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 № 2
на тему:
“Односвязные списки”

по дисциплине “Программирование. Дополнительные главы”

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

Санкт-Петербург
2015 г.

Цель

Получить практические навыки работы с односвязными списками.

Задание

Написать программу для: создания списка, контрольного вывода, удаления элементов списка, поиска и формирования нового списка, сортировки и вывод 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 значению года производства или количеству струн, есть возможность сортировки по количеству струн или году производства (по возрастанию), вывод результата.

Описание структуры

Для решения задач разработаны структуры:

```
typedef struct stWood
    {
       char Deck[10];       //дерево корпуса
       char Neck[10];       //дерево грифа
    } WOOD;

typedef struct stGuitars
   {
     char Name[10];         //название
       int Strings;         //количество струн
       int Year;            //год производства
       WOOD Wood;           //название дерева
   } GUITARS;

typedef struct stLIST*
   {
       GUITARS Guitars;  //информационные поля
       struct stLIST* next; //следующий элемент
   } LIST;
```

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 переменных представлено в Таблице 1.

Таблица 1. Описание переменных главной функции

Имя переменной	Тип	Назначение
Guitars_list	LIST*	Указатель на первый элемент исходного списка
Count	int	Количество гитар
New_Guitars_list	LIST*	Указатель на первый элемент сформированного списка
NewCount	int	Количество гитар в сформированной выборке
Process	bool	Флаг поиска данных(true – был поиск, false – не было поиска)
Key	int	Переменная выбора списка(0 – исходный, 1 - сформированный)
Q	int	Переменная выбора меню

Схема алгоритма главной функции

Схема главной алгоритма функции представлена на рисунке 2.

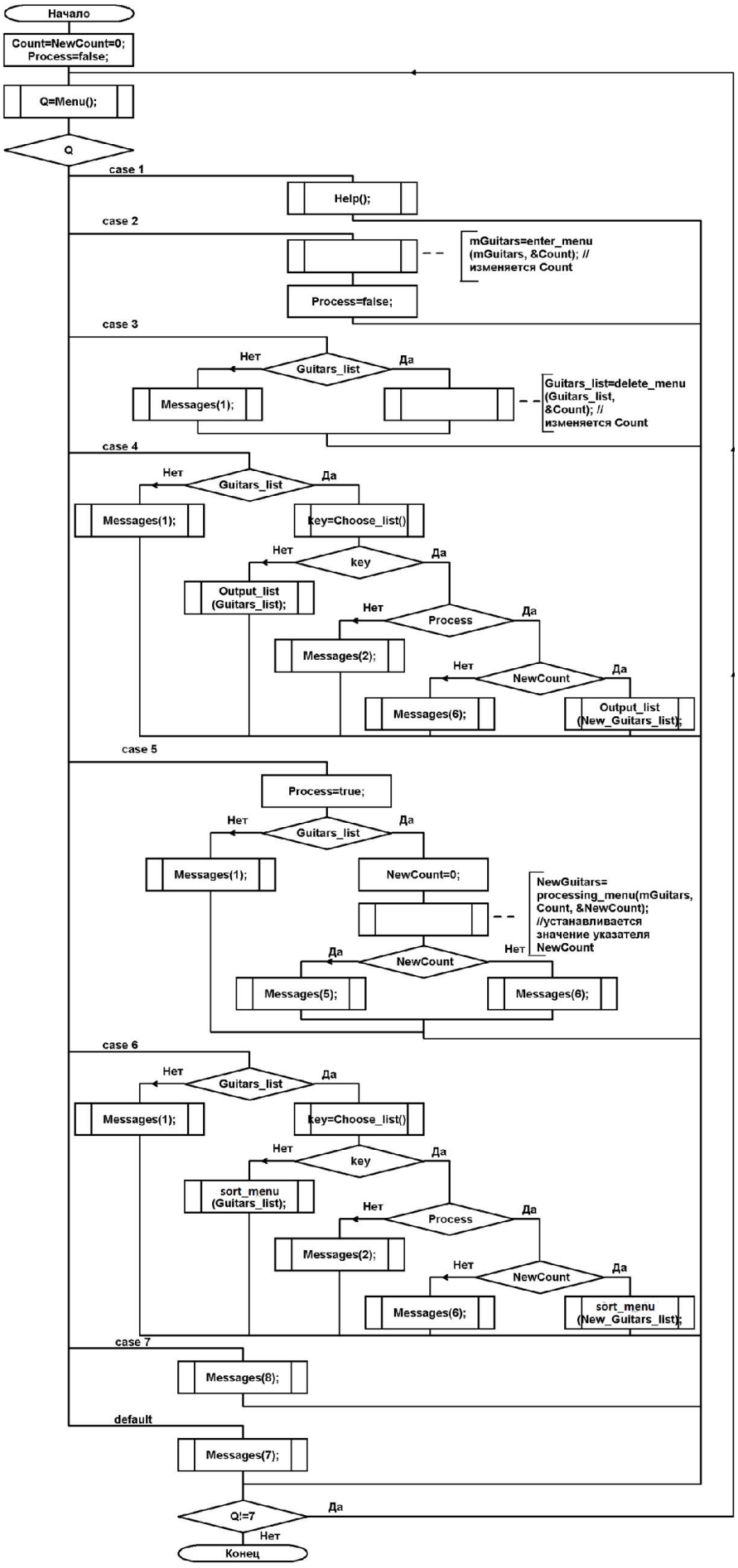

Рис. 2. Схема алгоритма главной функции.

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: int Menu();
Возвращаемое значение: номер пункта меню.
Пример вызова: Q=Menu();
Вызывающая функция: main.

Описание переменных

Описание переменных функции Menu представлено на рисунке 3.

Имя переменной	Тип	Назначение
Локальные переменные		
Q	int	Переменная выбора пункта меню

Рис. 3. Описание переменных функции Menu

Описание функции Messages

Назначение: Функция используется для ввода сообщений пользователю.
Прототип: void Messages(int); описание формальных переменных представлено на рисунке 4.
Возвращаемое значение: int номер вызываемого сообщения.
Пример вызова: Messages(1); описание фактических переменных представлено в таблице 1.
Вызывающая функция: main, enter_menu, delete_menu, sort_menu, processing_menu.
Сообщения:

Messages(1): "Сначала необходимо ввести данные"
Messages(2): "Вы ввели данные, но не обработали их"
" Вам необходимо выбрать 5 пункт меню для поиска данных"
Messages(3): " Для добавления последующих элементов, сначала необходимо добавить первый "
Messages(4): "Сортировка успешно завершена "
Messages(5): "Выборка из данных успешно сформирована"
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"
Messages(10): "Элемент удалён"
Messages(11): "Элемент успешно добавлен"
Messages(12): "Поиск успешно завершён"
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 4.

Имя переменной	Тип	Назначение
Формальные переменные		
Key	int	Вспомогательная переменная

Рис. 4. Описание переменных функции Messages

Описание функции enter_menu

Назначение: организация управления порядком вызова функций добавления элемента в список.
Прототип: LIST* enter_menu(LIST* Guitars_list, int* Count); описание формальных переменных представлено на рисунке 5.
Возвращаемое значение: указатель на первый элемент исходного списка.
Пример вызова: Guitars_ list=enter_menu(Guitars_ list, &Count); описание фактических переменных представлено в таблице 1.
Вызывающая функция: main.
Вызываемая функция: Add, enter_num, Messages.

Описание переменных

Описание переменных функции enter_menu представлено на рисунке 5.

Имя переменной	Тип	Назначение
Локальные переменные		
Q	int	Переменная выбора пункта меню
temp	int	Вспомогательная переменная
Формальные переменные		
Guitars_LIST*	LIST*	Указатель на первый элемент исходного списка
Count	int*	Указатель на кол-во гитар

Рис. 5. Описание переменных функции enter_menu

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 Choose_list

Назначение: выбор списка.
Прототип: int Choose_list();
Возвращаемое значение: ключ(0 – исходный список, 1 – сформированный список).
Пример вызова: key=Choose_list();
Вызывающая функция: main.

Описание переменных

Описание переменных функции Choose_list представлено на рисунке 6.

Имя переменной	Тип	Назначение
Key	int	Вспомогательная переменная

Рис. 6. Описание переменных функции Choose_list

Описание функции enter_num

Назначение: ввод чисел в заданном диапазоне.
Прототип: int enter_num(int first, int last); описание формальных переменных представлено на рисунке 7.
Возвращаемое значение: целое число.
Пример вызова: Guitars_list->Guitars.Strings=enter_num(1, 20); описание фактических переменных представлено на рисунке 6.
Вызывающая функция: Choose_list, enter_menu, enter_field, delete_menu, processing_menu.

Описание переменных

Описание переменных функции enter_num представлено на рисунке 7.

Имя переменной	Тип	Назначение
Локальные переменные		
num	int	Вспомогательная переменная
Формальные переменные		
first	int	Начальное число
last	int	Конечное число

Рис. 7. Описание переменных функции enter_num

Описание функции Output_list

Назначение: вывод списка.
Прототип: void Output_list (LIST* Guitars_list); описание формальных переменных представлено на рисунке 8.
Пример вызова: Output_list (Guitars_list); описание фактических переменных представлено в таблице 1.
Вызывающая функция: main.

Описание переменных

Описание переменных функции Output _list представлено на рисунке 8.

Имя переменной	Тип	Назначение
Формальные переменные		
Guitars_list	LIST*	Указатель на первый элемент исходного списка

Рис. 8. Описание переменных функции Output_list

Описание функции Add_First

Назначение: добавление 1-го элемента.
Прототип: void Add_First(LIST **Guitars_list); описание формальных переменных представлено на рисунке 9.
Пример вызова: Add(&Guitars_list); описание фактических переменных представлено на рисунке 5.
Вызывающая функция: enter_menu.
Вызываемая функция: enter_field.

Описание переменных

Описание переменных функции Add_First представлены на рисунке 9.

Имя переменной	Тип	Назначение
Локальные переменные		
temp	LIST*	Вспомогательная переменная
Формальные переменные		
Guitars_list	LIST**	Указатель на адрес первого элемента исходного списка

Рис. 9. Описание переменных функции Add_First

Описание функции Add

Назначение: добавление n-го элемента.
Прототип: void Add(LIST *Guitars_list, int n); описание формальных переменных представлено на рисунке 10.
Пример вызова: Add(Guitars_list, *Count); описание фактических переменных представлено на рисунке 5.
Вызывающая функция: enter_menu.
Вызываемая функция: enter_field.

Описание переменных

Описание переменных функции Add представлены на рисунке 10.

Имя переменной	Тип	Назначение
Локальные переменные		
temp	LIST*	Вспомогательная переменная
i	int	Счётчик
Формальные переменные		
Guitars_list	LIST*	Указатель на адрес первого элемента исходного списка
n	int	Номер позиции

Рис. 10. Описание переменных функции Add

Описание функции enter_field

Назначение: ввод информационных полей.
Прототип: LIST* enter_field();
Возвращаемое значение: указатель на элемент исходного списка.
Пример вызова: temp=enter_field();
Вызывающая функция: Add.
Вызываемая функция: enter_num.

Описание переменных

Описание переменных функции enter_field представлено на рисунке 11.

Имя переменной	Тип	Назначение
Локальные переменные		
Guitars_list	LIST*	Вспомогательная переменная

Рис. 11. Описание переменных функции enter_field

Описание функции delete_menu

Назначение: организация управления порядком вызова функций удаления элемента из списка.
Прототип: LIST* delete_menu(LIST* Guitars_list, int* Count); описание формальных переменных представлено на рисунке 12.
Возвращаемое значение: указатель на первый элемент списка.
Пример вызова: Guitars_list=delete_menu(Guitars_list, &Count); описание фактических переменных представлено в таблице 1.
Вызывающая функция: main.
Вызываемая функция: Del,enter_num, Messages, fr.

Описание переменных

Описание переменных функции delete_menu представлено на рисунке 12.

Имя переменной	Тип	Назначение
Локальные переменные		
Q	int	Переменная выбора пункта меню
temp	int	Вспомогательная переменная
Формальные переменные		
Guitars_list	LIST*	Указатель на первый элемент исходного списка
Count	int*	Указатель на кол-во гитар

Рис. 12. Описание переменных функции delete_menu

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 **Guitars_list, int n, int Key); описание формальных переменных представлено на рисунке 13 .
Пример вызова: Del(&Guitars_list, *Count, 1); описание фактических переменных представлено на рисунке 12.
Вызывающая функция: delete_menu.

Описание переменных

Описание переменных функции Del представлено на рисунке 13.

Имя переменной	Тип	Назначение
Локальные переменные		
pBwd	LIST*	Указатель на предыдущий элемент
pFwd	LIST*	Указатель на следующий элемент
i	Int	Счётчик
Формальные переменные		
Guitars_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. 13. Описание переменных функции Del

Описание функции fr

Назначение: очистка списка.
Прототип: void fr(LIST **Guitars_list); описание формальных переменных представлено на рисунке 14.
Пример вызова: fr(&Guitars_list); описание фактических переменных представлено на рисунке 12.
Вызывающая функция: main, delete_menu.

Описание переменных

Описание переменных функции fr представлено на рисунке 14.

Имя переменной	Тип	Назначение
Локальные переменные		
pBwd	LIST*	Указатель на предыдущий элемент
Формальные переменные		
Guitars_list	LIST*	Указатель на адрес первого элемента списка

Рис. 14. Описание переменных функции fr

Описание функции sort_menu

Назначение: организация меню сортировки.
Прототип: LIST* sort_menu(LIST* Guitars_list); описание формальных переменных представлено на рисунке 15.
Возвращаемое значение: указатель на первый элемент списка.
Пример вызова: Guitars_list=sort_menu(Guitars_list); описание фактических переменных представлено в таблице 1.
Вызывающая функция: main.
Вызываемая функция: Sort, Messages.

Описание переменных

Описание переменных функции sort _menu представлены на рисунке 15.

Имя переменной	Тип	Назначение
Локальные переменные		
temp	LIST*	Вспомогательная переменная
Q	int	Переменная выбора пункта подменю
Формальные переменные		
Guitars_list	LIST*	Указатель на первый элемент исходного списка

Рис. 15. Описание переменных функции sort _menu

Пункты меню

1: Год производства
2: Количество струн
3: Выход в главное меню

Описание функции Sort

Назначение: сортировка списка.
Прототип: LIST* Sort(LIST *Guitars_list, int key); описание формальных переменных представлено на рисунке 16.
Возвращаемое значение: указатель на первый элемент списка.
Пример вызова: temp=Sort(Guitars_list, 1); описание фактических переменных представлено на рисунке 15.
Вызывающая функция: sort_menu.

Описание переменных

Описание переменных функции Sort представлено на рисунке 16.

Имя переменной	Тип	Назначение
Локальные переменные		
temp	LIST*	Вспомогательная переменная
pFwd	LIST*	Указатель на следующий элемент
pBwd	LIST*	Указатель на предыдущий элемент
Sort	LIST*	Указатель на первый элемент списка
Формальные переменные		
Guitars_list	LIST*	Указатель на первый элемент списка
Key	int	Ключ (1 – по году производства, 2 – по количеству струн)

Рис. 16. Описание переменных функции Sort

Описание функции processing_menu

Назначение: организация меню обработки.
Прототип: LIST* processing_menu(LIST *Guitars_list, int *NewCount); описание формальных переменных представлено на рисунке 17.
Возвращаемое значение: указатель на первый элемент сформированного списка.
Пример вызова: New_Guitars_list=processing_menu(Guitars_list, &NewCount); описание фактических переменных представлено в таблице 1.
Вызывающая функция:main.
Вызываемая функция: processing, enter_num, Messages.

Описание переменных

Описание переменных функции processing_menu представлено на рисунке 17.

Имя переменной	Тип	Назначение
Локальные переменные		
New_Guitars_list	LIST*	Указатель на первый элемент сформированного списка
temp	int	Вспомогательная переменная
Q	int	Переменная выбора пункта подменю
Формальные переменные		
Guitars_list	LIST*	Указатель на первый элемент исходного списка
NewCount	int*	Указатель на кол-во гитар выборки

Рис. 17. Описание переменных функции processing_menu

Пункты меню

1: Год производства
2: Количество струн
3: Выход в главное меню

Описание функции Search

Назначение: поиск в списке.
Прототип: LIST* Search(LIST *Guitars_list,int *NewCount, int key, int tempint); описание формальных переменных представлено на рисунке 18.
Возвращаемое значение: указатель на первый элемент сформированного списка.
Пример вызова: New_Guitars_list=Search(Guitars_list, NewCount, 2, temp); описание фактических переменных представлено на рисунке 17.
Вызывающая функция: processing_menu.

Описание переменных

Описание переменных функции Search представлено на рисунке 18.

Имя переменной	Тип	Назначение
Локальные переменные		
Search	LIST*	Указатель на первый элемент сформированного списка
pFwd	LIST*	Указатель на следующий элемент
Формальные переменные		
Guitars_list	LIST*	Указатель на первый элемент списка
NewCount	int*	Указатель на количество гитар в сформированном списке
Key	Int	Ключ (1 – по году производства, 2 – по количеству струн)
tempint	Int	Вспомогательная переменна

Рис. 18. Описание переменных функции Search

Структура вызова функций

Структура вызова функций представлена на рисунке 19.

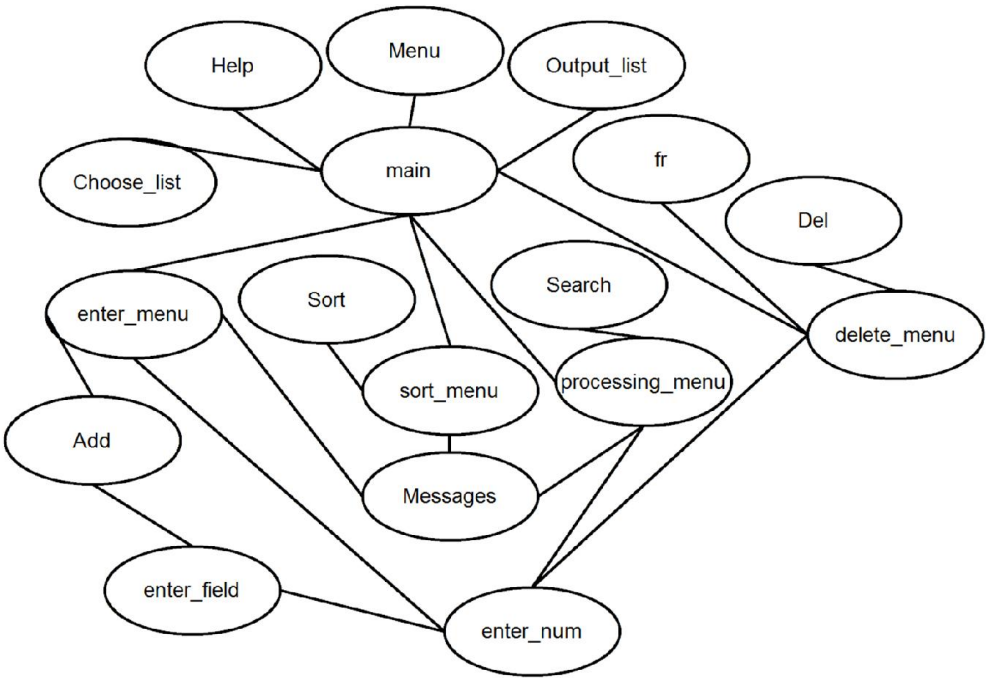

Рис. 19. Структура вызова функций

Текст программы с комментариями

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

```
typedef struct stWood
  {
    char Deck[10];                                                  //Дерево корпуса
    char Neck[10];                                                  //Дерево грифа
  } WOOD;

typedef struct stGuitars
  {
    char Name[10];                                                  //Название гитары
    int Strings;                                                    //Кол-во струн
    int Year;                                                       //Год производства
    WOOD Wood;                                                      //Дерево
  } GUITARS;

typedef struct stList
  {
    GUITARS Guitars;                                                //Структура
    struct stList *next;                                            //След. элемент
  } LIST;


void Help();                                                         //Прототип функции справка
int Menu();                                                          //Прототип функции Главного меню
int Choose_list();                                                   //Прототип функции выбора списка
LIST* enter_menu(LIST* Guitars_list, int* Count);                    //Прототип функции подменю ввода элементов
int enter_num(int first, int last);                                  //Прототип функции ввода целочисленных
значений в диапазоне
void Output_list(LIST *Guitars_list);                                //Прототип функции вывода списка
void Add_First(LIST **Guitars_list);                                 //Прототип функции добавления 1-го элемента
void Add(LIST *Guitars_list, int n);                                 //Прототип функции добавления n-го элемента
LIST* enter_field();                                                 //Прототип функции ввода инф. полей
LIST* delete_menu(LIST* Guitars_list, int* Count);                   //Прототип функции подменю удаления элементов
void Del(LIST **Guitars_list, int n, int Key);                       //Прототип функции удаления n-го элемента
void fr(LIST **Guitars_list);                                        //Прототип функции очистка списка
LIST* sort_menu(LIST* Guitars_list);                                 //Прототип функции подменю сортировки
LIST* Sort(LIST *Guitars_list, int key);                             //Прототип функции сортировки
LIST* processing_menu(LIST *Guitars_list, int *NewCount);            //Прототип функции подменю поиска
LIST* Search(LIST *Guitars_list,int *NewCount, int key, int tempint); //Прототип функции поиска
void Messages(int Key);                                              //Прототип функции вывода сообщения

int main()
  {
   system("mode con cols=80 lines=40");
   LIST *Guitars_list=NULL, *New_Guitars_list=NULL;
   int Q, NewCount=0, Count=0, key;
   bool Process=false; //process - был ли поиск
   setlocale(LC_ALL, "RUS");
   SetConsoleCP(1251);
   SetConsoleOutputCP(1251);
   do
    {
     Q = Menu();
     switch (Q)
      {
       case 1:
              Help();
              break;
       case 2:
              Guitars_list=enter_menu(Guitars_list, &Count);
              Process=false;
              break;
       case 3:
              if(Guitars_list)
                Guitars_list=delete_menu(Guitars_list, &Count);
              else
                Messages(9);
              break;
       case 4:
              if(Guitars_list)
                {
                 key=Choose_list();
                 if(key)
                   if(Process)
                     if(NewCount)
                       Output_list(New_Guitars_list);
                     else
                       Messages(6);
                   else
                     Messages(2);
                 else
                   Output_list(Guitars_list);
```

```
                }
              else
                Messages(1);
              break;
       case 5:
              Process=true;
              if(Guitars_list)
                {
                 NewCount=0;
                 fr(&New_Guitars_list);
                 New_Guitars_list=processing_menu(Guitars_list, &NewCount);
                 if(NewCount)
                   Messages(5);
                 else
                   Messages(6);
                }
              else
                Messages(1);
              break;
       case 6:
              if(Guitars_list)
                {
                 key=Choose_list();
                 if(key)
                   if(Process)
                     if(NewCount)
                       New_Guitars_list=sort_menu(New_Guitars_list);
                     else
                       Messages(6);
                   else
                     Messages(2);
                 else
                   Guitars_list=sort_menu(Guitars_list);
                }
              else
                Messages(1);
              break;
       case 7:
              Messages(8);
              break;
       default:
               Messages(7);
      }
     }
   while (Q!=7);
   fr(&Guitars_list);
   fr(&New_Guitars_list);
   return 0;
  }
//*****************************************************************************************
//Функция справка
void Help()
  {
   system("cls");
   puts("\n\n  Данная программа предназначена для создания односвязного списка с возможностью выборки.");
   puts("  Выборка составляется из гитар до выбранного года производства или по кол-ву\n струн.");
   puts("  Если возникли проблемы обращайтесь, пожалуйста, на электронную почту:");
   puts(" komdosh@gelezo2.ru\n");
   system("pause");
  }
//*****************************************************************************************
//Функция меню
int Menu()
  {
   int Q;
   system("cls");
   puts("Главное меню");
   puts("1 - Справка");
   puts("2 - Добавить элемент в список");
   puts("3 - Удалить элемент из списка");
   puts("4 - Вывод списков");
   puts("5 - Поиск");
   puts("6 - Сортировка");
   puts("7 - Выход");
   printf("Введите номер пункта - ");
   scanf("%d", &Q);
   printf("\n");
   fflush(stdin);
   return Q;
  }
//*****************************************************************************************
//Функция выбора списка
```

```
int Choose_list()
  {
   int key;
   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);
   key=enter_num(1, 2);
   return (key-1);
  }
//*****************************************************************************************
//Функция подменю добавления элементов
LIST* enter_menu(LIST* Guitars_list, int* Count)
  {
   int Q, temp;
   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 - ");
     scanf("%d", &Q);
     fflush(stdin);
     switch(Q)
      {
       case 1:
              Add_First(&Guitars_list);
              (*Count)++;
              Messages(11);
              break;
       case 2:
              if(*Count)
                {
                 Add(Guitars_list, 5678);

                 (*Count)++;
                 Messages(11);
                }
              else
                Messages(3);
              break;
       case 3:
              if(*Count)
                {
                 if((*Count)>1)
                   {
                    printf("Введите номер позиции, куда вставить элемент (от %d до %d): ", 2, *Count);
                    temp=enter_num(2, *Count);
                    Add(Guitars_list, (temp-1));
                    (*Count)++;
                    Messages(11);

                   }
                 else
                   Messages(13);
                }
              else
                Messages(3);
              break;
       default:
               if(Q!=4)
                 Messages(7);
      }
    }
   while(Q!=4);
   return Guitars_list;
  }
//*****************************************************************************************
//Функция добавления 1-го элемента
void Add_First(LIST **Guitars_list)
  {
    LIST *temp = enter_field();
    temp->next = (*Guitars_list);
    (*Guitars_list)=temp;
  }
//*****************************************************************************************
//Функция добавления n-го элемента
void Add(LIST *Guitars_list, int n)
  {
```

```
    LIST *temp = enter_field();
    for(int i = 1; i < n && Guitars_list->next; i++)
      Guitars_list = Guitars_list->next;
    if(Guitars_list->next)
      temp->next = Guitars_list->next;
    else
      temp->next = NULL;
    Guitars_list->next = temp;
  }
//*****************************************************************************************
//Функция ввода данных в поля
LIST* enter_field()
  {
   system("cls");
   LIST* Guitars_list=(LIST*)malloc(sizeof(LIST));
   printf("Введите марку гитары (кол-во символов от 1 до 10): ");
   do
    {
     //gets(Guitars_list->Guitars.Name);
     Guitars_list->Guitars.Name[0]='w';
     fflush(stdin);
     if(strlen(Guitars_list->Guitars.Name)<1 || strlen(Guitars_list->Guitars.Name)>10)
       printf("Возможно вы ошиблись при вводе?\n(кол-во символов не больше 10)\nПовторите ввод: ");
    }

   while(strlen(Guitars_list->Guitars.Name)<1 || strlen(Guitars_list->Guitars.Name)>10);
   printf("Введите количество струн (от %d до %d): ", 1, 20);
   Guitars_list->Guitars.Strings=enter_num( 1, 20);
   //Guitars_list->Guitars.Strings=1;
   printf("Введите год производства (от %d до %d): ", 1899, 2015);
   //Guitars_list->Guitars.Year=enter_num( 1899, 2015);
   Guitars_list->Guitars.Year=200;
   printf("Введите название дерева грифа (кол-во символов от 1 до 10): ");
   do
    {
     //gets(Guitars_list->Guitars.Wood.Neck);
     Guitars_list->Guitars.Wood.Neck[0]='w';
     fflush(stdin);
     if(strlen(Guitars_list->Guitars.Wood.Neck)<1 || strlen(Guitars_list->Guitars.Wood.Neck)>10)
       printf("Возможно вы ошиблись при вводе?\n(кол-во символов от 1 до 10)\nПовторите ввод: ");
    }

   while(strlen(Guitars_list->Guitars.Wood.Neck)<1 || strlen(Guitars_list->Guitars.Wood.Neck)>10);
   printf("Введите название дерева корпуса (кол-во символов от 1 до 10): ");
   do
    {
     //gets(Guitars_list->Guitars.Wood.Deck);
     Guitars_list->Guitars.Wood.Deck[0]='q';
     fflush(stdin);
     if(strlen(Guitars_list->Guitars.Wood.Deck)<1 || strlen(Guitars_list->Guitars.Wood.Deck)>10)
     printf("Возможно вы ошиблись при вводе?\n(кол-во символов от 1 до 10)\nПовторите ввод: ");
    }

   while(strlen(Guitars_list->Guitars.Wood.Deck)<1 || strlen(Guitars_list->Guitars.Wood.Deck)>10);
   return Guitars_list;
  }
//*****************************************************************************************
//Функция ввода целочисленных переменных в диапазоне
int enter_num(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 && str[i]!='\n'; i++)
       {
        for(int j=0; numbers[j]!='\0' && !check_num; j++)
          if(str[i]==numbers[j] || str[i]=='\n')
            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
LIST* delete_menu(LIST* Guitars_list, int* Count)
  {
   int Q,temp;
   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 - ");
     scanf("%d", &Q);
     fflush(stdin);
     switch(Q)
      {
       case 1:
              if(Guitars_list)
                {
                 Del(&Guitars_list, 0, 0);
                 (*Count)--;
                 Messages(10);
                }
              else
                Messages(9);
              break;
       case 2:
              if(Guitars_list)
                {
                 Del(&Guitars_list, 0, 1);
                 (*Count)--;
                 Messages(10);
                }
              else
                Messages(9);
              break;
       case 3:
              if(Guitars_list)
                {
                 if((*Count)>2)
                   {
                    printf("Введите номер позиции элемента, который следует удалить (от %d до %d): ", 2, (*Count));
                    temp=enter_num(2, (*Count));
                    Del(&Guitars_list, (temp-1), 2);
                    (*Count)--;
                    Messages(10);

                   }
                 else
                   Messages(13);
                }
              else
                Messages(9);
              break;


       case 4:
              fr(&Guitars_list);
              (*Count)=0;
              Messages(9);
              break;



       default:
               if(Q!=5)
                 Messages(7);
      }
    }
   while(Q!=5);
   return Guitars_list;
```

```
  }
//*****************************************************************************************
//Функция удаления n-го элемента в списке
void Del(LIST **Guitars_list, int n, int Key)
  {
   LIST *pBwd = (*Guitars_list), *pFwd = NULL;
   switch(Key)
    {
     case 0:
            *Guitars_list = (*Guitars_list)->next;
            free(pBwd);
            break;
     case 1:
            pFwd = *Guitars_list;
            while(pFwd->next)
             {
              pBwd = pFwd;
              pFwd = pFwd->next;
             }
            if(!pBwd)
              {
               free(*Guitars_list);
               *Guitars_list = NULL;
              }
            else
              {
               free(pFwd->next);
               pBwd->next = NULL;
              }
            break;
     case 2:
            for(int i = 1; i < n && (*Guitars_list)->next; i++)
              (*Guitars_list) = (*Guitars_list)->next;
            pBwd = (*Guitars_list);
            pFwd = pBwd->next;
            pBwd->next = pFwd->next;
            free(pFwd);
            break;

    }
  }
//*****************************************************************************************
//Функция очистки списка
void fr(LIST **Guitars_list)
  {
   LIST *pBwd=NULL;
   if(*Guitars_list)
     {
      while((*Guitars_list)->next)
       {
        pBwd = (*Guitars_list);
        (*Guitars_list) = (*Guitars_list)->next;
        free(pBwd);
       }
      free(*Guitars_list);
      (*Guitars_list)=NULL;
     }
  }
//*****************************************************************************************
//Функция подменю сортировки
LIST* sort_menu(LIST* Guitars_list)
  {
   LIST *temp=NULL;
   int Q;
   do
    {
     system("cls");
     puts("Меню сортировки, выберите по какому пункту сделать сортировку");
     puts("1 - Год производства");
     puts("2 - Количество струн");
     puts("3 - Выход в главное меню");
     printf("Введите номер пункта - ");
     scanf("%d", &Q);
     fflush(stdin);
     if(Q!=3)
       if(Q==1||Q==2)
         temp=Sort(Guitars_list, Q);
       else
         Messages(7);
     if(Q!=3)
       Messages(4);
    }
```

```
   while(Q!=3);

   return temp;
  }
//*****************************************************************************************
//Функция сортировки данных
LIST* Sort(LIST *Guitars_list, int key)
  {
   LIST *temp,*sort=NULL,*pFwd,*pBwd;
   while(Guitars_list)
    {
     temp = Guitars_list;
     Guitars_list = Guitars_list->next;
     for(pFwd=sort,pBwd=NULL;  pFwd && (temp->Guitars.Year > pFwd->Guitars.Year && key==1 ||
         temp->Guitars.Strings > pFwd->Guitars.Strings && key==2); pBwd=pFwd,pFwd=pFwd->next);
        if(!pBwd)
          {
           temp->next=sort;
           sort=temp;
          }
        else
          {
           temp->next=pFwd;
           pBwd->next=temp;
          }
    }
   return sort;
  }
//*****************************************************************************************
//Функция подменю обработки
LIST* processing_menu(LIST *Guitars_list, int *NewCount)
  {
   LIST *New_Guitars_list=NULL;
   const char* str;
   int Q, temp, tempfirst, templast;
   do
    {
     system("cls");
     puts("Меню поиска, выберите по какому пункту сделать выборку");
     puts("1 - Год производства");
     puts("2 - Количество струн");
     puts("3 - Выход в главное меню");
     printf("Введите номер пункта - ");
     scanf("%d", &Q);
     fflush(stdin);
     switch(Q)
      {
       case 1:
              tempfirst=1899;
              templast=2015;
              str="до какого года производства";
              break;
       case 2:
              tempfirst=1;
              templast=20;
              str="до какого количества струн";
              break;
       default:
              if(Q!=3)
                 Messages(7);
      }
     if(Q==1||Q==2)
       {
        printf("Введите %s выводить результаты (от %d до %d): ", str, tempfirst, templast);
        temp=enter_num(tempfirst, templast);
        *NewCount=0;
        New_Guitars_list=Search(Guitars_list, NewCount, Q, temp);
       }
    }
   while(Q!=3);
   return New_Guitars_list;
  }
//*****************************************************************************************
//Функция обработки данных
LIST* Search(LIST *Guitars_list,int *NewCount, int key, int tempint)
  {
   LIST *search=NULL,*pFwd;
     while(Guitars_list)
      {
       if(Guitars_list->Guitars.Year<=tempint && key==1 || Guitars_list->Guitars.Strings<=tempint && key==2)
         {
          pFwd=search;
```

```
          search=(LIST*)malloc(sizeof(LIST));
          search->Guitars=Guitars_list->Guitars;
          search->next=pFwd;
          (*NewCount)++;
         }
       Guitars_list=Guitars_list->next;
      }
   return search;
  }
//*****************************************************************************************
//Функция вывода данных
void Output_list(LIST *Guitars_list)
  {
   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(Guitars_list)
    {
     printf("%12s  | %18d | %14d | %11s | %6s ",Guitars_list->Guitars.Name, Guitars_list->Guitars.Year,
            Guitars_list->Guitars.Strings, Guitars_list->Guitars.Wood.Deck, Guitars_list->
            Guitars.Wood.Neck);
            printf("\n=================================================================================");
     if(Guitars_list->next)
       printf("Для вывода следующего элемента нажмите любую клавишу\r");
     else
        puts("Для завершения просмотра нажмите любую клавишу");
     getch();
     Guitars_list=Guitars_list->next;
    }
  }
//*****************************************************************************************
//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 обработали их");
            puts("Вам необходимо выбрать 5 пункт меню для поиска данных");
            break;
     case 3:
            puts("Для добавления последующих элементов, сначала необходимо добавить первый");
            break;
     case 4:
            puts("Сортировка успешно завершена");
            break;
     case 5:
            puts("Выборка из данных успешно сформирована");
            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");
            break;
     case 10:
             puts("Элемент удалён");
             break;
     case 11:
             puts("Элемент успешно добавлен");
             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 с односвязными списками на языке программирования «C/C++».