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

Отчёт

по лабораторной работе № 3

на тему:

“Двусвязные списки”

по дисциплине “Программирование. Дополнительные главы”

Выполнил: студент гр. 4306 Табаков А.В.

Принял: к.т.н., доцент Сискович Т.И.

Санкт-Петербург  
2015 г.

**Цель**

Получить практические навыки работы с двусвязными списками.

**Задание**

Написать программу для: создания списка, контрольного вывода, удаления элементов списка, поиска, формирования нового списка, сортировки и вывода результата поиска.

**Уточнение задания**

В программе должно быть использовано простейшее меню. Выполнение программы должно быть многократным по желанию пользователя. Пользователь добавляет элемент в список и вводит данные в информационные поля структур. Структура содержит информационные поля о гитарах. Условия для обработки – поиск элементов в списке по значениям полей структур, есть возможность сортировки, вывод результата.

**Описание структуры**

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

**Контрольные примеры**

Контрольные примеры представлены на рисунке 1*.*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **№ примера** | **Исходные данные** | | | | | | | **Результат** | | | | |
|
| **Марка** | **Год производства** | **Кол-во струн** | **Материал** | | **Критерии поиска** | | **Марка** | **Год производства** | **Кол-во струн** | **Материал** | |
| **корпус** | **гриф** | **Strings** | **Year** | **корпус** | **гриф** |
| **1** | 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 |  | | | | |
| **2** | Gibson | 1964 | 6 | Ольха | Кедр | 7 |  | Dean | 1991 | 7 | Липа | Клён |
| Fender | 1983 | 6 | Сосна | Клён |  | | | | |
| Dean | 1991 | 7 | Липа | Клён |
| **3** | Gibson | 1964 | 6 | Ольха | Кедр |  | 1964 | Gibson | 1964 | 6 | Ольха | Кедр |
| Fender | 1983 | 6 | Сосна | Клён |  | | | | |
| Dean | 1991 | 7 | Липа | Клён |

Рис. 1. Контрольные примеры

**Описание главной функции**

Назначение: организация управления порядком вызова функций.

**Описание переменных функции**

Описание переменныхпредставлено вТаблице 1.

Таблица 1*.*Описание переменных главной функции

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| list | LIST | Указатель на первый элемент исходного списка |
| New\_list | LIST | Указатель на первый элемент сформированного списка |
| Q | int | Переменная выбора меню |

**Описание функций**

**Описание функции help**

Назначение: вывод справки.

Прототип: void help();

Пример вызова: help();

Вызывающая функция: main.

**Описание функции menu**

Назначение: вывод меню программы.

Прототип: void menu();

Возвращаемое значение: номер пункта меню.

Пример вызова: menu();

Вызывающая функция: main.

**Описание функции messages**

Назначение: Функция используется для ввода сообщений пользователю.

Прототип: void messages(int Key); описание формальных переменных представлено на рисунке 2.

Пример вызова: messages(1);

Вызывающая функция: main, enterMenu, deleteMenu, sortMenu, processingMenu, outputList.

Сообщения:

messages(1): "Сначала необходимо ввести данные"

messages(2): "Вы ввели данные, но не создали выборку"

"Вам необходимо выбрать 5 пункт меню для поиска данных"

messages(3): "Список удалён"

messages(4): "Сортировка успешно завершена "

"Для просмотра выберите пункт 'Вывод списка'"

messages(5): "Выборка из данных успешно сформирована"

"Для просмотра выберите пункт 'Вывод списка' -> 'Сформированный список'"

messages(6): "Выборка из данных не была сформирована"

"В исходных данных не нашлось таких результатов"

messages(7): "Что-то пошло не так, введите пункт меню повторно"

messages(8): "До новых встреч!"

messages(9): "Список пуст"

"Для выполнения этого действия, создайте список"

messages(10): "Элемент успешно удалён"

messages(11): "Элемент успешно добавлен"

"Для просмотра выберите пункт 'Вывод списка' -> '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"

**Описание переменных**

Описание переменных функции messages представлено на рисунке 2.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Формальные переменные | | |
| Key | int | Вспомогательная переменная |

Рис. 2. Описание переменных функции messages

**Описание функции chooseList**

Назначение: выбор списка.

Прототип: int chooseList();

Возвращаемое значение: ключ(0 – исходный список, 1 – сформированный список)

Пример вызова: Key=chooseList();

Вызывающая функция: main.

Вызываемая функция: enterNum.

**Описание функции from**

Назначение: список с конца или с начала.

Прототип: int from();

Возвращаемое значение: ключ(0 – с начала, 1 – с конца)

Пример вызова: From=from();

Вызывающая функция: outputMenu.

Вызываемая функция: enterNum.

**Описание функции enterMenu**

Назначение: организация управления порядком вызова функций добавления элемента в список.

Прототип:void enterMenu(LIST\* list); описание формальных переменных представлено на рис. 3.

Пример вызова: enterMenu(&list); описание фактических переменных представлено в таблице 1.

Вызывающая функция: main.

Вызываемая функция: addNth, enterNum, messages, count.

**Описание переменных**

Описание переменных функции enterMenu представлено на рисунке 3.

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

Рис. 3*.*Описание переменных функции enterMenu

**Пункты меню**

1: Добавить элемент в начало списка

2: Добавить элемент в конец списка

3: Добавить элемент на выбранную позицию

4: Вернуться в главное меню

**Описание функции enterNum**

Назначение: ввод чисел в заданном диапазоне.

Прототип: int enterNum(int first, int last); описание формальных переменных представлено на рис.4.

Возвращаемое значение: целое число.

Пример вызова: Q=enterNum(1, 7);

Вызывающая функция: main, chooseList, from, enterMenu, enterField, deleteMenu, sortMenu processingMenu.

**Описание переменных**

Описание переменных функции enterNum представлено на рисунке 4.

|  |  |  |
| --- | --- | --- |
| **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. 4*.*Описание переменных функции enterNum

**Описание функции outputMenu**

Назначение: подменю вывода списка.

Прототип: void outputList (LIST list); описание формальных переменных представлено на рис. 5.

Пример вызова: outputList (list); описание фактических переменных представлено в таблице 1.

Вызывающая функция: main.

Вызываемая функция: messages, from.

**Описание функции outputList**

Назначение: вывод списка.

Прототип: void outputList (LIST list); описание формальных переменных представлено на рис. 5.

Пример вызова: outputList (list); описание фактических переменных представлено в таблице 1.

Вызывающая функция: main.

Вызываемая функция: messages, getElem.

**Описание переменных**

Описание переменных функции outputList представлено на рисунке 5.

|  |  |  |
| --- | --- | --- |
| **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. 5*.*Описание переменных функции outputList

**Описание функции addFirst**

Назначение: добавление 1-го элемента.

Прототип: void addFirst(LIST \*list); описание формальных переменных представлено на рис. 6.

Пример вызова: addFirst (&list); описание фактических переменных представлено в таблице 1.

Вызывающая функция: enterMenu.

Вызываемая функция: enterField.

**Описание переменных**

Описание переменных функции addFirst представлены на рисунке 6.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Локальные переменные | | |
| temp | LIST | Вспомогательная переменная |
| Формальные переменные | | |
| list | LIST\* | Указатель на адрес первого элемента исходного списка |

Рис. 6*.*Описание переменных функции addFirst

**Описание функции addNth**

Назначение: добавление n-го элемента.

Прототип: void addNth(LIST list, int n); описание формальных переменных представлено на рисунке 7.

Пример вызова: addNth(list, 2); описание фактических переменных представлено в таблице 1.

Вызывающая функция: enterMenu.

Вызываемая функция: enterField, getElem.

**Описание переменных**

Описание переменных функции addNth представлены на рисунке 7.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Локальные переменные | | |
| temp | LIST | Вспомогательная переменная |
| Формальные переменные | | |
| list | LIST\* | Указатель на адрес первого элемента исходного списка |
| n | int | Номер позиции |

Рис. 7*.*Описание переменных функции addNth

**Описание функции enterField**

Назначение: ввод информационных полей.

Прототип: LIST enterField();

Возвращаемое значение: указатель на элемент исходного списка.

Пример вызова: temp=enterField();

Вызывающая функция: addNth.

Вызываемая функция: enterNum.

**Описание переменных**

Описание переменных функции enterField представлено на рисунке 8.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Локальные переменные | | |
| list | LIST | Вспомогательная переменная |

Рис. 8*.*Описание переменных функции enterField

**Описание функции enterWord**

Назначение: ввод слова, длины 10.

Прототип: char\* enterWord();

Возвращаемое значение: указатель на первый символ строки.

Пример вызова: tempstr=enterWord();

Вызывающая функция: enterField, processingMenu.

**Описание переменных**

Описание переменных функции enterField представлено на рисунке 9.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Локальные переменные | | |
| temp | char\* | Указатель на первый символ строки |
| str | char | Вспомогательная переменная |

Рис. 9*.*Описание переменных функции enterField

**Описание функции deleteMenu**

Назначение: организация управления порядком вызова функций удаления элемента из списка.

Прототип:void deleteMenu(LIST \*list); описание формальных переменных представлено на рис. 10.

Пример вызова: deleteMenu(&list); описание фактических переменных представлено на рис. 10.

Вызывающая функция: main.

Вызываемая функция: del, enterNum, messages, fr, count.

**Описание переменных**

Описание переменных функции deleteMenu представлено на рисунке 10.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Локальные переменные | | |
| Q | int | Переменная выбора пункта меню |
| Формальные переменные | | |
| list | LIST\* | Указатель на первый элемент исходного списка |

Рис. 10*.*Описание переменных функции deleteMenu

**Пункты меню**

1: Удалить первый элемент в списке

2: Удалить последний элемент в списке

3: Удалить элемент по его позиции

4: Очистить список

5: Вернуться в главное меню

**Описание функции del**

Назначение: удаление n-го элемента.

Прототип: void del(LIST \*list, int Key, int n); описание формальных переменных представлено на рисунке 11.

Пример вызова: del(&list, 2, 2); описание фактических переменных представлено в таблице 1.

Вызывающая функция: deleteMenu.

Вызываемая функция: getElem.

**Описание переменных**

Описание переменных функции del представлено на рисунке 11.

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

Рис. 11*.*Описание переменных функции del

**Описание функции fr**

Назначение: освобождение памяти.

Прототип: void fr(LIST \*list); описание формальных переменных представлено на рисунке 12.

Пример вызова: fr(&list); описание фактических переменных представлено в таблице 1.

Вызывающая функция: main, deleteMenu.

**Описание переменных**

Описание переменных функции fr представлено на рисунке 12.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Формальные переменные | | |
| list | LIST\* | Указатель на адрес первого элемента списка |

Рис. 12*.*Описание переменных функции fr

**Описание функции sortMenu**

Назначение: организация меню сортировки.

Прототип: LIST sortMenu(LIST list); описание формальных переменных представлено на рис. 13.

Возвращаемое значение: указатель на первый элемент списка.

Пример вызова: list=sortMenu(list); описание фактических переменных представлено в таблице 1.

Вызывающая функция: main.

Вызываемая функция: sort, messages, enterNum.

**Описание переменных**

Описание переменных функции sortMenu представлены на рисунке 13.

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

Рис. 13*.*Описание переменных функции sortMenu

**Пункты меню**

1: Название

2: Количество струн

3: Год производства

4: Дерево грифа

5: Дерево корпуса

6: Выход в главное меню

**Описание функции sort**

Назначение: сортировка списка.

Прототип:LIST sort(LIST list, int key); описание формальных переменных представлено на рис. 14.

Возвращаемое значение: указатель на первый элемент отсортированного списка.

Пример вызова: temp=sort(list, 1); описание фактических переменных представлено в таблице 1.

Вызывающая функция: sortMenu.

Вызываемая функция: getElem.

**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 |
| 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

**Описание функции processingMenu**

Назначение: организация меню обработки.

Прототип:LIST processingMenu(LIST list); описание формальных переменных представлено на рисунке 15.

Возвращаемое значение: указатель на первый элемент сформированного списка.

Пример вызова: New\_list=processingMenu(list); описание фактических переменных представлено в таблице 1.

Вызываемая функция: search, enterNum, messages.

**Описание переменных**

Описание переменных функции processingMenu 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 processingMenu

**Пункты меню**

1: Название

2: Количество струн

3: Год производства

4: Дерево грифа

5: Дерево корпуса

6: Выход в главное меню

**Описание функции search**

Назначение: поиск в списке.

Прототип: LIST search(LIST list , int Key, int tempint, char\* str); описание формальных переменных представлено на рисунке 16.

Возвращаемое значение: указатель на первый элемент сформированного списка.

Пример вызова: New\_List=search(List, 2, temp, tempstr); описание фактических переменных представлено в таблице 1 и на рисунке 15.

Вызывающая функция: processingMenu.

**Описание переменных**

Описание переменных функции search представлено на рисунке 16.

|  |  |  |
| --- | --- | --- |
| **Имя переменной** | **Тип** | **Назначение** |
| Локальные переменные | | |
| search | LIST | Указатель на первый элемент сформированного списка |
| pFwd | LIST | Указатель на следующий элемент |
| 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) |
| tempint | Int | Вспомогательная переменна |

Рис. 16*.*Описание переменных функции search

**Описание функции count**

Назначение: подсчёт количества элементов.

Прототип: int count(LIST list); описание формальных переменных представлено на рисунке 17.

Возвращаемое значение: количество элементов в списке.

Пример вызова: temp=count(list); описание фактических переменных представлено в таблице 1.

Вызывающая функция: enterMenu, deleteMenu, processingMenu.

**Описание переменных**

Описание переменных функции count представлено на рисунке 17.

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

Рис. 17*.*Описание переменных функции count

**Описание функции getElem**

Назначение: вернуть n-ый элемент.

Прототип:LIST getElem(LIST list, int Key, int n); описание формальных переменных представлено на рисунке 18.

Возвращаемое значение: n-ый элемент списка.

Пример вызова: if(getElem(list, 0,5)); описание фактических переменных представлено в таблице 1.

Вызывающая функция: outputList, sort, del, addNth.

**Описание переменных**

Описание переменных функции count представлено на рисунке 18.

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

Рис. 18*.*Описание переменных функции count

**Структура вызова функций**

Структура вызова функций представлена на рисунке 19.

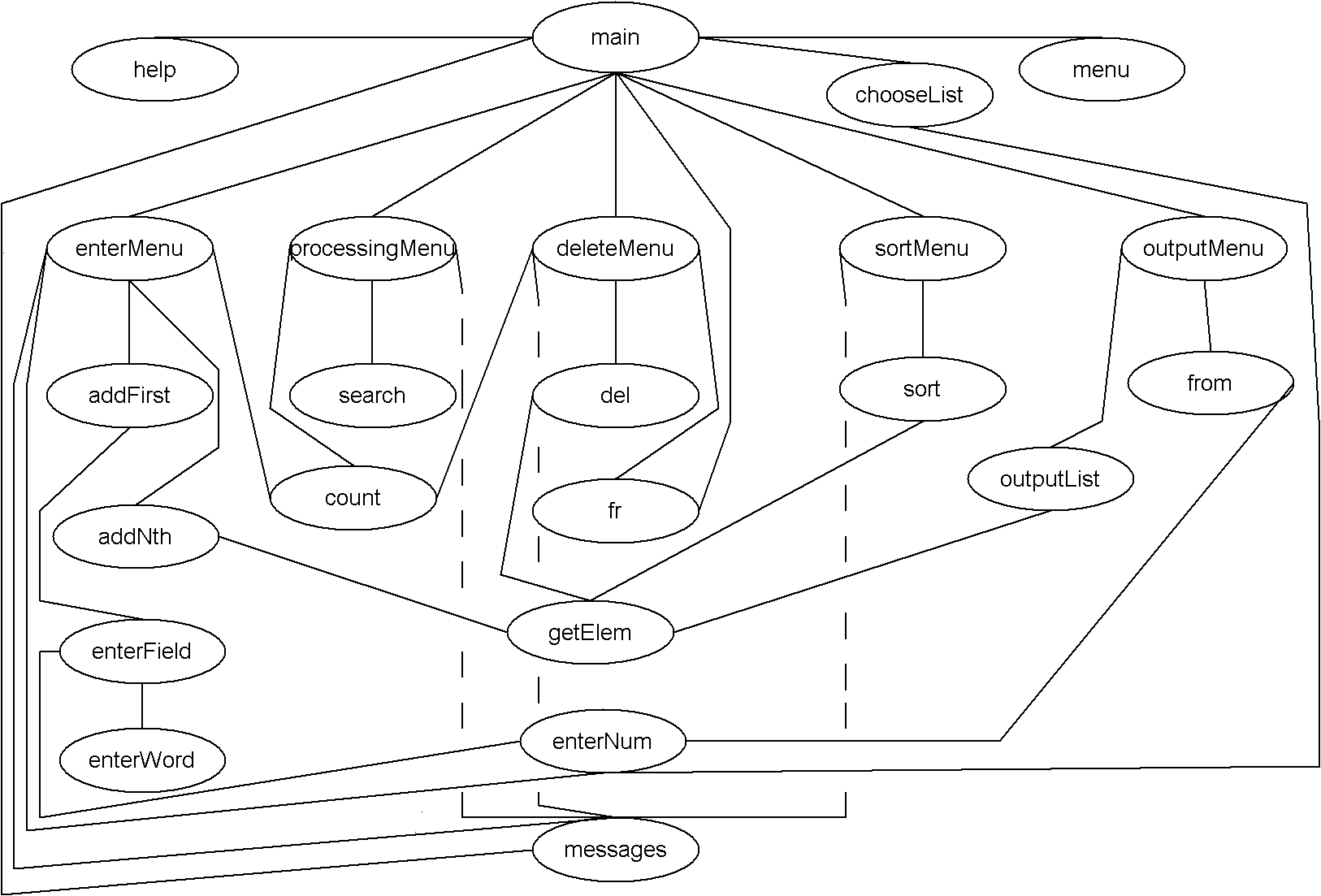
****

Рис. 19. Структура вызова функций

**Текст программы с комментариями**

#include <stdlib.h>

#include <locale.h>

#include <windows.h>

#include <conio.h>

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

void help(); //Прототип функции справка

void menu(); //Прототип функции Главного меню

void messages(int Key); //Прототип функции вывода сообщения

int chooseList(); //Прототип функции выбора списка

int from(); //Прототип функции список с конца или с начала

void enterMenu(LIST\* list); //Прототип функции подменю ввода элементов

int enterNum(int first, int last); //Прототип функции ввода целочисленных значений в

//диапазоне

void outputMenu(LIST list); //Прототип функции подменю вывода

void outputList(LIST list, int Key); //Прототип функции вывода списка

void addFirst(LIST \*list); //Прототип функции добавления 1-го элемента

void addNth(LIST list, int n); //Прототип функции добавления n-го элемента

LIST enterField(); //Прототип функции ввода инф. полей

char\* enterWord(); //Прототип функции ввода слова

void deleteMenu(LIST\* list); //Прототип функции подменю удаления элементов

void del(LIST \*list, int Key, int n); //Прототип функции удаления n-го элемента

void fr(LIST \*list); //Прототип функции очистка списка

LIST sortMenu(LIST list); //Прототип функции подменю сортировки

LIST sort(LIST list, int Key); //Прототип функции сортировки

LIST processingMenu(LIST list); //Прототип функции подменю поиска

LIST search(LIST list, int Key, int num, char\* str); //Прототип функции поиска

LIST getElem(LIST list, int Key, int n); //Прототип функции нахождения элементов

int count(LIST list); //Прототип функции подсчёта количества элементов

int main()

{

system("mode con cols=80 lines=20");

LIST list=NULL, New\_list=NULL;

int Q;

setlocale(LC\_ALL, "RUS");

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

do

{

menu();

switch (Q=enterNum(1, 7))

{

case 1:

help();

break;

case 2:

enterMenu(&list);

break;

case 3:

deleteMenu(&list);

break;

case 4:

if(chooseList())

outputMenu(New\_list);

else

outputMenu(list);

break;

case 5:

fr(&New\_list);

New\_list=processingMenu(list);

break;

case 6:

if(chooseList())

New\_list=sortMenu(New\_list);

else

list=sortMenu(list);

break;

case 7:

messages(8);

break;

}

}

while(Q!=7);

fr(&list);

fr(&New\_list);

return 0;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция справка

void help()

{

system("cls");

puts("\n\n Данная программа предназначена для создания двусвязного списка с возможностью выборки.");

puts(" Выборка составляется из гитар до выбранного года производства или по количеству струн.");

puts(" Если возникли проблемы обращайтесь, пожалуйста, на электронную почту:");

puts(" komdosh@gelezo2.ru\n");

system("pause");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция меню

void menu()

{

system("cls");

puts("Главное меню");

puts("1 - Справка");

puts("2 - Добавить элемент в список");

puts("3 - Удалить элемент из списка");

puts("4 - Вывод списка");

puts("5 - Поиск");

puts("6 - Сортировка");

puts("7 - Выход");

printf("Введите номер пункта - ");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция выбора списка

int chooseList()

{

system("cls");

puts("Для какого списка выполнить это действие?");

puts("1 - Исходный список");

puts("2 - Сформированный список");

printf("Введите номер пункта (от %d до %d): ", 1, 2);

return (enterNum(1, 2)-1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция "список с конца или с начала"

int from()

{

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);

return (enterNum(1, 2)-1);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция подменю добавления элементов

void enterMenu(LIST\* list)

{

int Q;

do

{

system("cls");

puts("Меню добавления элементов");

puts("1 - Добавить элемент в начало списка");

puts("2 - Добавить элемент в конец списка");

puts("3 - Добавить элемент на выбранную позицию");

puts("4 - Вернуться в главное меню");

printf("Введите номер пункта - ");

switch(Q=enterNum(1,4))

{

case 1:

addFirst(list);

messages(11);

break;

case 2:

if(\*list)

{

addNth(\*list, count(\*list));

messages(11);

}

else

messages(3);

break;

case 3:

if(\*list)

if(count(\*list)>1)

{

printf("Введите номер позиции, куда вставить элемент (от %d до %d): ", 2,

(count(\*list)>2)?count(\*list)-1:count(\*list));

addNth(\*list, enterNum(2, count(\*list))-1);

messages(11);

}

else

messages(12);

else

messages(3);

break;

}

}

while(Q!=4);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция добавления 1-го элемента

void addFirst(LIST \*list)

{

LIST temp = enterField();

if(\*list)

(\*list)->prev=temp;

temp->prev = NULL;

temp->next = \*list;

\*list=temp;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция добавления n-го или последнего элемента

void addNth(LIST list, int n)

{

LIST temp = enterField();

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;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция ввода данных в поля

LIST enterField()

{

system("cls");

LIST list=(LIST)malloc(sizeof(sLIST));

printf("Введите марку гитары (кол-во символов от 1 до 10): ");

list->Guitars.Name=enterWord();

printf("Введите количество струн (от %d до %d): ", 1, 20);

list->Guitars.Strings=enterNum(1, 20);

printf("Введите год производства (от %d до %d): ", 1899, 2015);

list->Guitars.Year=enterNum(1899, 2015);

printf("Введите название дерева грифа (кол-во символов от 1 до 10): ");

list->Guitars.Wood.Neck=enterWord();

printf("Введите название дерева корпуса (кол-во символов от 1 до 10): ");

list->Guitars.Wood.Deck=enterWord();

return list;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция ввода слова

char\* enterWord()

{

char str[10], \*temp;

do

{

gets(str);

fflush(stdin);

if(strlen(str)<1 || strlen(str)>10)

printf("Возможно вы ошиблись при вводе?\n(кол-во символов от 1 до 10)\nПовторите ввод: ");

}

while(strlen(str)<1 || strlen(str)>10);

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[4];

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

void deleteMenu(LIST\* list)

{

if(!\*list)

{

messages(9);

return;

}

int Q;

do

{

system("cls");

puts("Меню удаления элементов");

puts("1 - Удалить первый элемент в списке");

puts("2 - Удалить последний элемент в списке");

puts("3 - Удалить элемент по его позиции");

puts("4 - Очистить список");

puts("5 - Вернутся в главное меню");

printf("Введите номер пункта - ");

switch(Q=enterNum(1, 5))

{

case 1:

del(list, 0, 0);

messages(10);

break;

case 2:

del(list, 1, 0);

messages(10);

break;

case 3:

if(count(\*list)>1)

{

printf("Введите номер позиции элемента, который следует удалить (от %d до %d): ", 2,

(count(\*list)>2)?count(\*list)-1:count(\*list));

del(list, 2, enterNum(2, count(\*list)));

messages(10);

}

else

messages(13);

break;

case 4:

fr(list);

messages(3);

break;

}

}

while(Q!=5);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция удаления n-го элемента в списке

void del(LIST \*list, int Key, int n) //Key: 0 - начало, 1 - конец, 2 - н-ый

{

LIST temp;

switch(Key)

{

case 0:

if((\*list)->next)

{

\*list=(\*list)->next;

free((\*list)->prev);

}

else

{

free(\*list);

\*list=NULL;

}

break;

case 1:

temp=getElem(\*list, 1, 0);

if(!temp->prev)

\*list = NULL;

else

temp->prev->next = NULL;

free(temp);

break;

case 2:

temp = getElem(\*list, 2, n);

temp->prev->next = temp->next;

temp->next->prev = temp->prev;

free(temp);

break;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция освобождения памяти

void fr(LIST \*list)

{

if(\*list)

{

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;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция подменю сортировки

LIST sortMenu(LIST\* list)

{

if(!list)

{

messages(9);

return NULL;

}

LIST Sort;

system("cls");

puts("Меню сортировки");

puts("По какому пункту сделать сортировку?");

puts("1 - Название");

puts("2 - Количество струн");

puts("3 - Год производства");

puts("4 - Дерево грифа");

puts("5 - Дерево корпуса");

puts("6 - Выход в главное меню");

printf("Введите номер пункта - ");

int Q=enterNum(1,6);

if(Q>0&&Q<6)

{

Sort=sort(list, Q);

messages(4);

}

return Sort;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//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 && (pFwd->Guitars.Name[0] > temp->Guitars.Name[0] && Key==1 || temp->

Guitars.Strings > pFwd->Guitars.Strings && Key==2 || temp->Guitars.Year > pFwd->Guitars.Year && Key==3 ||

pFwd->Guitars.Wood.Neck[0] > temp->Guitars.Wood.Neck[0] && Key==4 || pFwd->Guitars.Wood.Deck[0] >

temp->Guitars.Wood.Deck[0] && Key==5); 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 processingMenu(LIST list)

{

if(!list)

{

messages(9);

return NULL;

}

LIST New\_list=NULL;

const char\* str;

char\* tempstr;

int Q, temp, first, last;

system("cls");

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");

printf("Введите номер пункта - ");

switch(Q=enterNum(1,6))

{

case 1:

first=1899;

last=2015;

str="до какого года производства";

break;

case 2:

first=1;

last=20;

str="до какого количества струн";

break;

case 3:

str="названию";

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);

}

if(Q==1||Q==4||Q==5)

{

printf("Введите по какому %s выводить результаты - ", str);

tempstr=enterWord();

}

if(Q>0&&Q<6)

New\_list=search(list, Q, temp, tempstr);

if(count(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, pFwd;

for(; list; list=list->next)

if(!strcmp(list->Guitars.Name, str) && Key==1 || list->Guitars.Strings<=num && Key==2 ||

list->Guitars.Year<=num && Key==3 || !strcmp(list->Guitars.Wood.Neck, str) &&

Key==4 || !strcmp(list->Guitars.Wood.Deck, str) && Key==5)

{

pFwd=search;

search=(LIST)malloc(sizeof(sLIST));

search->Guitars=list->Guitars;

search->next=pFwd;

search->prev=NULL;

if(pFwd)

pFwd->prev=search;

}

return search;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция подменю вывода

void outputMenu(LIST list)

{

if(!list)

{

messages(9);

return;

}

outputList(list, from());

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция вывода данных

void outputList(LIST list, int Key)

{

if(!list)

{

messages(9);

return;

}

system("mode con cols=80 lines=47");

if(Key)

list=getElem(list, 1, 0);

system("cls");

printf("================================================================================");

printf("%12s | %18s | %14s | %17s\n", " ", " ", " ", "Дерево:");

printf("%12s | %18s | %14s | %s\n", "Название", "Год производства", "Кол-во струн","\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

printf("%12s | %18s | %14s | %11s | %6s\n", " ", " ", " ", "Корпус", "Гриф" );

printf("================================================================================");

while(list)

{

printf("%12s | %18d | %14d | %11s | %6s ",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;

}

system("mode con cols=80 lines=20");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//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 кол-ва элементов

int count(LIST list)

{

int Count;

for(Count=0; list; list=list->next, Count++);

return Count;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

//Функция вывода сообщений пользователю

void messages(int Key)

{

system("cls");

switch(Key)

{

case 1:

puts("Сначала необходимо ввести данные");

break;

case 2:

puts("Вы ввели данные, но не создали выборку");

puts("Вам необходимо выбрать 5 пункт меню для поиска данных");

break;

case 3:

puts("Список удалён");

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 списка' -> 'Сформированный список'");

break;

case 6:

puts("Выборка из данных не была сформирована");

puts("В исходных данных не нашлось таких результатов");

break;

case 7:

puts("Что-то пошло не так, введите пункт меню повторно");

break;

case 8:

puts("До новых встреч!");

break;

case 9:

puts("Список пуст");

puts("Для выполнения этого действия, создайте список");

break;

case 10:

puts("Элемент успешно удалён");

break;

case 11:

puts("Элемент успешно добавлен");

puts("Для просмотра выберите пункт 'Вывод списка' -> 'Исходный список'");

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;

}

system("pause");

}

**Результаты решения задачи**

При выполнении программы были получены результаты, совпадающие со значениями, приведенными на рисунке 1. Ошибок не обнаружено.

**Вывод**

При выполнении лабораторной работы были получены практические навыки работы c двусвязными списками на языке программирования «С/C++».