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 № 4
на тему:
“Работа с файлами”

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 файлами.

Задание

Написать программу для: создания списка, с возможностью чтения информационных полей с файла и записью информационных полей в файл.

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. Существует два способа вывода списка: на экран и в файл. В пункте добавления элемента, можно создать список из информационных полей заранее сохранённого файла.

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				
	Марка	Год производства	Количество струн	Материал	
				корпус	гриф
1	Gibson	1964	6	Ольха	Кедр
	Fender	1983	6	Сосна	Клён
	Dean	1991	7	Липа	Клён
2	Gibson	1964	6	Ольха	Кедр
	Fender	1983	6	Сосна	Клён
	Dean	1991	7	Липа	Клён
3	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
list	LIST	Указатель на первый элемент исходного 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();
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(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(14): "Список успешно записан в файл"
messages(15): "Список успешно считан с файла"
messages(17): " Файл пустой или файла с таким именем не существует"
messages(18): "Список не записан. Возникли проблемы с файлом, обратитесь к разработчику. "

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 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
temp	LIST	Указатель на элемент списка для вставки
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: Открыть список с файла
5: 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, enterMenu, enterField.

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, enterNum, recordFile, outputList.

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: outputMenu.
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 fileName

Назначение: ввод имени файла.
Прототип: const char* fileName(int Key); описание формальных переменных представлено на рисунке 6.
Возвращаемое значение: имя файла.
Пример вызова: name=fileName(1);
Вызывающая функция: enterMenu, outputMenu.
Вызываемая функция: enterWord.

Описание переменных

Описание переменных функции fileName представлены на рисунке 6.

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

Рис. 6. Описание переменных функции fileName

Описание функции addNth

Назначение: добавление n-го элемента.
Прототип: LIST addNth(LIST list, LIST temp int n); описание формальных переменных представлено на рисунке 7.
Возвращаемое значение: указатель на первый элемент списка.
Пример вызова: addNth(list, temp, 2); описание фактических переменных представлено в таблице 1.
Вызывающая функция: enterMenu.
Вызываемая функция: 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: enterMenu.
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, fileName.

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

Рис. 9. Описание переменных функции enterField

Описание функции fr

Назначение: освобождение памяти.
Прототип: void fr(LIST *list); описание формальных переменных представлено на рисунке 10.
Пример вызова: fr(&list); описание фактических переменных представлено в таблице 1.
Вызывающая функция: main.

Описание переменных

Описание переменных функции fr представлено на рисунке 10.

Имя переменной	Тип	Назначение
Формальные переменные		
List	LIST*	Указатель на адрес первого элемента списка

Рис. 10. Описание переменных функции fr

Описание функции recordFile

Назначение: запись в файл.
Прототип: void recordFile(LIST list, const char* name); описание формальных переменных представлено на рисунке 11.
Пример вызова: recordFile(list, name); описание фактических переменных представлено в табл. 1.
Вызывающая функция: enterMenu.

Описание переменных

Описание переменных функции recordFile представлено на рисунке 11.

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

Рис. 11. Описание переменных функции recordFile

Описание функции readFile

Назначение: запись в файл.
Прототип: LIST readFile(const char* name); описание формальных переменных представлено на рисунке 12.
Пример вызова: list=readFile(name); описание фактических переменных представлено в таблице 1.
Вызывающая функция: outputMenu.
Вызываемая функция: addNth, getElem, count.

Описание переменных

Описание переменных функции readFile представлено на рисунке 12.

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

Рис. 12. Описание переменных функции readFile

Описание функции count

Назначение: подсчёт количества элементов.
Прототип: int count(LIST list); описание формальных переменных представлено на рисунке 14.
Возвращаемое значение: количество элементов в списке.
Пример вызова: temp=count(list); описание фактических переменных представлено в таблице 1.
Вызывающая функция: enterMenu, readFile.

Описание переменных

Описание переменных функции count представлено на рисунке 14.

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

Рис. 14. 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, addNth.

Описание переменных

Описание переменных функции count представлено на рисунке 15.

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

Рис. 15. Описание переменных функции count

Схема вызова функций

Схема вызова функций представлена на рисунке 16.

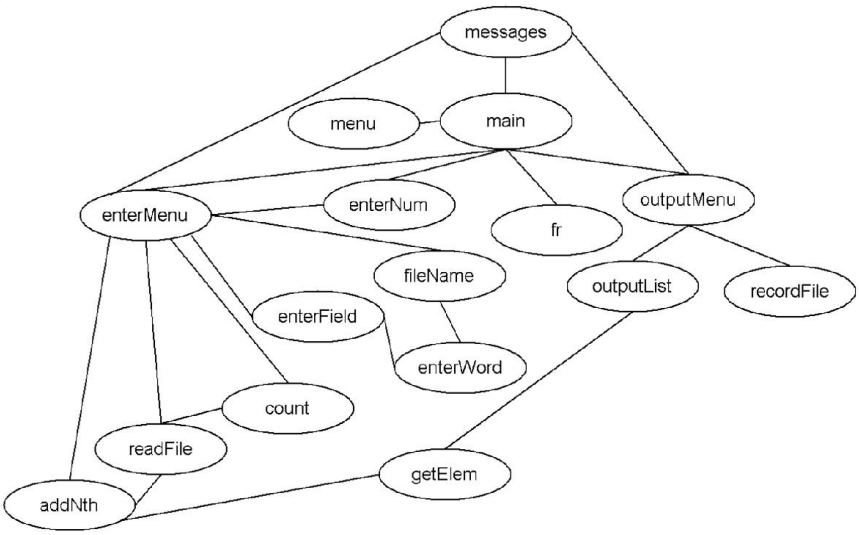

Рис. 16. Схема вызова функций

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 с информационными полями
    struct stList *next;                                    //Следующий элемент
    struct stList *prev;                                    //Предыдущий элемент
  } sLIST;

typedef sLIST* LIST;                                        //Указатель на элемент списка

void menu();                                                //Прототип функции Главного меню
void messages(int Key);                                     //Прототип функции вывода сообщения
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
LIST addNth(LIST list, LIST temp, int n);                   //Прототип функции добавления n-го элемента
LIST enterField();                                          //Прототип функции ввода инф. полей
void enterWord(char* str);                                  //Прототип функции ввода слова
void fr(LIST *list);                                        //Прототип функции освобождения списка
const char* fileName(int Key);                              //Прототип функции ввода имени файла
int recordFile(LIST list, const char* name);                //Прототип функции записи в файл
LIST readFile(const char* name);                            //Прототип функции чтения файла
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;
   int Q;
   system("chcp 1251");
   do
    {
     menu();
     switch (Q=enterNum(1, 4))
      {
       case 1:
              enterMenu(&list);
              break;
       case 2:
              outputMenu(list);
              break;
       case 3:
              messages(8);
              break;
      }
    }
   while(Q!=3);
   fr(&list);
   return 0;
  }
//*****************************************************************************
//Функция меню
void menu()
  {
   system("cls");
   puts("Главное меню");
   puts("1 - Добавление элементов в список");
```

```
   puts("2 - Вывод списка");
   puts("3 - Выход");
   printf("Введите номер пункта - ");
  }
//*****************************************************************************************
//Функция подменю добавления элементов
void enterMenu(LIST* list)
 {
   LIST temp;
   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 - Открыть список из файла");
    puts("5 - Вернуться в главное меню");
    printf("Введите номер пункта - ");
    switch(Q=enterNum(1,5))
     {
       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(3);
              break;
       case 3:
             if(*list)
               if(count(*list)>1)
                 {
                  temp=enterField();
                  printf("Введите номер позиции, куда вставить элемент (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(3);
              break;
      case 4:
             if(*list)
               {
                system("cls");
                puts("Текущий список будет удалён, продолжить?");
                puts("Для продолжения нажмите Enter, для отмены нажмите любую другую клавишу");
                if(getch()=='\n')
                  {
                   fr(list);
                   *list=readFile(fileName(2));
                  }
               }
             else
               *list=readFile(fileName(2));
             if(*list)
               messages(15);
             else
               messages(17);
             break;
      }
   }
  while(Q!=5);
 }
//*****************************************************************************************
//Функция добавления n-го элемента
LIST addNth(LIST list, LIST temp, int n)
  {
    if(!n)
     {
       if(list)
```

```
        list->prev = temp;
       temp->next = list;
       temp->prev = NULL;
       list = temp;
       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 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): ");
   enterWord(list->Guitars.Name);
   printf("Введите количество струн (от %d до %d): ", 1, 20);
   list->Guitars.Strings=enterNum(1, 20);
   //list->Guitars.Strings=1;
   printf("Введите год производства (от %d до %d): ", 1899, 2015);
   //list->Guitars.Year=enterNum(1899, 2015);
   list->Guitars.Year=200;
   printf("Введите название дерева грифа (кол-во символов от 1 до 10): ");
   enterWord(list->Guitars.Wood.Neck);
   printf("Введите название дерева корпуса (кол-во символов от 1 до 10): ");
   enterWord(list->Guitars.Wood.Deck);
   return list;
  }
//*****************************************************************************************
//Функция ввода слова
void enterWord(char* str)
  {
    do
     {
      //gets(str);
      strcpy(str, "Something");
      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);
  }
//*****************************************************************************************
//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 освобождения памяти
void fr(LIST *list)
  {
   if(*list)
     {
      for(;(*list)->next; (*list)=(*list)->next, free((*list)->prev));
      free(*list);
      (*list)=NULL;
     }
  }
//*****************************************************************************************
//Функция ввода названия файла
const char* fileName(int Key)
 {
    system("cls");
    char name[10], *temp;
    switch(Key)
    {
     case 1:
             puts("Задайте имя файла");
             puts("Если такой файл не существует, он будет создан, иначе - перезаписан");
             break;
     case 2:
             puts("Введите имя файла");
             break;
     }
    printf("Имя файла - ");
    enterWord(name);
    temp=(char*)malloc(strlen(name)*sizeof(char));
    strcpy(temp, name);
    return temp;
 }
//*****************************************************************************************
//Функция записи в файл
int recordFile(LIST list, const char* name)
  {
   FILE* file;
   system("cls");
   if(file=fopen(name,"w"))
     {
      while(list)
       {
        fwrite(&(list->Guitars),sizeof(list->Guitars),1,file);
        list=list->next;
       }
      fclose(file);
      return 0;
     }
   else
     return 1;
  }
//*****************************************************************************************
//Функция чтения файла
LIST readFile(const char* name)
  {
   LIST list=NULL, temp;
   FILE* file;
   system("cls");
   if(!(file=fopen(name,"r")))
     return NULL;
   while(!feof(file))
    {
     temp=(LIST)malloc(sizeof(sLIST));
     fread(&(temp->Guitars),sizeof(temp->Guitars),1,file);
     if(!list)
       list=addNth(list, temp, 0);
     else
       list=addNth(list, temp, count(list));
    }
   temp=getElem(list, 1, 0);
   if(!temp->prev)
```

```
     list = NULL;
    else
      temp->prev->next = NULL;
    free(temp);
    fclose(file);
    return getElem(list,0,0);
   }
//*****************************************************************************************
//Функция подменю вывода
void outputMenu(LIST list)
  {
   system("cls");
   if(list)
    {
     puts("Меню вывода списка");
     puts("1 - Вывести список на экран");
     puts("2 - Сохранить список в файл");
     puts("3 - Вернуться в главное меню");
     switch(enterNum(1,3))
      {
       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);
              outputList(list, enterNum(1,2)-1);
              break;
       case 2:
              if(!recordFile(list, fileName(1)))
                messages(14);
              else
                messages(18);
              break;
      }
    }
   else
     messages(9);
  }
//*****************************************************************************************
//Функция вывода данных
void outputList(LIST list, int Key)
  {
   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 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 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 14:
             puts("Список успешно записан в файл");
             break;
     case 15:
             puts("Список успешно считан с файла");
             break;
     case 17:
             puts("Файл пустой или файла с таким именем не существует");
             puts("Измените имя файла!");
             break;
     case 18:
             puts("Список не записан. Возникли проблемы с файлом, обратитесь к разработчику.");
             puts("Komdosh@gelezo2.ru");
             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 файлами на языке программирования «C/C++».