# Лабораторная работа.3. Строки, функции, файлы

**Цель выполнения лабораторной работы**

Освоение работы со строками, функциями и файлами в языке Си.

**Порядок выполнения работы**

Обучаемые разрабатывают программы в соответствии с представленным описанием по выданным им индивидуальным заданиям.

**Символы и строки**

Приступая к решению задач этого раздела, следует вспомнить, что:

каждому символу соответствует число — код символа;

в С++ строка — это массив символов;

последним символом строки обязательно должен быть **нуль-символ**, код которого равен 0, и который в тексте программы изображается так: '\0';

сообщения или подсказки, используемые в программе, удобно представить как массив указателей на строки и инициализировать массив, задать сообщения в инструкции объявления массива:

```
char *mes[] ={"Сообщение 1","Сообщение 2", ... , "«Сообщение"} ;
```

если вводимая во время работы программы строка содержит пробелы, то функция scanf вводит только часть строки до первого пробела, а функция gers — всю строку, в том числе и соответствующий клавише <Enter> символ ' \n'.

Рассмотрим пример реализации программы для работы с символами строк.

**Задание 1.**

Скомпилировать представленные 2 примера программ. Выявить особенности работы с символами строк.

**Пример программы 1:**

Написать программу, которая проверяет, является ли введенная с клавиатуры строка дробным числом.

Код программы:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   char st[20];   // строка
   int i;        // номер проверяемого символа
   int ok = 0;   // пусть строка — не дробное число

   printf("Enter a digit with floating point, and press <Enter>");
   printf("->");
   scanf("%s", &st);
```

```
    i = 0;
    if  (st[i] >= '1' && st[i] <='9') // первый символ - цифра
    {
        //за цифрой могут быть еще цифры
        while ( st[i] >= '1' && st[i] <='9' )
        i++;
        //за цифрами должна быть точка
        if (st[i] == '.')
        {
          i++;
          //за точкой должна быть хотя бы одна цифра
          if  (st[i] >='1' && st[i] <='9')
          {
           //и еще цифры
           while ( st[i] >= '1' && st[i] <='9' )
              i++;
           ok = 1; // похоже строка - дробное число
           }
        }
     }
     printf("String: %s \n is",st);
     if ( st[i] || !ok )
        printf(" not");
     printf(" digit with floating point.\n");

     printf("\n Press enter for finish <Enter>\n");
     getch();
}
```

Анализ кода программы:

Условие `st[i] >= '1'` сравнивает значение кода символа с индексом i из введенной строки и значение кода символа «1». Как можно видеть по таблице кодов символов (см. ниже Таблица 5) коды символы чисел располагаются по порядку, поэтому можно задать представленное условие для определения того, что символ является числом.

Кроме этого исходя из даных (Таблица 5) возможно получать значение цифры по коду символа:

```
int a = st[0] - 48; // значение цифры первого символа
```

Это конечно верно только при условии, что символ st[0] является цифрой, т.е. его код лежит в пределах от48 до 57.

**Таблица 5 Таблица значений Ascii символов**

| Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char | Dec | Hex | Oct | Char |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | 32 | 20 | 40 | [space] | 64 | 40 | 100 | @ | 96 | 60 | 140 | ` |
| 1 | 1 | 1 | | 33 | 21 | 41 | ! | 65 | 41 | 101 | A | 97 | 61 | 141 | a |
| 2 | 2 | 2 | | 34 | 22 | 42 | " | 66 | 42 | 102 | B | 98 | 62 | 142 | b |
| 3 | 3 | 3 | | 35 | 23 | 43 | # | 67 | 43 | 103 | C | 99 | 63 | 143 | c |
| 4 | 4 | 4 | | 36 | 24 | 44 | $ | 68 | 44 | 104 | D | 100 | 64 | 144 | d |
| 5 | 5 | 5 | | 37 | 25 | 45 | % | 69 | 45 | 105 | E | 101 | 65 | 145 | e |
| 6 | 6 | 6 | | 38 | 26 | 46 | & | 70 | 46 | 106 | F | 102 | 66 | 146 | f |
| 7 | 7 | 7 | | 39 | 27 | 47 | ' | 71 | 47 | 107 | G | 103 | 67 | 147 | g |
| 8 | 8 | 10 | | 40 | 28 | 50 | ( | 72 | 48 | 110 | H | 104 | 68 | 150 | h |
| 9 | 9 | 11 | | 41 | 29 | 51 | ) | 73 | 49 | 111 | I | 105 | 69 | 151 | i |
| 10 | A | 12 | | 42 | 2A | 52 | * | 74 | 4A | 112 | J | 106 | 6A | 152 | j |
| 11 | B | 13 | | 43 | 2B | 53 | + | 75 | 4B | 113 | K | 107 | 6B | 153 | k |
| 12 | C | 14 | | 44 | 2C | 54 | , | 76 | 4C | 114 | L | 108 | 6C | 154 | l |
| 13 | D | 15 | | 45 | 2D | 55 | - | 77 | 4D | 115 | M | 109 | 6D | 155 | m |
| 14 | E | 16 | | 46 | 2E | 56 | . | 78 | 4E | 116 | N | 110 | 6E | 156 | n |
| 15 | F | 17 | | 47 | 2F | 57 | / | 79 | 4F | 117 | O | 111 | 6F | 157 | o |
| 16 | 10 | 20 | | 48 | 30 | 60 | 0 | 80 | 50 | 120 | P | 112 | 70 | 160 | p |
| 17 | 11 | 21 | | 49 | 31 | 61 | 1 | 81 | 51 | 121 | Q | 113 | 71 | 161 | q |
| 18 | 12 | 22 | | 50 | 32 | 62 | 2 | 82 | 52 | 122 | R | 114 | 72 | 162 | r |
| 19 | 13 | 23 | | 51 | 33 | 63 | 3 | 83 | 53 | 123 | S | 115 | 73 | 163 | s |
| 20 | 14 | 24 | | 52 | 34 | 64 | 4 | 84 | 54 | 124 | T | 116 | 74 | 164 | t |
| 21 | 15 | 25 | | 53 | 35 | 65 | 5 | 85 | 55 | 125 | U | 117 | 75 | 165 | u |
| 22 | 16 | 26 | | 54 | 36 | 66 | 6 | 86 | 56 | 126 | V | 118 | 76 | 166 | v |
| 23 | 17 | 27 | | 55 | 37 | 67 | 7 | 87 | 57 | 127 | W | 119 | 77 | 167 | w |
| 24 | 18 | 30 | | 56 | 38 | 70 | 8 | 88 | 58 | 130 | X | 120 | 78 | 170 | x |
| 25 | 19 | 31 | | 57 | 39 | 71 | 9 | 89 | 59 | 131 | Y | 121 | 79 | 171 | y |
| 26 | 1A | 32 | | 58 | 3A | 72 | : | 90 | 5A | 132 | Z | 122 | 7A | 172 | z |
| 27 | 1B | 33 | | 59 | 3B | 73 | ; | 91 | 5B | 133 | [ | 123 | 7B | 173 | { |
| 28 | 1C | 34 | | 60 | 3C | 74 | < | 92 | 5C | 134 | \ | 124 | 7C | 174 | \| |
| 29 | 1D | 35 | | 61 | 3D | 75 | = | 93 | 5D | 135 | ] | 125 | 7D | 175 | } |
| 30 | 1E | 36 | | 62 | 3E | 76 | > | 94 | 5E | 136 | ^ | 126 | 7E | 176 | ~ |
| 31 | 1F | 37 | | 63 | 3F | 77 | ? | 95 | 5F | 137 | _ | 127 | 7F | 177 | |

**Пример программы 2:**

Написать программу, которая удаляет из введенной с клавиатуры строки начальные пробелы.

```
#include <stdio.h>
#include <conio.h>
#include "string.h"
void main()
{
     unsigned char sst[80];  // Строка
     unsigned char dst[80];  // Буффер
     int i,j;

     printf("Remove the whitespaces symbols in the beginning\n");
     printf("Enter a string:");

     i=0;
     while ((sst[i] = getch()) != 13 && i < 79)
     {
        putch(sst[i]);
        i++;
     }
     sst[i] = '\0'; // нулевой символ в конце строки

     i = 0; j = 0;
     // найдем первый символ, отличный от пробела
     while( sst[i] && sst[i] == ' ')
         i++;

     // здесь i - номер первого символа, отличного от пробела,
     // скопируем sst в dst

     while (sst[i])
         dst[j++] = sst[i++];
     dst[j] = '\0';
```

```
        printf("\nString without of whitespaces:%s\n",dst);

        printf("\n Press enter for finish <Enter> \n");
        getch();
}
```

Анализ кода программы:

```
        while ((sst[i] = getch()) != 13 && i < 79)
           putch(sst[i++]);
```

Ввод символов осуществляется с помощью сочетания использования в бесконечном цикле while 2-х функций getch и putch. При этом getch получает код сивола и возвращает его в массив sst, а putch выводит в консоль полученный символ для пользователя.

Проверка содержимого введенной строчки производится в друго цикле while. При этом по достижении первого не пустого символа цикл прекращает выполнение. В этом случае итератор I указывает на этот первый не пустой символ.

В третьем применении цикла while выход из цикла осуществляетя при достижение нулевого символа в конце строки. В этомцикле производится копирование символов в другой массив.

Дополнительгная информация по приемам работы с символами доступна в прилагаемой эллектронной версии книги В.В. Подбельский, С.С. Фомин «Программирование на языке Си» раздел 4.3 страница 192-202.

**Функции ввода-вывода**

**printf**

Синтаксис:

```
int printf (Формат, Список Переменых);
```

Выводит на экран значения переменных. Формат вывода задается в строке форматирования, которая помимо спецификатора формата может содержать текст и управляющие символы. Значение первой переменной выводится в соответствии с первым спецификатором формата, второй — со вторым, и т. д.

Спецификаторы формата (необязательный параметр n задает ширину поля вывода).

| Спецификатор | Форма вывода |
|---|---|
| %ni %nd | Десятичное число со знаком |
| %nu | Беззнаковое целое десятичное число |
| %n.mf | Дробное число с десятичной точкой. Необязательный параметр m задает количество цифр дробной части |
| %ne | Дробное число с десятичной точкой или, если число не может быть представлено в форме с десятичной точкой, в экспоненциальной форме |
| %ns | Строка символов |

| | |
|---|---|
| %nc | Символ |

Управляющие и специальные символы.

| Символ | Действие |
|---|---|
| \n | Переводит курсор в начало следующей строки |
| \t | Переводит курсор в очередную позицию табуляции |
| \\ | Бэкслэш |
| \’ | Кавычка |

Заголовочный файл: <stdio.h>

**scanf**

Синтаксис:

```
int scanf (const char* Формат,   СписокАдресовПеременных) ;
```

Вводит с клавиатуры значения переменных, в соответствии с указанным спецификатором формата. Первая переменная получает значение в соответствии с первым спецификатором формата, вторая — со вторым и т. д.

В качестве параметра функции scanf должны передаваться адреса переменных, а не их имена.

| Спецификатор | Вводит |
|---|---|
| %i %d | Десятичное число со знаком |
| %u | Беззнаковое целое десятичное число |
| %f %e | Дробное число |
| %s | Строка символов |
| %c | Символ |

Заголовочный файл: <stdio.h>

**puts**

Синтаксис:

```
puts(const char* Строка);
```

Выводит на экран строку символов и переводит курсор в начало следующей строки экрана. В качестве параметра функции можно использовать строковую константу или строковую переменную.

Заголовочный файл: <stdio.h>

**gets**

Синтаксис:

```
char *gets(char*  s) ;
```

Вводит с клавиатуры строку символов. Вводимая строка может содержать пробелы.

Заголовочный файл: <stdio.h>

**putch**

Синтаксис:

```
int putch(int c);
```

Выводит на экран символ. Заголовочный файл: <conio.h>

**getch**

Синтаксис:

```
int getch(void);
```

Возвращает код символа нажатой клавиши. Если нажата служебная клавиша, то функция getch возвращает 0. В этом случае, для того, чтобы определить, какая служебная клавиша нажата, нужно обратиться к функции getch еще раз.

Функция getch не выводит на экран символ, соответствующий нажатой клавише.

Заголовочный файл: <conio.h>

**cputs**

Синтаксис:

```
cputs(const char*   Строка);
```

Выводит на экран строку. Цвет выводимых символов можно задать при помощи функции textcolor, цвет фона — при помощи функции textbackground.

Для перехода к началу следующей строки вместо \n следует использовать символы \n\r, иначе курсор лишь переводится1 на новую строку, но не возвращается к левой границе окна. То же самое относится и к функции cprintf.

Заголовочный файл: <conio.h>

**cprintf**

Как и функция printf, функция cprintf используется для вывода на экран сообщений и значений переменных. При этом имеется возможность задать цвет выводимых символов (функция textcolor) и цвет фона (textbackground).

Заголовочный файл: <conio.h>

**textcolor**

Синтаксис:

```
void textcolor(int Цвет);
```

Задает цвет для выводимого функциями cputs и cprintf текста. В качестве параметра Цвет обычно используют одну из перечисленных ниже именованных констант.

| Цвет | Константа | Значение константы |
|---|---|---|
| Черный | BLACK | 0 |
| Синий | BLUE | 1 |
| Зеленый | GREEN | 2 |
| Бирюзовый | CYAN | 3 |
| Красный | RED | 4 |
| Сиреневый | MAGENTA | 5 |
| Коричневый | BROWN | 6 |

| Цвет | Константа | Значение константы |
|---|---|---|
| Светло-серый | LIGHTGRAY | 7 |
| Серый | DARKGRAY | 8 |
| Голубой | LIGHTBLUE | 9 |
| Светло-зеленый | LIGHTGREEN | 10 |
| Светло-бирюзовый | LIGHTCYAN | 11 |
| Алый | LIGHTRED | 12 |
| Светло-сиреневый | LIGHTMAGENTA | 13 |
| Желтый | YELLOW | 14 |
| Белый (яркий) | WHITE | 15 |

Заголовочный файл: <conio.h>

**textbackground**

Синтаксис:

```
void textbackground(int Цвет);
```

Задает цвет фона, на котором появляется текст, выводимый функциями cputs и cprintf. В качестве параметра Цвет обычно используют одну из перечисленных ниже именованных констант.

| Цвет | Константа | Значение константы |
|---|---|---|
| Черный | BLACK | 0 |
| Синий | BLUE | 1 |
| Зеленый | GREEN | 2 |
| Бирюзовый | CYAN | 3 |
| Красный | RED | 4 |
| Сиреневый | MAGENTA | 5 |
| Коричневый | BROWN | 6 |
| Светло-серый | LIGHTGRAY | 7 |

Заголовочный файл: <conio.h>

**gotoxy**

Синтаксис:

```
void gotoxy(int x,   int y)
```

Переводит курсор в позицию с указанными координатами. Координата x задает номер колонки, координата y — номер строки, на пересечении которых находится знакоместо, куда переводится курсор.

Заголовочный файл: <conio.h>

**clrscr**

Синтаксис:

```
void clrscr (void)
```

Очищает экран и закрашивает его цветом, заданным функцией textbackground.

Заголовочный файл: <conio.h>

**window**

Синтаксис:

```
void window(int xl,   int yl,   int x2,   int y2) ;
```

Определяет окно — область экрана. Параметры xl, yl задают координаты левого верхнего угла окна относительно экрана, параметра x2,y2 — правого нижнего.

Заголовочный файл: <conio.h>

### Функции работы со строками

**strcat**

Синтаксис:

```
char *strcat(char*  Строка1,   const char*  Строка2)
```

Объединяет строки Строка 1 и Строка2 и записывает результат в строку Строка 1.

Заголовочный файл: <string.h>

**strcpy**

Синтаксис:

```
char *strcpy(char*  Строка 1,   const char*  Строка2)
```

Копирует строку Строка1 в строку Строка2.

Заголовочный файл: <string.h>

**strlen**

Синтаксис:

```
int strlen(const char* Строка)
```

Возвращает длину строки. Нулевой символ не учитывается.

Заголовочный файл: <string.h>

**strcmp**

Синтаксис:

```
int strcmp (const char* Строка1, const char* Строка2)
```

Сравнивает строки Строка 1 и Строка2. Возвращает 0, если строки равны, число меньше нуля, если Строка 1 < Строка2 и число больше нуля, если Строка 1 > Строка2.

Заголовочный файл: <string.h>

**strlwr**

Синтаксис:

```
char* strlwr(char* Строка)
```

Преобразует строчные символы строки в прописные (обрабатывает только буквы латинского алфавита).

Заголовочный файл: <string.h>

**strupr**

Синтаксис:

```
char*  strupr(char*  Строка)
```

Преобразует прописные символы строки в строчные (обрабатывает только буквы латинского алфавита).

Заголовочный файл: <string.h>

**strset**

Синтаксис:

```
char* strset(char* Строка,   char Символ)
```

Заполняет строку указанным при вызове функции символом.

Заголовочный файл: <string.h>

**strchr**

Синтаксис:

```
char* strchr(const char* Строка,   int Символ)
```

Выполняет поиск символа в строке и возвращает указатель на первый найденный символ или, если символ найден, NULL.

Заголовочный файл: <string.h>

**Индивидуальное задание №1**

1. Написать программу, которая запрашивает имя пользователя и здоровается с ним. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

```
Как Вас зовут?
Введите  свои имя и фамилию,   затем нажмите <Enter>
-> Вася Иванов
Здравствуйте, Вася Иванов!
```

2. Написать программу, которая запрашивает у пользователя имя и отчество, затем здоровается с ним. Для ввода используйте функцию getch().
3. Напишите программу, которая вычисляет длину введенной с клавиатуры строки.
4. Напишите программу, которая выводит на экран сообщение в "телеграфном" стиле: буквы сообщения должны появляться по одной, с некоторой задержкой.
5. Напишите программу, которая выводит код введенного пользователем символа. Программа должна завершать работу в результате ввода, например, точки. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

```
Введите символ и нажмите <Enter>.
```

```
Для завершения введите точку.
-> 1
Символ:   1 Код: 4 9
-> 2
Символ:   2  Код:   50
-> ы
Символ:   ы Код:   235
->.
```

6. Написать программу, которая выводит на экран первую часть таблицы кодировки символов (символы с кодами от 0 до 127). Таблица должна состоять из восьми колонок и шестнадцати строк. В первой колонке должны быть символы с кодом от О до 15, во второй — от 16 до 31 и т. д.
7. Написать программу, которая в введенной с клавиатуры строке преобразует строчные буквы латинского алфавита в прописные. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

```
Введите строку текста и нажмите <Enter>
-> learning C++ for programming under the Windows systems
Строка, преобразованная к верхнему регистру:
learning C++ for programming under the Windows systems
```

8. Написать программу, которая удаляет из введенной с клавиатуры строки первые 2 слова если это цифры.
9. Написать программу, которая проверяет, является ли введенная с клавиатуры строка целым числом. Рекомендуемый вид экрана во время выполнения программы приведен ниже (данные, введенные пользователем, выделены полужирным шрифтом).

```
Введите число и нажмите  <Enter> ->  23.5
Введенная  строка не  является целым числом.
```

10. Написать программу, которая проверяет, является ли введенная с клавиатуры строка двоичным числом.
11. Написать программу, которая проверяет, является ли введенная с клавиатуры строка шестнадцатеричным числом.
12. Написать программу, которая преобразует введенное с клавиатуры восьмиразрядное двоичное число в десятичное. Рекомендуемый вид экрана во время выполнения программы приведен ниже (введенные пользователем данные выделены полужирным шрифтом).

```
Введите восьмиразрядное двоичное число
и нажмите <Enter>
-> 11101010
Двоичному числу 11101010 соответствует десятичное 234
Для завершения нажмите <Enter>
```

13. Написать программу, которая преобразует введенное с клавиатуры двухразрядное шестнадцатеричное число в десятичное.
14. Написать программу, которая преобразует введенное пользователем десятичное число в число в указанной восьмеричной системе счисления. Рекомендуемый вид экрана во время выполнения программы приведен ниже.

```
Введите целое число  ->  67
```

```
  Десятичному числу 67   соответствует число   103 по основанию системы
исчисления 8
```

15. Написать программу, которая преобразует введенное пользователем десятичное число в шестнадцатеричное.

**Функции**

Приступая к решению задач этого раздела, следует вспомнить, что:

- для передачи данных в функцию надо использовать только параметры. Глобальные переменные, т. е. переменные, объявленные вне функции, использовать не рекомендуется;
- тип каждого фактического параметра (константы или переменной) в инструкции вызова функции должен совпадать с типом соответствующего формального параметра, указанного в объявлении функции;
- если параметр функции используется для возврата результата, то в объявлении функции этот параметр должен быть ссылкой, а в инструкции вызова функции в качестве фактического параметра должен быть указан адрес переменной.

Дополнительгная информация по приемам работы с функциями доступна в прилагаемой эллектронной версии книги В.В. Подбельский, С.С. Фомин «Программирование на языке Си» разделах 5.1, 5.2, 5.3 стр. 203-227.

**Задание 2.**

Скомпилировать представленный пример программы. Выявить особенности работы с функциями.

Пример:

Написать функцию, которая выводит строку, состоящую из одинаковых символов. Длина строки и символ являются параметрами процедуры.

```
#include "stdio.h"
#include "conio.h"

// Выводит на экран строку состоящую
// из n заданных символов
void line(char ch, int n)
{
    int i;
    for (i = 0; i < n; i++)
        putch(ch);
}

// возвращает длину строки
int length(char* st)
{
    int l = 0;  // длина строки
    char* p = st;  // указатель на символ

    while ( *p++  )
        l++;
    return(l);
}

void main()
```

```
{
    char mes[] = "Hello, World!\0"; // строка с нулевым символом
    int len;

    len = length(mes); // получает длинну строки

    line('*', len); // выводим символы полученной длинны
    printf("\nHello, World!\n");
    line('*',len); // выводим символы
    printf("\n Press enter for finish <Enter>");
    getch();
}
```

Анализ кода программы:

Объявлены 2 функции для их использования в главном методе main. Функции реализованы перед методом main, в противном случае необходимо было объявить прототипы этих функций пред методом main.

В функции length для подсчета колличества символов используется указатель, который в цикле while постепенно перемещается от сивола к символу пока не достигнет конечного нулевого символа строки. В качестве аргумента функции length переадется указатель на первый символ строки.

**Индивидуальное задание №2**

1. Написать функцию, которая вычисляет объем цилиндра. Параметрами функции должны быть радиус и высота цилиндра.
2. Написать функцию, которая возвращает максимальное из двух целых чисел, полученных в качестве аргумента.
3. Написать функцию, которая сравнивает два целых числа и возвращает результат сравнения в виде одного из знаков: >, < или =.
4. Написать функцию, которая вычисляет сопротивление цепи, состоящей из двух резисторов. Параметрами функции являются величины сопротивлений и тип соединения (последовательное или параллельное). Функция должна проверять корректность параметров: если неверно указан тип соединения, то функция должна возвращать -1.
5. Написать функцию, которая вычисляет значение аь. Числа a и b могут быть любыми дробными положительными числами.
6. Написать функцию Procent, которая возвращает процент от полученного в качестве аргумента числа.
7. Написать функцию "Факториал" и программу, использующую эту функцию для вывода таблицы факториалов.
8. Написать функцию Dohod, которая вычисляет доход по вкладу. Исходными данными для функции являются: величина вклада, процентная ставка (годовых) и срок вклада (количество дней).
9. Написать функцию glasn, которая возвращает 1, если символ, полученный функцией в качестве аргумента, является гласной буквой латинского алфавита, и ноль — в противном случае.
10. Написать функцию sogl, которая возвращает 1, если символ, полученный функцией в качестве аргумента, является согласной буквой латинского алфавита, и 0 — в противном случае.

11. Написать функцию, которая возвращает преобразованную к верхнему регистру строку, полученную в качестве аргумента.
12. Написать функцию, обеспечивающую решение квадратного уравнения. Параметрами функции должны быть коэффициенты и корни уравнения. Значение, возвращаемое функцией, должно передавать в вызывающую программу информацию о наличии у уравнения корней: 2 — два разных корня, 1 — корни одинаковые, 0 — уравнение не имеет решения. Кроме того, функция должна проверять корректность исходных данных. Если исходные данные неверные, то функция должна возвращать -1.
13. Написать функцию, которая выводит на экран строку, из слов, разделяемых символами подчеркивания вместо пробелов.
14. Написать функцию, которая вычисляет объем и площадь поверхности параллелепипеда.

**Задачи повышенной сложности**

15. Написать функцию, обеспечивающую ввод с клавиатуры целого положительного числа. При нажатии клавиши соответствующий символ должен появляться на экране только в том случае, если этот символ является цифрой. Функция должна позволять редактировать введенное число при помощи клавиши <Backspace>. При нажатии клавиши <Enter> функция должна завершать работу и возвращать введенное число.
16. Написать функцию, обеспечивающую ввод с клавиатуры дробного числа. При нажатии клавиши соответствующий символ должен появляться на экране только в том случае, если этот символ является допустимым в данной позиции. Например, функция не должна допускать ввод более чем одной точки и знака минус не в первой позиции. Функция должна позволять редактировать введенное число при помощи клавиши <Backspace>. При нажатии клавиши <Enter> функция должна завершать работу и возвращать введенное число.
17. Написать программу, реализующую игру "21”. Действия по выдаче очередной карты игроку и компьютеру реализуйте в виде функции.

**Файлы**

Приступая к решению задач этого раздела, следует вспомнить, что:

В программе, которая выполняет операции чтения из файла или запись в файл, должна быть объявлена переменная-указатель на тип FILE.

Для того чтобы файл стал доступен, его нужно открыть, указав, для выполнения какой операции открывается файл (чтение, запись или обновление данных) и тип файла (двоичный или текстовый).

При работе с файлами возможны ошибки. Поэтому рекомендуется проверять результат выполнения потенциально опасных, с точки зрения возникновения ошибок, операций с файлами (например, fopen).

Если в инструкции открытия файла путь к файлу не указан, то по умолчанию предполагается, что файл находится в том же каталоге, что и выполняемый файл программы.

При записи в тексте программы пути к файлу, следует помнить, что символ \ в строковых константах обозначает начало специальной последовательности символов (например, \n). Поэтому, чтобы путь к файлу был интерпретирован правильно, символ \ следует продублировать (например, c:\\temp).

Чтение данных из текстового файла обеспечивает функция fscanf, запись — fprintf.

По завершении работы с файлом его нужно обязательно закрыть (функция fclose).

Дополнительгная информация по приемам работы с файлами доступна в прилагаемой эллектронной версии книги В.В. Подбельский, С.С. Фомин «Программирование на языке Си» разделах 7.1.1, 7.1.2, 7.1.3 стр. 325-368.

**Задание 3.**

Скомпилировать представленный пример программы. Выявить особенности работы с файлами.

Пример программы:

Напишите программу, которая дописывает в находящийся на диске: файл phone.txt имя, фамилию и номер телефона, например, вашего товарища. Если файла на диске нет, то программа должна создать его. В файле каждый элемент данных (имя, фамилия, телефон) должен находиться в отдельной строке. В конце программа выводит строки из файла. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
Добавление в телефонный справочник
Фамилия -> Сидоров
Имя -> Вася
Телефон -> 234-84-37
Информация добавлена.
Для завершения работы нажмите <Enter>
```

Код программы:

```
#include "stdio.h"
#include "conio.h"

#define FNAME "phones.txt\0" // имя файла
void main()
{
    char fname[20] = FNAME;
    FILE *f;  // файл данных

    char fam[15];  // фамилия
    char name[15]; // имя
    char tel[9];   // номер телефона

    puts("\n Adding the new nomber to the phones book \n");

    // Открыть файл в режиме добавления (a) текста (t)
    // Если файла с таким именем нет, то он будет создан
    if ((f = fopen(fname, "at")) == NULL)
```

```
    {
        printf("Error in file open operation! \n");
        getch();
        return;
    }

    // получим данные от пользователя
    printf("Second name ->");
    scanf("%s", &fam);
    printf("First name ->");
    scanf("%s", &name);
    printf("Phone number ->");
    scanf("%s", &tel);
    // и запишем их в файл
    fprintf(f,"%s %s %s \n", fam, name, tel);
    puts("\n Data were added.");
    fclose(f);     // закрытие файла

    // открытие для чтения
    if ((f = fopen(fname, "rt")) == NULL)
    {
        printf("Error in file open operation!  %s \n", fname);
        getch();
        return;
    }

    printf("\nContent of file is following: \n");
    while (!feof(f))
    {
         fscanf(f,"%s %s %s \n", &fam, &name, &tel);

         printf("%s %s %s \n", fam, name, tel);
    }
    fclose(f);     // закрытие файла

    printf("\n Press enter for finish <Enter>\n");
    getch();
}
```

Анализ кода программы:

Объявлены в программе выполняется 2 операции:

- ввод данных в файл в виде строковых переменных и
- вывод данных из текстового файла по форматированным строкам.

В первой части файл открывает в режиме добавления текста с флагами “at”. Поэтому при вызове функции fprintf данные добаляются в конец файла.

Во второй части программы файл открывает для чтения текста с флагами “rt”. Чтение строк из файла осуществояется в цикле. Условием цикла являетс достижения конца файла при последней операции чтения.

### Функции работы с файлами

**fopen**

Синтаксис:

```
FILE* fopen(const char * Имя,   const char* Режим)
```

Открывает файл с указанным именем для действия, которое задается параметром Режим.

| Режим | Действие |
|---|---|
| r | Только запись. Файл открывается только для чтения |
| w | Чтение. Файл открывается для записи. Если файл с указанным в качестве первого параметра функции fopen уже существует, то новые данные записываются поверх старых, т. е. старый файл фактически уничтожается |
| a | Добавление. Файл открывается для записи данных в конец существующего файла. Если файл с указанным в качестве первого параметра функции fopen не существует, то он будет создан |

Если файл открывается как текстовый, то после символьной константы, определяющей режим открытия файла, нужно добавить символ t. Например, строка rt задает, что для чтения открывается текстовый файл.

В случае успешного открытия файла функция fopen возвращает указатель на поток, из которого можно читать или в который можно записывать. Если по какой-либо причине операция открытия файла не была выполнена, fopen возвращает null. В этом случае, чтобы получить информацию о причине ошибки, следует обратиться к функции ferror.

Заголовочный файл: <stdio.h>

**fprintf**

Синтаксис:

```
int fprintf(FILE * Поток,   Формат,   СписокПеременых) ;
```

Выполняет форматированный вывод (см. printf) в файл, связанный с потоком, указанным в качестве первого параметра.

Файл, связанный с потоком, должен быть открыт как текстовый, в режиме, допускающем запись (см. fopen).

Заголовочный файл: <stdio.h>

**fscanf**

Синтаксис:

```
int fscanf(FILE *Поток, const char* Формат,   СписокАдр)/
```

Выполняет форматированное (см. scanf) чтение значений переменных из файла, связанного с потоком, указанным в качестве первого параметра.

Файл, связанный с потоком, должен быть открыт как текстовый, в режиме, допускающем чтение (см. fopen).

Заголовочный файл: <stdio.h>

**fgets**

Синтаксис:

```
char* fgets(char *Строка, int КолСимволов,   FILE *Поток)
```

Читает из указанного потока символы и записывает их в строку, указанную при вызове функции. Чтение заканчивается, если прочитан символ с номером КолСимволов-1 или если очередной символ является символом новой строки.

Прочитанный из файла символ новой строки заменяется нулевым символом.

Файл, связанный с потоком, должен быть открыт как текстовый, в режиме, допускающем чтение (см. fopen).

Заголовочный файл: <stdio.h>

**fputs**

Синтаксис:

```
char* fputs(char *Строка,   FILE *Поток)
```

Записывает в указанный поток строку символов. Символ конца строки, нуль-символ, в поток не записывается.

Файл, связанный с потоком, должен быть открыт как текстовый, в режиме, допускающем запись (см. fopen).

Заголовочный файл: <stdio.h>

**ferror**

Синтаксис:

```
int ferror(FILE*  Поток)
```

Возвращает ненулевое значение, если последняя операция с указанным потоком завершилась ошибкой.

Заголовочный файл: <stdio.h>

**feof**

Синтаксис:

```
int feof(FILE*  Поток)
```

Возвращает ненулевое значение, если в результате выполнения последней операции чтения из потока достигнут конец файла.

Заголовочный файл: <stdio.h>

**fclose**

Синтаксис:

```
int fclose(FILE*  Поток)
```

Закрывает указанный поток.

Заголовочный файл: <stdio.h>

**Индивидуальное задание №3**

1. Напишите программу, которая на сменном диске компьютера (диск D:) создает файл numbers.txt и записывает в него 5 введенных пользователем целых чисел. Просмотрите при помощи редактора текста, например, встроенного в Norton Commander, созданный файл. Убедитесь, что каждое число находится в отдельной строке.
2. Напишите программу, которая дописывает в файл D:\numbers.txt пять введенных пользователем целых чисел. Убедитесь при помощи редактора текста, что в файле находятся новые числа.

3. Напишите программу, которая выводит на экран содержимое файла D:\numbers.txt. и выводит сумму чисел файла.
4. Напишите программу, которая вычисляет среднее арифметическое чисел, находящихся в файле D:\numbers.txt.
5. Напишите программу, которая записывает в указанный файл данные находящиеся в двумерном массиве дробного типа. Объявление массива будет выглядеть следующим образом:

```
#define NR 3
#define NC 6
float a[NR][NC] =
     {
        15.0,16.5,18.0,19.5,21.0,24.0,
        16.5,18.0,19.5,21.0,22.5,24.0,
        18.0,19.5,21.0,22.5,24.0,27.0
     };
```

6. Напишите программу, которая загружает из указанного файла данные в двумерный массив дробного типа и показывает его пользователю.
7. Напишите программу, которая позволяет просматривать текстовые файлы (выводит на экран содержимое файла), например, файлы исходных программ C++. Имя просматриваемого файла должно передаваться программе в качестве параметра, в командной строке во время ее запуска.
8. Напишите программу (модифицируйте представленный пример), которая позволяет за один сеанс работы добавить информацию о нескольких людях в файл D:\phone.txt. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
Добавление в телефонный справочник.
Для завершения вместо ввода фамилии нажмите <Enter>
Фамилия -> Сидоров
Имя -> Вася
Телефон -> 234-84-37
Информация добавлена.
Фамилия -> Орлов
Имя -> Андрей
Телефон -> 552-18-40
Информация добавлена.
Фамилия ->
Ввод  завершен
Для  завершения работы нажмите  <Enter>
```

9. Напишите программу (модифицируйте представленный пример), которая позволяет найти в телефонном справочнике (D:\phone.txt) нужные сведения. Программа должна запрашивать фамилию человека и выводить его телефон. Если в справочнике есть люди с одинаковыми фамилиями, то' программа должна вывести список всех этих людей. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
Поиск в телефонном справочнике.
Введите фамилию и нажмите <Enter>. Для завершения работы с
программой сразу после приглашения нажмите <Enter>
-> Петров
В справочнике данных о Петров нет.
-> Иванов
```

```
Иванов Вася 578-12-45
Иванов Сергей 244-34-02
->
```

10. Напишите программу (модифицируйте представленный пример), которая позволяет удалить в телефонном справочнике (D:\phone.txt) нужные сведения. Программа должна выводить данные из файла с нумерацией строк записей и предлагать выбрать номер записи для удаления. Рекомендуемый вид экрана во время работы программы приведен ниже.

```
Данные из файла:
1. Иванов Вася 578-12-45
2. Сидоров Сергей 254-22-02
3. Иванов Иван 898-12-45
4. Петров Петр 255-34-12
Выберите номер записи для удаления ->3
Запись номер 3 удалена.
Данные из файла:
1. Иванов Вася 578-12-45
2. Сидоров Сергей 254-22-02
3. Петров Петр 255-34-12
Нажмите <Enter> для завершения
```

11. Для выполнения перезаписи в файл содержимого его необходимо стереть полностью. Для этого используйте следующую последовательность вызовов (см. подробнее стр. 370):

```
#include "stdio.h"
#include "conio.h"
#include <sys\stat.h>
#include <sys\file.h>
#define FNAME "phones.txt\0" // имя файла
void main()
{
  char fname[20] = FNAME;
  int fd;
…
  //открытие файла со стиранием его содержимого
  fd = open(fname,O_RDWR|O_CREAT|O_TRUNC, S_IWRITE);
  close(fd);
}
```

**Требования к отчетам по лабораторной работе №3**

Отчет должен содержать описание решения всех индивидуальных заданий в следующем виде:

Текст задания;

Блок схема выполнения программы;

Код программы;

Текст результата выполнения.

Отчеты принимаются либо в электронном виде в документе не старше Word 2003.

Либо в рукописном варианте.