## ЛАБОРАТОРНАЯ РАБОТА № 31

## ЗАПИСЬ И ЧТЕНИЕ БЛОКОВ ДАННЫХ ИЗ ФАЙЛА

**Цель работы:** получение практических навыков по записи и чтению блоков данных из файла.

### Краткие теоретические сведения

Специальные функции обмена с файлами имеются только для символьного и целого типов данных. В общем случае используются функции чтения и записи блоков данных. С их помощью можно записывать в файл и читать из файла вещественные числа, массивы, строки, структуры. При этом сохраняется форма внутреннего представления данных.

Функция записи блока данных имеет прототип

```
int  fread(void*buf,  int bytes,  int n,  FILE*fptr);
```

Здесь

buf – указатель на адрес данных, записываемых в файл;

bytes – длина в байтах одной единицы записи (блока данных);

n – число блоков, передаваемых в файл;

fptr – указатель на поток.

Если запись выполнилась благополучно, то функция возвращает число записанных блоков (значение n).

Функция чтения блока данных из файла имеет прототип

```
int  fwrite(void*buf,  int bytes,  int n,  FILE*fptr);
```

**Пример 4.** следующая программа организует запись блоков в файл строки (символьного массива), а также чтение и вывод на экран записанной информации.

```
# include <stdio.h>
# include <string.h>
void main()
{ FILE  *stream;
```

```
char msg[ ]=”this is a test”;
char buf[20];
if (( stream=fopen(“DUMMY.FILL”,”w+”))==NULL)
 {puts(“Не могу открыть файл \n”); return;}
// Запись строки в файл
fwrite(msg, strlen(msg)+1, 1, stream);
// Установка указателя на начало файла
fseek(stream, 0, SEEK_SET);
// Чтение строки из файла
fread(buf, strlen(msg)+1, 1, stream);
printf(“%s \n”,buf);
fclose(stream);
}
```

В этой программе поток открывается в режиме w+ (создание для записи с последующим чтением). Поэтому закрывать файл после записи не потребовалось. Новым элементом данной программы по сравнению с предыдущими является использование функции установки указателя потока в заданную позицию. Ее формат

```
int  fseek(указатель_на поток, смещение, начало_отсчета);
```

*Начало отсчета* задается одной из констант, определенных в файле stdio.h:

SEEK_SET (имеет значение 0) – начало файла;

SEEK_CUR (имеет значение 1) – текущая позиция;

SEEK_END (имеет значение 2) – конец файла.

*Смещение* определяет число байт, на которое надо сместить указатель относительно заданного начала отсчета. Смещение может быть как положительным, так и отрицательным числом. Оба параметра имеют тип long.

**Форматный обмен с файлами**

С помощью функции форматного вывода можно формировать на диске текстовый файл с результатами вычислений, представленными в символьном виде. В дальнейшем этот файл может быть просмотрен на экране, распечатан на принтере, отредактирован с помощью текстового редактора. Общий вид функции форматного вывода:

int fprintf (указатель_на_поток, форматная_строка, список_переменных);

Использовавшаяся нами ранее функция printf () для организации вывода на экран является частным вариантом функции fprintf (). Функция printf () работает лишь со стандартным потоком stdin, который всегда связывается системой с дисплеем. Не будет ошибкой, если в программе вместо printf () написать fprintf (stdin, …).

Правила использования спецификаторов форматов при записи в файлы на диске точно такие же, как и при выводе на экран.

**Пример 5.** Составим программу, по которой будет рассчитана и записана в файл таблица квадратных корней для целых чисел от 1 до 10. Для контроля эта же таблица выводится на экран.

```
//Таблица квадратных корней
#include <stdio.h>
#include <iostream.h>
#include <math.h>
void main()
{  FILE *fp;
   int x;
   fp = fopen("test.dat", "w");
   //Вывод на экран и в файл шапки таблицы
   printf("\t Таблица квадратных корней \n");
   fprintf(fp, "\t Таблица квадратных корней \n ");
   printf("\t  x\t\tsqrt(x) \n");
   fprintf(fp, "\t  x\t\tsqrt(x) \n ");
   \\Вычисление и вывод таблицы квадратных корней
    \\на экран и в файл
    for(x = 1; x<=10; x++)
    {   printf("\t%f\t%f\n", float(x), sqrt(float(x)));
        fprintf(fp, "\t%f\t%f\n", float(x), sqrt(float(x)));
     }
     fclose(fp);         }
```

Форматный ввод из текстового файла осуществляется с помощью функции fscanf (), общий формат которой выглядит следующим образом:

int fscanf(указатель_на_поток, форматная_строка, список_адресов_переменных);

Данной функцией удобно пользоваться в тех случаях, когда исходные данные заранее подготавливаются в текстовом файле.

В следующем примере числовые данные из файла test.dat, полученного в результате выполнения предыдущей программы, вводятся в числовые массивы x и y. Для контроля значения элементов массивов выводятся на экран. Предварительно с помощью текстового редактора в файле test.dat удаляются две первые строки с заголовками. В результате в файле останутся только числа.

**Пример 6.**

```
\\Ввод  чисел из файла
#include <stdio.h>
#include <iostream.h>
#include <math.h>
void main()
{  FILE *fp;
   int i;
   float x [10], y [10];
   fp = fopen("test.dat","r");
   for (i = 0; i<10; i++)
   {  fscanf(fp, "%f%f", &x[i], &y[i]);
     printf("%f  %f\n", x[i], y[i]);
    }
      fclose(fp);
}
```

### Порядок выполнения работы

1. Изучить теоретические сведения.
2. Выполнить задание.

### Задания для выполнения

1. Набрать и выполнить примеры.
2. Написать аналогичные программы для таблицы умножения целых чисел от 2 до 10.