 20);
  printf("================================================================================");
  printf("Введите год производства (от %d до %d): ", 1899, 2015);
  list->Guitars.Year = enterNum(1899, 2015);
  printf("================================================================================");
  printf("Введите название дерева грифа\nКоличество символов от 1 до 10, в строке не может быть
         пробелов\nДерево грифа:");
  list->Guitars.Wood.Neck = enterWord();
  printf("================================================================================");
  printf("Введите название дерева корпуса\nКоличество символов от 1 до 10, в строке не может быть
         пробелов\nДерево корпуса:");
  list->Guitars.Wood.Deck = enterWord();
  return list;
  }
//*****************************************************************************
//Функция ввода слова
char* enterWord()
  {
    char str[11], *temp;
    unsigned int i;
    do
    {
    gets(str);
    fflush(stdin);
    for (i = 0; i < strlen(str) && str[i] != ' '; i++);
    i = (i == strlen(str))?0:1;
    if (strlen(str)<1 || strlen(str)>10 || i)
      printf("Возможно вы ошиблись при вводе?\nКоличество символов от 1 до 10, в строке не может
              быть пробелов\nПовторите ввод: ");
    }
    while (strlen(str)<1 || strlen(str)>10 || i);
    temp = (char*) malloc(strlen(str)*sizeof(char));
    strcpy(temp, str);
    return temp;
  }
//*****************************************************************************
//Функция ввода целочисленных переменных в диапазоне
int enterNum(int first, int last)
  {
  int num;
  bool check_num, check_all;
  char str[5];
  const char numbers[] = "0123456789";
  do
    {
    check_all = true;
    check_num = false;
    scanf("%s", &str);
    fflush(stdin);
    for (int i = 0; str[i] != '\0' && check_all; i++)
      {
      for (int j = 0; numbers[j] != '\0' && !check_num; j++)
      if (str[i] == numbers[j] || str[i] == '\0')
        check_num = true;
```

```
      if (check_num)
        check_num = false;
      else
        check_all = false;
      }
    if (check_all)
      num = atoi(str);
    else
      printf("В строку попало что-то кроме числа, повторите ввод:\n");
    if ((num < first || num > last) && check_all)
      printf("Возможно вы ошиблись при вводе?\nВведите число от %d до %d\nПовторите ввод: ", first,
             last);
    }
    while (num < first || num > last || !check_all);
    return num;
  }
//*****************************************************************************************
//Функция подменю удаления элементов
int deleteMenu(LIST* list)
  {
   int Q;
   if (!*list)
       return 1;
   do
      {
      switch (Q = menus(2))
        {
          case 1:
            if (!del(list, 0, 0))
              messages(10);
            else
              messages(9);
            break;
          case 2:
            if (!del(list, 1, 0))
              messages(10);
            else
              messages(9);
            break;
          case 3:
            if (count(*list) > 1)
              {
              printf("Введите номер позиции карточки, которую следует удалить (от %d до %d): ", 2,
                      (count(*list) > 2) ? count(*list) - 1 : count(*list));
              if (!del(list, 2, enterNum(2, count(*list))))
                messages(10);
              else
                messages(9);
              }
            else
              messages(13);
            break;
          case 4:
            if (!del(list, 3, 0))
              messages(3);
            else
              messages(9);
            break;
        }
      }
   while (Q != 5);
   return 0;
  }
//*****************************************************************************************
//Функция удаления n-го элемента в списке
int del(LIST *list, int Key, int n) //Key: 0 - начало, 1 - конец, 2 - н-ый, 3 - весь список
  {
  if (!*list)
    return 1;
  LIST temp;
  switch (Key)
    {
      case 0:
```

```
        if ((*list)->next)
          {
          free((*list)->Guitars.Name);
          free((*list)->Guitars.Wood.Deck);
          free((*list)->Guitars.Wood.Neck);
          *list = (*list)->next;
          free((*list)->prev);
          (*list)->prev = NULL;
          }
        else
          {
          free((*list)->Guitars.Name);
          free((*list)->Guitars.Wood.Deck);
          free((*list)->Guitars.Wood.Neck);
          free(*list);
          *list = NULL;
          }
        return 0;
        break;
      case 1:
        temp = getElem(*list, 1, 0);
        if (!temp->prev)
          *list = NULL;
        else
          temp->prev->next = NULL;
        free(temp->Guitars.Name);
        free(temp->Guitars.Wood.Deck);
        free(temp->Guitars.Wood.Neck);
        free(temp);
        return 0;
        break;
      case 2:
        temp = getElem(*list, 2, n);
        temp->prev->next = temp->next;
        temp->next->prev = temp->prev;
        free(temp->Guitars.Name);
        free(temp->Guitars.Wood.Deck);
        free(temp->Guitars.Wood.Neck);
        free(temp);
        return 0;
        break;
      case 3:
        for (; (*list)->next; (*list) = (*list)->next, free((*list)->prev->Guitars.Name),
             free((*list)->prev->Guitars.Wood.Deck), free((*list)->prev->Guitars.Wood.Neck),
             free((*list)->prev));
        free((*list)->Guitars.Name);
        free((*list)->Guitars.Wood.Deck);
        free((*list)->Guitars.Wood.Neck);
        free(*list);
        (*list) = NULL;
        return 0;
        break;
    }
  return 0;
  }
//*****************************************************************************************
//Функция подменю сортировки
int sortMenu(LIST *list)
  {
  if (!*list)
    return 1;
  int Q = menus(3);
  if (Q == 6)
    return 2;
  else
    *list = sort(*list, Q);
  return 0;
  }
//*****************************************************************************************
//Функция сортировки данных
LIST sort(LIST list, int Key)
  {
  LIST temp, sort = NULL, pFwd, pBwd;
```

```
  bool check = true;
  while (list)
    {
    temp = list;
    list = list->next;
    for (pFwd = sort, pBwd = NULL; pFwd && (Key == 1 && strcmp(pFwd->Guitars.Name,
         temp->Guitars.Name)>0 || Key == 2 && temp->Guitars.Strings > pFwd->Guitars.Strings ||
         Key == 3 && temp->Guitars.Year > pFwd->Guitars.Year || Key == 4 &&
         strcmp(pFwd->Guitars.Wood.Neck, temp->Guitars.Wood.Neck)>0 ||
         Key == 5 && strcmp(pFwd->Guitars.Wood.Deck, temp->Guitars.Wood.Deck)>0); pBwd = pFwd,
         pFwd = pFwd->next);
    if (!pBwd)
      {
      temp->next = sort;
      sort = temp;
      sort->prev = NULL;
      check = false;
      }
    else
      {
      temp->next = pFwd;
      pBwd->next = temp;
      }
    }
  if (check)
    return getElem(list, 0, 0);
  for (list = sort, sort = sort->next; sort->next; sort->prev = list, list = list->next,
       sort = sort->next);
  return getElem(sort, 0, 0);
  }
//*****************************************************************************************
//Функция подменю обработки
LIST searchMenu(LIST list)
  {
  if (!list)
    {
    messages(9);
    return NULL;
    }
  LIST New_list = NULL;
  const char* str = NULL;
  char* tempstr = NULL;
  int Q, temp=0, first=0, last=0;
  system("cls");
  switch (Q = menus(4))
    {
      case 1:
        str = "названию";
        break;
      case 2:
        first = 1;
        last = 20;
        str = "до какого количества струн";
        break;
      case 3:
        first = 1899;
        last = 2015;
        str = "до какого года производства";
        break;
      case 4:
        str = "дереву грифа";
        break;
      case 5:
        str = "дереву корпуса";
        break;
    }
  if (Q == 2 || Q == 3)
    {
    printf("Введите %s выводить результаты (от %d до %d): ", str, first, last);
    temp = enterNum(first, last);
    }
  else
    {
```

```
    if (Q != 6)
      {
      printf("Введите по какому %s выводить результаты - ", str);
      tempstr = enterWord();
      }
    }
  if (Q!=6)
    {
    New_list = search(list, Q, temp, tempstr);
    if (New_list)
      messages(5);
    else
      messages(6);
    }
  return New_list;
  }
//*****************************************************************************************
//Функция поиска данных
LIST search(LIST list, int Key, int num, char* str)
  {
  LIST search = NULL, temp = NULL;
  for (; list; list = list->next)
  if (Key == 1 && !strcmp(list->Guitars.Name, str) || Key == 2 && list->Guitars.Strings <= num ||
      Key == 3 && list->Guitars.Year <= num || Key == 4 && !strcmp(list->Guitars.Wood.Neck, str)
      || Key == 5 && !strcmp(list->Guitars.Wood.Deck, str))
    {
    temp = (LIST) malloc(sizeof(sLIST));
    temp->Guitars = list->Guitars;
    search = addNth(search, temp, 0);
    }
   return search;
  }
//*****************************************************************************************
//Функция ввода названия файла
char* fileName(int Key)
  {
  system("cls");
  char *name;
  switch (Key)
    {
      case 1:
        puts("Задайте имя файла");
        puts("Если такой файл не существует, он будет создан, иначе - перезаписан");
        break;
      case 2:
        puts("!Если имеется исходная картотека, то элементы будут добавлены к ней");
        puts("1 - Ввести название файла с клавиатуры");
        puts("2 - Открыть демонстрационный вариант");
        printf("Введите номер пункта - ");
        if (!(enterNum(1, 2) - 1))
          puts("Введите имя файла");
        else
          {
          name = (char*) malloc(6*sizeof(char));
          strcpy(name, "Guitars");
          return name;
          }
        break;
    }
  printf("Имя файла - ");
  name = enterWord();
  return name;
  }
//*****************************************************************************************
//Функция записи в файл
int recordFile(LIST list, const char* name)
  {
  FILE* file;
  system("cls");
  if (file = fopen(name, "w"))
    {
    fprintf(file, "%d\n", count(list));
    while (list)
```

```
      {
      fprintf(file, "%s %d %d %s %s\n", list->Guitars.Name, list->Guitars.Strings,
              list->Guitars.Year, list->Guitars.Wood.Neck, list->Guitars.Wood.Deck);
      list = list->next;
      }
    fclose(file);
    return 0;
    }
  else
    return 1;
  }
//*****************************************************************************************
//Функция чтения файла
int readFile(LIST *list, const char* name)
  {
  LIST temp;
  FILE* file;
  char tempChar;
  int a=1;
  system("cls");
  if (!(file = fopen(name, "r")))
    return 1;         //Файла не существует
  fseek(file, 0, SEEK_END);
  if (!ftell(file))
    return 2;         //Файл пустой
  rewind(file);
  int Count, i;
  fscanf(file, "%d", &Count);
  fgetc(file);
  temp = (LIST) malloc(sizeof(sLIST));
  while (Count)
    {
    temp = (LIST) malloc(sizeof(sLIST));
    temp->Guitars.Name = NULL;
    for (i = 0, tempChar = fgetc(file); tempChar != ' '; tempChar = fgetc(file), i++)
      {
      temp->Guitars.Name = (char*) realloc(temp->Guitars.Name, (i + 1)*sizeof(char));
      temp->Guitars.Name[i] = tempChar;
      }
    temp->Guitars.Name[i] = '\0';
    fscanf(file, "%d", &(temp->Guitars.Strings));
    fgetc(file);
    fscanf(file, "%d", &(temp->Guitars.Year));
    fgetc(file);
    temp->Guitars.Wood.Neck = NULL;
    for (i = 0, tempChar = fgetc(file); tempChar != ' '; tempChar = fgetc(file), i++)
      {
      temp->Guitars.Wood.Neck = (char*) realloc (temp->Guitars.Wood.Neck, (i + 1)*sizeof(char));
      temp->Guitars.Wood.Neck[i] = tempChar;
      }
    temp->Guitars.Wood.Neck[i] = '\0';
    temp->Guitars.Wood.Deck = NULL;
    for (i = 0, tempChar = fgetc(file); tempChar != '\n'; tempChar = fgetc(file), i++)
      {
      temp->Guitars.Wood.Deck = (char*) realloc(temp->Guitars.Wood.Deck, (i + 1)*sizeof(char));
      temp->Guitars.Wood.Deck[i] = tempChar;
      }
    temp->Guitars.Wood.Deck[i] = '\0';
    if (!*list)
      *list = addNth(*list, temp, 0);
    else
      *list = addNth(*list, temp, count(*list));
    Count--;
    }
  fclose(file);
  return 0;
  }
//*****************************************************************************************
//Функция подменю вывода
int outputMenu(LIST list)
  {
  if (list)
    {
```

```
    switch (menus(5))
      {
        case 1:
          system("cls");
          puts("Список с начала или с конца?");
          puts("1 - С начала");
          puts("2 - С конца");
          printf("Введите номер пункта (от %d до %d): ", 1, 2);
          if(outputList(list, enterNum(1, 2) - 1))
            messages(9);
          break;
        case 2:
          if (!recordFile(list, fileName(1)))
            messages(14);
          else
            messages(18);
          break;
      }
    }
  else
    messages(9);
  return 0;
  }
//*****************************************************************************************
//Функция вывода данных
int outputList(LIST list, int Key) //Key: 0 - с начала, 1 - с конца
  {
  if(!list)
    return 1;
  system("mode con cols=85 lines=47");
  if (Key)
    list = getElem(list, 1, 0);
  system("cls");
  int i = 1;
  printf("=========================================================================================");
  printf("%2s | %12s | %18s | %14s | %17s\n", " ", " ", " ", "Количество ", "Дерево:");
  printf("%2s | %12s | %18s | %14s | %s\n","№", "Название  ", "Год производства ", "струн    ",
         "_____________________________");
  printf("%2s | %12s | %18s | %14s | %11s | %6s\n", " ", " ", " ", " ", " ", "Корпус  ", "Гриф");
  printf("=========================================================================================");
  while (list)
    {
    printf("%2d | %12s | %18d | %14d | %11s | %6s ", i, list->Guitars.Name, list->Guitars.Year,
           list->Guitars.Strings, list->Guitars.Wood.Deck, list->Guitars.Wood.Neck);

printf("\n=========================================================================================");
    if (list->prev && Key || list->next && !Key)
      printf("Для вывода следующего элемента нажмите любую клавишу, для выхода нажмите Esc\r");
    else
      puts("Для завершения просмотра нажмите Esc");
    if (_getch() == 27)
      return 0;
    list = (Key) ? list->prev : list->next;
    i++;
    }
  do
    {

    }
  while (_getch() != 27);
  system("mode con cols=80 lines=20");
  return 0;
  }
//*****************************************************************************************
//Функция нахождения последнего элемента
LIST getElem(LIST list, int Key, int n) //Key: 0 - начало, 1 - конец, 2 - n-ый
  {
  for (; list->prev && Key == 0; list = list->prev);
  for (; list->next && Key == 1; list = list->next);
  for (int i = 1; i < n && list->next && Key == 2; i++, list = list->next);
  return list;
  }
//*****************************************************************************************
```

```
   //Функция подсчёта количества элементов
   int count(LIST list)
     {
     int Count;
     for (Count = 0; list; list = list->next, Count++);
     return Count;
     }    {
       temp = (LIST) malloc(sizeof(sLIST));
       temp->Guitars = list->Guitars;
       search = addNth(search, temp, 0);
       }
      return search;
  }
//*****************************************************************************************
//Функция ввода названия файла
char* fileName(int Key)
  {
     system("cls");
     char *name;
     switch (Key)
       {
         case 1:
           puts("Задайте имя файла");
           puts("Если такой файл не существует, он будет создан, иначе - перезаписан");
           break;
         case 2:
           puts("1 - Ввести название файла с клавиатуры");
           puts("2 - Открыть демонстрационный вариант");
           printf("Введите номер пункта - ");
           if (!(enterNum(1,2)-1))
             puts("Введите имя файла");
           else
             {
             name = (char*) malloc(6*sizeof(char));
             strcpy(name, "Guitars");
             return name;
             }
           break;
       }
     printf("Имя файла - ");
     name = enterWord();
     return name;
  }
//*****************************************************************************************
//Функция записи в файл
int recordFile(LIST list, const char* name)
  {
     FILE* file;
     system("cls");
     if (file = fopen(name, "w"))
       {
       fprintf(file, "%d\n", count(list));
       while (list)
         {
         fprintf(file, "%s %d %d %s %s\n", list->Guitars.Name, list->Guitars.Strings,
                 list->Guitars.Year, list->Guitars.Wood.Neck, list->Guitars.Wood.Deck);
         list = list->next;
         }
       fclose(file);
       return 0;
       }
     else
       return 1;
  }
//*****************************************************************************************
//Функция чтения файла
int readFile(LIST *list, const char* name)
  {
     LIST temp;
     FILE* file;
     char tempChar;
     system("cls");
     if (!(file = fopen(name, "r")))
```

```
      return 1;         //Файла не существует
    fseek(file, 0, SEEK_END);
    if (!ftell(file))
      return 2;         //Файл пустой
    rewind(file);
    int Count, i;
    fscanf(file, "%d", &Count);
    fgetc(file);
    temp = (LIST) malloc(sizeof(sLIST));
    while (Count)
      {
      temp = (LIST) malloc(sizeof(sLIST));
      temp->Guitars.Name = NULL;
      for (i = 0, tempChar = fgetc(file); tempChar != ' '; tempChar = fgetc(file), i++)
        {
        temp->Guitars.Name = (char*) realloc(temp->Guitars.Name, (i + 1)*sizeof(char));
        temp->Guitars.Name[i] = tempChar;
        }
      temp->Guitars.Name[i] = '\0';
      fscanf(file, "%d", &(temp->Guitars.Strings));
      fgetc(file);
      fscanf(file, "%d", &(temp->Guitars.Year));
      fgetc(file);
      temp->Guitars.Wood.Neck = NULL;
      for (i = 0, tempChar = fgetc(file); tempChar != ' '; tempChar = fgetc(file), i++)
        {
        temp->Guitars.Wood.Neck = (char*) realloc (temp->Guitars.Wood.Neck, (i + 1)*sizeof(char));
        temp->Guitars.Wood.Neck[i] = tempChar;
        }
      temp->Guitars.Wood.Neck[i] = '\0';
      temp->Guitars.Wood.Deck = NULL;
      for (i = 0, tempChar = fgetc(file); tempChar != '\n'; tempChar = fgetc(file), i++)
        {
        temp->Guitars.Wood.Deck = (char*) realloc(temp->Guitars.Wood.Deck, (i + 1)*sizeof(char));
        temp->Guitars.Wood.Deck[i] = tempChar;
        }
      temp->Guitars.Wood.Deck[i] = '\0';
      if (!*list)
        *list = addNth(*list, temp, 0);
      else
        *list = addNth(*list, temp, count(*list));
      Count--;
      }
    fclose(file);
    return 0;
  }
//*****************************************************************************************
//Функция подменю вывода
int outputMenu(LIST list)
  {
     if (list)
       {
       switch (menus(5))
         {
           case 1:
             system("cls");
             puts("Список с начала или с конца?");
             puts("1 - С начала");
             puts("2 - С конца");
             printf("Введите номер пункта (от %d до %d): ", 1, 2);
             if(outputList(list, enterNum(1, 2) - 1))
               messages(9);
             break;
           case 2:
             if (!recordFile(list, fileName(1)))
               messages(14);
             else
               messages(18);
             break;
         }
       }
     else
       messages(9);
```

```
    return 0;
  }
//*****************************************************************************************
//Функция вывода данных
int outputList(LIST list, int Key) //Key: 0 - с начала, 1 - с конца
  {
    if(!list)
      return 1;
    system("mode con cols=85 lines=47");
    if (Key)
      list = getElem(list, 1, 0);
    system("cls");
    int i=1;
    printf("=========================================================================================");
    printf("%2s | %12s | %18s | %14s | %17s\n", " ", " ", " ", "Количество ", "Дерево:");
    printf("%2s | %12s | %18s | %14s | %s\n","№", "Название  ", "Год производства ", "струн    ",
           "_____________________");
    printf("%2s | %12s | %18s | %14s | %11s | %6s\n", " ", " ", " ", " ", " ", "Корпус  ", "Гриф");
    printf("=========================================================================================");
    while (list)
      {
      printf("%2d | %12s | %18d | %14d | %11s | %6s ", i, list->Guitars.Name, list->Guitars.Year,
              list->Guitars.Strings, list->Guitars.Wood.Deck, list->Guitars.Wood.Neck);
    printf("\n=========================================================================================");
      if (list->prev && Key || list->next && !Key)
        printf("Для вывода следующего элемента нажмите любую клавишу\r");
      else
        puts("Для завершения просмотра нажмите любую клавишу");
      _getch();
      list = (Key) ? list->prev : list->next;
      i++;
      }
    system("mode con cols=80 lines=20");
    return 0;
  }
//*****************************************************************************************
//Функция нахождения последнего элемента
LIST getElem(LIST list, int Key, int n) //Key: 0 - начало, 1 - конец, 2 - n-ый
  {
    for (; list->prev && Key == 0; list = list->prev);
    for (; list->next && Key == 1; list = list->next);
    for (int i = 1; i < n && list->next && Key == 2; i++, list = list->next);
    return list;
  }
//*****************************************************************************************
//Функция подсчёта количества элементов
int count(LIST list)
  {
    int Count;
    for (Count = 0; list; list = list->next, Count++);
    return Count;
  }
```

9. Инструкция пользователю

- Для вызова справки необходимо выбрать 0-ой пункт меню.
  Для входа в подменю добавления карточек необходимо выбрать 1-ый пункт меню.
  - Для добавления карточки на соответствующую позицию, выберите требуемый пункт подменю и следуйте подсказкам ввода.
  - Для добавления элементов из файла сохранённого на жёстком диске, выберите 4 - ый пункт подменю.
  - Для возврата в главное меню необходимо выбрать 5-ый пункт подменю.
  Для входа в подменю редакции карточки необходимо выбрать 2-ой пункт меню.
  - Для того чтобы посмотреть под каким номером находится карточка выберите 1 - ый пункт подменю.
  - Для изменения соответствующего информационного поля n - ой карточки выберите требуемый пункт меню и следуйте подсказкам ввода.
  - Для возврата в главное меню выберите 7-ой пункт меню.

- Для входа в подменю удаления карточки необходимо выбрать 3-ий пункт меню.
  - Для удаления карточки по соответствующей позиции, выберите требуемый пункт подменю.
  - Для удаления всей картотеки, выберите 4 - ый подменю.
  - Для возврата в главное меню выберите 5-ый пункт подменю.
- Для входа в подменю вывода картотеки списка необходимо выбрать 4-ый пункт меню.
  - Программа предложит выбрать, для какой картотеки выполнить вывод, введите «1» для вывода исходной или «2» для вывода сформированной.
  - Для вывода выбранной картотеки на экран выберите 1 - ый пункт подменю.
  - Для сохранения картотеки в файл выберите 2 - ой пункт подменю и следуйте подсказкам.
  - Для возврата в главное меню выберите 3-ий пункт подменю.
- Для входа в подменю поиска карточек необходимо выбрать 5-ый пункт меню.
  - Для поиска карточек по параметру, выберите требуемый пункт подменю и следуйте подсказкам.
  - Для возврата в главное меню выберите 6-ой пункт подменю.
- Для входа в подменю сортировки картотеки необходимо выбрать 6-ой пункт меню.
  - Программа предложит выбрать, для какой картотеки выполнить вывод, введите «1» для вывода исходной или «2» для вывода сформированной.
  - Для сортировки картотеки по параметру, выберите требуемый пункт подменю.
  - Для возврата в главное меню выберите 6-ой пункт подменю.
- Для выхода из программы необходимо выбрать 7-ой пункт меню.

10. Набор тестов

Проверка программы выполняется с помощью тестов. Тестирование – проверка определённой части программы, сравнение результатов выданных программой для специально выбранных исходных данных, с ожидаемыми результатами.

Тест 1. Проверка правильности работы всей программы.

- Вводим контрольные примеры из рисунка 1, и получаем результаты, совпадающие с ожидаемыми значениями из рисунка.

Тест 2. Проверка правильности работы функции добавления.

- Пытаемся добавить элемент в конец списка, не добавив первый элемент, получаем сообщение о том, что список пуст.
- Пытаемся добавить элемент на n-ую позицию, не добавив первый элемент, получаем сообщение о том, что список пуст.
- Пытаемся добавить элемент на 7-ую позицию, добавив только один элемент, получаем сообщение об ошибке с подсказкой.
- Пытаемся добавить элемент на 5-ую позицию, добавив только 3 элемента, получаем сообщение об ошибке с подсказкой, сколько всего элементов в списке.

Тест 3. Проверка правильности работы функции редактирования.

- Пытаемся войти в подменю редактирования, не добавив ни один элемент, получаем сообщение о том, что список пуст.
- Пытаемся отредактировать название 5-ого элемента в списке из 3-х элементов, получаем сообщение об ошибке с подсказкой, сколько всего элементов в списке.

Тест 4. Проверка правильности работы функции удаления.

- Пытаемся удалить первый элемент из пустого списка, получаем сообщение о том, что список пуст.
- Пытаемся удалить последний элемент из пустого списка, получаем сообщение о том, что список пуст.
- Пытаемся удалить 7-ой элемент из пустого списка, получаем сообщение об ошибке с подсказкой.
- Пытаемся очистить пустой список, получаем сообщение об ошибке с подсказкой.

- Пытаемся удалить 7-ой элемент из списка, состоящего из 4-х элементов, получаем сообщение об ошибке с подсказкой.

Тест 5. Проверка правильности работы функции вывода.

- Пытаемся вывести элементы исходного списка на экран, выбираем исходный список, не добавив ни одного элемента, получаем сообщение о том, что список пуст.
- Пытаемся вывести элементы исходного списка на экран, выбираем исходный список, не добавив ни одного элемента, получаем сообщение о том, что список пуст.
- Пытаемся вывести элементы сформированного списка на экран, выбираем сформированный список, не выполнив поиск, получаем сообщение о том, что список пуст.

Тест 6.Проверка правильности работы функций отвечающих за работу с файлами.

- Пытаемся открыть несуществующий файл, получаем соответствующую ошибку.
- Пытаемся открыть пустой файл, получаем соответствующе сообщение об ошибке.

Тест 7. Проверка правильности работы функции ввода целочисленных значений в диапазоне.

- Пытаемся ввести числовые значения, которые выходят за пределы заданного диапазона, получаем сообщение об ошибке с подсказкой.
- Пытаемся ввести в поле для числа - слово, получаем соответствующее сообщение об ошибке.
- Пытаемся ввести дробное значение числа, получаем сообщение об ошибке.

Тест 8. Проверка правильности работы функции ввода слова длины 10.

- Пытаемся ввести строку с 11 символами, получаем соответствующее сообщение об ошибке.
- Пытаемся ввести строку с пробелом, получаем сообщение об ошибке.
- Пытаемся оставить поле пустым, получаем сообщение об ошибке.

11. Результаты решения задачи

При выполнении программы были получены результаты, совпадающие со значениями, приведенными на рисунке 1. Также были проведены тесты из п. набор тестов. Ошибок не обнаружено.

Вывод

При выполнении курсовой работы были получены практические навыки работы с электронной картотекой на языке программирования «C\C++».