## ЛАБОРАТОРНАЯ РАБОТА № 30

## СОЗДАНИЕ ФАЙЛА. ЗАПИСЬ И ЧТЕНИЕ ДАННЫХ

**Цель работы:** получение практических навыков по созданию файлов, записи и чтения данных.

### Краткие теоретические сведения

Аналогом понятия внутреннего файла в языке СИ является понятие потока. Потоку в СИ не ставится в соответствие тип. *Поток – это байтовая последовательность, передаваемая в процессе ввода-вывода.*

Поток должен быть связан с каким-либо внешним устройством или файлом на диске.

Основные отличия файлов в СИ состоят в следующем: здесь отсутствует понятие типа файла и, следовательно, фиксированной структуры записи файла. Любой файл рассматривается как байтовая последовательность:

| | | | | |
|---|---|---|---|---|
| байт 0 | байт 1 | байт 2 | … | EOF |

EOF – стандартная константа, обозначающая признак конца файла.

Существуют стандартные потоки и потоки, объявляемые в программе. Последние обычно связываются с файлами на диске, создаваемыми программистами. Стандартные потоки назначаются и открываются системой автоматически. С началом работы любой программы открываются 5 стандартных потоков, из которых основными являются следующие:

- stdin – поток стандартного ввода (обычно связан с клавиатурой);
- stdout – поток стандартного вывода (обычно связан с дисплеем);

- stderr – вывод сообщений об ошибках (связан с дисплеем).

Поток для работы с дисковым файлом должен быть открыт в программе.

Работа с дисковым файлом начинается с объявления указателя на поток. Формат такого объявления:

FILE *имя_указателя;

Например, FILE *fp;

Слово FILE является стандартным именем структурного типа, объявленного в заголовочном файле stdio.h. В структуре FILE содержится информация, с помощью которой ведется работа с потоком, в частности: указатель на буфер, указатель (индикатор) текущей позиции в потоке и т.д.

Следующий шаг – открытие потока, которое производится с помощью стандартной функции **fopen()**. Эта функция возвращает конкретное значение для указателя на поток и поэтому ее значение присваивается объявленному ранее указателю. Соответствующий оператор имеет формат:

имя_указателя=fopen(имя_файла, режим_открытия);

Параметры функции fopen() являются строками, которые могут быть как константами, так и указателями на символьные массивы. Например,

fp=fopen(“test.dat”,”r”);

Здесь test.dat – это имя физического файла в текущем каталоге диска, с котрым теперь будет связан поток с указателем fp. Параметр режима r означает, что файл открыт для чтения.

Существуют следующие режимы открытия файла и соответсвующие им парметры:

| *Параметр* | *Режим* |
|---|---|
| r | открыть для чтения |
| w | создать для записи |
| a | открыть для добавления |
| r+ | открыть для чтения и записи |
| w+ | создать для чтения и записи |
| a+ | открыть для добавления или создать для чтения и записи |

Открытие уже существующего файла для записи ведет к потере прежней информации в нем. Если такой файл еще не су-

ществовал, то он создастся. Открывать для чтения можно только существующий файл.

Поток может быть открыт либо для текстового, либо для двоичного режима обмена.

Смысл понятия текстового файла: это последовательность символов, которая делится на строки специальными кодами – возврат каретки (код 13) и перевод строки (код 10). Если файл открыт в текстовом режиме, то при чтении из такого файла комбинация символов «возврат каретки – перевод строки» преобразуется в один символ \n – переход к новой строке.

При записи в файл осуществляется обратное преобразование. При работе с двоичным файлом никаких преобразований символов не происходит, т.е. информация переносится без всяких изменений.

Указанные выше параметры режимов открывают текстовые файлы. Если требуется указать на двоичный файл, то к параметру добавляется буква b. Например, rb, wb или r+b. В некоторых компиляторах текстовый режим обмена обозначается буквой t, т.е. записывается a+t или rt.

Если при открытии потока по какой-либо причине возникла ошибка, то функция fopen() возвращает значение константы NULL. Эта константа также определена в файле stdio.h. ошибка может возникнуть из-за отсутствия открываемого файла на диске, нехватки места в динамической памяти и т.п. Поэтому желательно контролировать правильность прохождения процедуры открытия файла:

```
FILE  *fp;
If (fp=fopen(“test.dat”,”r”)==NULL
  { puts(“Не могу открыть файл\n”);
    return; }
```

В случае ошибки программа завершит выполнение с закрытием всех ранее открытых файлов.

Закрытие файла осуществляет функция fclose(), прототип которой имеет вид:

```
int  fclose(FILE  *fptr);
```

Здесь fptr обозначает формальное имя указателя на закрываемый поток. Функция возвращает ноль, если операция закрытия прошла успешно. Другая величина означает ошибку.

**Запись и чтение символов**

Запись символов в поток производится функцией putc() с прототипом

int putc(int ch, FILE *fptr);

Если операция прошла успешно, то возвращается записанный символ. В случае ошибки возвращается константа EOF.

Считывание символа из потока, открытого для чтения, производится функцией gets() с прототипом

int gets(FILE *fptr);

Функция возвращает значение считываемого из файла символа. Если достигнут конец файла, то возвращается значение EOF. Это происходит лишь в результате чтения кода EOF.

Функция gets() возвращает значение типа int. То же самое можно сказать и про аргумент ch в описании функции puts(). Используется же в обоих случаях только младший байт. Поэтому обмен при обращении может происходить и с переменными типа char.

**Пример 1.** Составить программу записи в файл символьной последовательности, вводимой с клавиатуры. Пусть признаком завершения ввода будет символ *.

```
//Запись символов в файл
# include <stdio.h>
void main()
{ FILE *fp;   char c;
if (( fp=fopen("test.dat","w"))==NULL)
 {puts("Не могу открыть файл! \n"); return;}
puts("Вводите символы. Признак конца – *");
while ((c=getchar())!='*')   putc(c,fp);
fclose(fp);  }
```

В результате на диске (в каталоге, определяемом системой) будет создан файл с именем test.dat, который заполнится вводимыми символами. Символ * в файл не запишется.

**Пример 2.** Файл требуется последовательно прочитать и содержимое вывести на экран.

```
 //Чтение символов из файла
# include <stdio.h>
# include <conio.h>
void main()
{ FILE  *fp;   char  c;
clrscr();
if (( fp=fopen(“test.dat”,”r”))==NULL)
 {puts(“Не могу открыть файл! \n”); return;}
while ((c=getc(fp))!=EOF)  putchar(c);
fclose(fp);
}
```

**Запись и чтение целых чисел**

Запись целых чисел в поток без преобразования их в символьную форму производится функцией putw() с прототипом

```
int  putw(int, FILE *fptr);
```

Если операция прошла успешно, то возвращается записанное число. В случае ошибки возвращается константа EOF.

Считывание целого числа из потока, открытого для чтения, производится функцией getw() с прототипом

```
int   getw(FILE  *fptr);
```

Функция возвращает значение считываемого из файла числа. Если прочитан конец файла, то возвращается значение EOF.

**Пример 3.** Составить программу записи в файл последовательности целых чисел, вводимых с клавиатуры, чтения их из файла и вывода на экран. Признаком конца ввода будет число 9999.

```
//Запись и чтение целых чисел
# include <stdio.h>
# include <conio.h>
void main()
{ FILE  *fp;    int x;
clrscr();
// Открытие потока для записи
if (( fp=fopen(“test.dat”,”w”))==NULL)
 {puts(“Не могу открыть файл для записи! \n”); return;}
puts(“Вводите числа. Признак конца – 9999”);
```

```
scanf(“%d”,&x);
while (x!=9999)
{putw(x,fp);  scanf(“%d”,&x);}
fclose(fp);    // Закрытие потока в режиме записи
// Открытие потока для чтения
if (( fp=fopen(“test.dat”,”r”))==NULL)
 {puts(“Не могу открыть файл для чтения! \n”); return;}
while ((x=getw(fp))!=EOF)  printf(“\n%d”,x);
fclose(fp);   }
```

После завершения ввода чисел в файл его необходимо закрыть. При этом происходит сброс накопленных в буфере значений на диск. Только после этого можно открывать файл для чтения. Указатель устанавливается на начало потока, и дальнейшее чтение будет происходить от начала до конца файла.

### Порядок выполнения работы

1. Изучить теоретические сведения.
2. Выполнить задание.

### Задания для выполнения

1. Набрать и выполнить примеры.
2. Заполнить файл f последовательного доступа целыми числами, полученными с помощью генератора случайных чисел. Вывести его на экран. Получить в файле g те компоненты f, которые являются четными.