

## C / C++

Язык C не имеет ключевых слов для организации **ввода / вывода**. Вместо них используются библиотечные функции описанные в заголовочном файле `<stdio.h>`. Язык C++ поддерживает две системы **ввода / вывода**: унаследованную от C и новую, объектно-ориентированную систему, включающую как новые функции, так и операторы ввода/вывода (заголовочный файл `<iostream.h>`). Обе системы абсолютно совместимы (в одном исходном файле C++).

### Система **ввода / вывода** языка C.

В C имеются консольные и файловые функции **ввода / вывода**. Консольные функции работают с клавиатурой и экраном (и, на самом деле, со стандартными потоками, которые можно переназначать).

### Основные функции консольного ввода/вывода:

Функция	Действие	Примечания
<code>getchar ();</code>	Читает символ с клавиатуры и отображает на экране	Обычно ожидает Enter
<code>putchar (ch);</code>	Отображает символ на экране	
<code>gets (str);</code>	Читает строку с клавиатуры	
<code>puts (str);</code>	Отображает строку на экране	
<code>printf (“[text]%fmt, ...,a,b,...”);</code>	Осуществляет форматный вывод списка переменных на экран	
<code>scanf (“%fmt, ...”, &amp;a”,&amp;b, ...);</code>	Осуществляет форматный ввод списка переменных с клавиатуры	

C / C++

## Потоки и файлы

**Поток** – это логическое устройство ввода/вывода, являющееся *универсальным интерфейсом* между программой и физическими устройствами. Они называются **файлами**.

В языке C отсутствуют типы файлов, а, следовательно, отсутствует и записеориентированный обмен.

В C / C++ существует только потокоориентированный обмен данными.

Потоки бывают двух видов: **текстовые** и **бинарные (двоичные)**.

**Текстовый поток** – это последовательность символов. Стандарт C позволяет (но не требует!) организовывать потоки в виде строк с символом EOL в конце. При обмене с физическим устройством в текстовом потоке могут происходить определенные преобразования (например, EOL заменяться на CR, LF).

**Бинарный поток** – это последовательность байтов, однозначно соответствующая байтам на внешнем физическом устройстве. Никакого преобразования при обмене не происходит. В конце бинарного потока могут добавляться нулевые байты, например, для полного заполнения сектора на физическом диске.

## C / C++

В языке C / C++ файлами могут быть файлы на дисках и реальные устройства ввода / вывода.

**Последовательный доступ** возможен для всех файлов, а **прямой доступ** – не для всех; для дисковых файлов – возможен, а для большинства принтеров – нет.

Для **прямого доступа** файл должен поддерживать запросы не местоположение текущей позиции. При открытии такого файла указатель текущей позиции в файле устанавливается в начало файла. При чтении/записи символа текущая позиция смещается (к концу файла).

Имена файлов представляются в виде литералов (строковых констант): "e:\\VC\\history.cpp", включающих имя файла и, при необходимости, путь к нему (в кавычках, обратная косая черта –удвоенная).

По умолчанию есть пять **стандартных потоков**, открывающихся при начале работы программы:

- **stdin** – ввод данных (клавиатура),
- **stdout** – вывод данных (экран),
- **stderr** – вывод сообщений об ошибках (экран),
- **stdaux** – дополнительный поток,
- **stdprn** – печать (принтер).

C / C++

## Организация ввода/вывода в файл

1. Определить указатель файла.
2. Открыть поток (связать его с файлом и задать режим обмена).
3. Выполнить операции ввода/вывода.
4. Закрыть файл.

Указатель файла – это указатель на структуру типа **FILE**, содержащую сведения о файле: имя, статус, указатель текущей позиции в начало файла и др. В этой структуре – блоке управления файлом не следует ничего изменять.

Объявление переменной-указателя файла: **FILE \*fp;**

Указатель файла - **fp** используется в дальнейших операциях с потоком. Его передают функциям ввода\вывода в качестве параметра, определяющего поток.

При открытии потока с ним связывается область памяти – **буфер**. При выводе данные накапливаются в буфере до его заполнения (а потом записываются в файл - выгружаются) или до закрытия потока. Чтение из файла производится блоками, размер их равен буферу. При аварийном завершении программы данные в буфере могут пропасть.

C \ C++

## Подпрограммы для работы с файлами

**Функция `fopen (fname, mode)`**; – открывает поток и связывает его с файлом `fname`.  
Возвращает указатель файла или `NULL`.

Здесь `fname` – имя файла – указатель на строку символов (литерал), может включать путь к файлу, а `mode` – режим, который определяет назначение файла.

Напр.: `FILE *fp; fp = fopen ("d:\\BC\\test.txt", "w");`

### Значения параметра режима (mode):

`r (rt)` – открыть текстовый файл для чтения,

`w (wt)` – создать текстовый файл для записи,

`a (wa)` – добавить записи в конец текстового файла,

`rb` – открыть бинарный файл для чтения,

`wb` – создать бинарный файл для записи,

`ab` – добавить записи в конец бинарного файла.

`r+` – открыть текстовый файл для чтения и записи,

`w+` – создать текстовый файл для чтения и записи,

`a+` – добавить записи в конец текстового файла или создать текстовый файл для чтения и записи,

`r+b (rb+)` – открыть бинарный файл для чтения и записи,

`w+b (wb+)` – создать бинарный файл для чтения и записи,

`a+b (ab+)` – добавить записи в конец бинарного файла или создать бинарный файл для чтения и записи.

Файл не существует: `r (rb, r+, rb+)` – работа `fopen()` завершается отказом (`NULL`), `w (...)` – файл создается, а `(...)` – файл создается.

Файл существует: `r (rb, r+, rb+)` – файл открывается, `w (...)` – файл удаляется и открывается снова, а `(...)` – запись в конец файла.

Напр. – правильное открывать файл с проверкой:

```
FILE *fp; if ((fp=fopen (" d:\\BC\\test.txt", "w"))==NULL)  
{ printf("Ошибка при открытии файла\n"); exit(1); }
```

C \ C++

## Подпрограммы для работы с файлами

**Функция `fclose (fp)`**; – явно закрывает поток ввода/вывода `fp`. Возвращает 0 при успешной операции закрытия или EOF в случае ошибки. Файл закрывается автоматически при завершении программы.

Здесь `fp` – указатель файла.

Напр.: `fclose (fp)`; Функция записывает в файл все данные из буфера, а также освобождает блок управления файлом.

**Функция `remove (fname)`**; – удаляет существующий файл. Возвращает 0 при успешной операции закрытия иначе - ненулевое.

Здесь `fname` – имя файла (литерал), может включать путь к файлу.

Напр.: `remove (fp)`; Очищает буфер путем немедленной засылки записываемых данных на физическое устройство (файл).

**Функция `fflush (fp)`**; – записывает данные из буфера, при этом файл остается открытым. Возвращает 0 при успешной операции закрытия или EOF в случае ошибки.

Здесь `fp` – указатель файла.

Напр.: `fflush (fp)`; Очищает буфер путем немедленной засылки записываемых данных на физическое устройство (файл). При вызове с пустым (NULL) указателем файла – производит запись буферов во все открытые файлы.

**Функция `rename(oldfname, newfname)`**; – переименовывает существующий файл или папку. Возвращает 0 при успешной операции закрытия, иначе – ненулевое значение.

Здесь `oldfname` – старое имя файла (литерал), может включать путь к файлу, а `newfname` – новое имя файла.

Напр.: `rename ("test", "text.txt")`; Новое имя файла не должно совпадать ни с одним существующим в данном каталоге.

C \ C++

## Подпрограммы для работы с файлами

Функция **ferror (fp);** – возвращает код ошибки при работе с потоком. Возвращает 0 при отсутствии ошибок, иначе – код ошибки (целое число).

Здесь **fp** – указатель файла.

Напр.: ferror (fp); чтобы определить природу ошибки надо исп. функцию perror().

Функция **perror (s);** – печатает строку вида "[s]: Сообщение об ошибке" на **stderr**.

Где **s** – текстовая строка до Сообщения об ошибке.

Напр.: perror ("Ошибка при работе с файлом"); Сообщение об ошибке – значение глобальной переменной **errno**, преобразованное в строку.

Функция **clearerr (fp);** – обнуляет флаг ошибки для потока **fp**.

Где **fp** – указатель файла.

Напр.: clearerr (fp);

C \ C++

## Подпрограммы для работы с файлами

**Функция `fgetc (fp)`**; – читает символ из потока как значение `unsigned char`, преобразованный в целое (`int`). Здесь `fp` – указатель файла.

Напр.: `fgetc (fp)`: Берется символ находящийся за текущей позицией, а затем указатель текущей позиции увеличивается на 1. Если символ не прочитался, возвращается EOF. Т.к. EOF – действительное целое, то при работе с бинарными файлами надо исп. функцию `feof()`.

**Функция `fgets (s, n, fp)`**; – читает из входного потока `fp` не более `n-1` символов в строку `s`. Возвращает `s` при отсутствии ошибок, иначе – NULL.

Здесь `s` – указатель строки, `n` – количество читаемых символов (начиная с 0), `fp` – указатель файла.

Напр.: `fgets (str, num, fp)`: Чтение завершается при достижении `num-1`, конца строки (EOL) или конца файла (EOF). Прочитанная строка автоматически завершается '\0'. EOL не отбрасывается, а помещается в конец строки `str`.

**Функция `fscanf (fp, fmt..., list...)`**; – читает из потока форматированные данные, как `scanf`. Возвращает число записанных переменных или EOF. Где `fp` – указатель файла, `fmt` - форматы вводимых данных, `list` – список переменных.

Напр.: `fscanf (fp, "%f", &g)`;

**Функция `fread (buf, size, count, fp)`**; – чи-тает из потока `fp` `count` объектов размером `size` и помещает их в массив `buf`. Указатель сдвигается. Возвращает число прочитанных элементов.  
Напр.: `fread (b, sizeof(int), n, fp)`;

C \ C++

## Подпрограммы для работы с файлами

**Функция `fputc (ch, fp);`** – записывает символ `ch` в поток `fp`. Возвращает значение записанного символа или EOF. Здесь `ch` – символ, `fp` – указатель файла.

Напр.: `fputc ('a', fp);`

**Функция `fputs (s, fp);`** – записывает в заданный поток `fp` строку `s`. Возвращает неотрицательное при отсутствии ошибок, иначе – EOF.

Здесь `s` – указатель строки, `fp` – указатель файла.

Напр.: `fputs (str, fp);` Символ `'\0'` не записывается.

**Функция `fprintf (fp, fmt..., list...);`** – записывает в поток форматированные данные, как `printf`. Возвращает число записанных переменных или EOF.

Здесь `fp` – указатель файла, `fmt` - форматы выводимых данных, `list` – список переменных.

Напр.: `fprintf (fp, "%f", g);`

**Функция `fwrite (buf, size, count, fp);`** – записывает в поток `fp` `count` объектов размером `size` из массива символов адресуемого указателем `buf`. Указатель сдвигается. Возвращает число записанных элементов.

Напр.: `fwrite (b, sizeof(int), n, fp);`

C \ C++

## Подпрограммы для работы с файлами

**Функция feof (fp);** – проверяет достигнут ли конец файла (EOF).

Возвращает ненулевое значение когда достигнут EOF или 0, если не достигнут.

Здесь fp – указатель файла.

Напр.: feof (fp); Полезен при работе с бинарными файлами, т.к. маркер конца файла – полноценное двоичное целое.

**Функция fgetpos (fp, pos);** – сохраняет в объекте pos (типа fpos\_t) текущую позицию в файле, связанном с потоком fp. Возвращает 0 при отсутствии ошибок, иначе – ненулевое значение.

Здесь fp – указатель файла, pos – текущая позиция.

Напр.: fgetpos (fp, fpos);

**Функция fsetpos (fp, pos);** – перемещает текущую позицию в файле, связанном с потоком fp, на позицию, заданную указателем pos. Возвращает 0 при успешном завершении и ненулевое значение при ошибке.

Здесь fp – указатель файла, pos – указатель позиции.

Напр.: fscanf (fp, &pos); Значение текущей позиции pos предварительно запрашивается функцией fgetpos().

**Функция rewind (fp);** – перемещает указатель текущей позиции в начало заданного потока fp и очищает флаги ошибок.

Напр.: rewind (fp);

C \C++

## Подпрограммы для работы с файлами

**Функция `fseek (fp, offset, origin)`**; – устанавливает текущую позицию в файле связанном с потоком `fp`, на позицию `offset`, отсчитываемую от `origin`.

Возвращает 0 при успешной операции и ненулевое значение при ошибке.

Здесь `fp` – указатель файла; `origin` – начало отсчета, определяется макросами `SEEK_SET` (начало файла), `SEEK_CUR` (текущая позиция), `SEEK_END` (конец файла); `offset` – смещение текущей позиции от начала отчета (в байтах).

Напр.: `fseek (fp, sizeof(gh), SEEK_CUR)`; Эта функция исп. преимущественно при работе с бинарными файлами. Для текстовых файлов `origin` должен быть всегда `SEEK_SET`, а `offset` – брать значение из функции `ftell()` для потока `fp` или нуль.

**Функция `ftell (fp)`**; – возвращает текущую позицию в файле, связанном с потоком `fp`. При возникновении ошибки функция возвращает -1.

Здесь `fp` – указатель файла.

Напр.: `ftell (fp)`; Для бинарных потоков значение `ftell()` равно количеству байтов от начала файла до текущей позиции. Для текстовых файлов возвращаемое значение используется исключительно как аргумент функции `fseek()`. (Суть дела в возможных преобразованиях в текстовых файлах.)

C \ C++

## Подпрограммы для работы с файлами

**Функция `setbuf (fp, buf);`** – устанавливает буфер ввода/вывода для файла связанного с потоком `fp`, или отключает буферизацию, если `buf = 0`.

Здесь `fp` – указатель файла; `buf` – указатель на буфер.

Напр.: `setbuf (fp, buffer);` Размер буфера, задаваемого программистом, равен константе `BUFSIZ`.

**Функция `setvbuf (fp, buf, mode, size);`** – устанавливает буфер обмена, заданный указателем `buf`, для потока `fp`. При возникновении ошибки функция возвращает `-1`.

Здесь `fp` – указатель файла, `buf` – указатель на массив символов, `size` – размер буфера в байтах, `mode` – режим буферизации (определяется константами как: `_IOFBF`, `_IOLBF`, `_IONBF`).

Напр.: `setvbuf(fp, buffer, _IOFDF, 128);`

**Функция `tmpfile ()`**; – открывает временный двоичный файл для ввода/вывода и возвращает указатель на связанный с файлом поток. При возникновении ошибки функция возвращает `0`.

Напр.: `FILE *tmpfile (void);` Автоматически исп. уникальное имя файла. Временный файл автоматически удаляется при закрытии файла или завершении программы. Количество временных файлов определяется константой `TMP_MAX` (предел устанавливается константой `FOPEN_MAX`).

**Функция `tmpnam (name);`** – создает уникальное имя для временного файла. При возникновении ошибки функция возвращает `0`, иначе указатель на массив имен файлов..

Здесь `fp` – указатель файла,

Напр.: `tmpnam (fnam);`

C \ C++

## Подпрограммы для работы с файлами

Функция **freopen (fname, mode, fp);** - связывает существующий поток с другим файлом - **fname**. Возвращает указатель файла или NULL (при ошибке). Если поток **fp** был открыт, то он закрывается и создается новый поток.

Здесь **fname** – новое имя файла, **mode** – режим, который определяет назначение файла, **fp** – указатель закрываемого файла.

Напр.: freopen ('d', w+, p); Эту функцию можно использовать, в частности, для перенаправления потоков.

Функция **ungetc (ch, fp);** – возвращает символ **ch** в поток **fp**.

Здесь **fp** – указатель файла; **ch** – возвращаемый символ.

При возникновении ошибки функция возвращает значение EOF, при отсутствии ошибок - **ch**.

Напр.: ungetc ('d', fp); Вызов функций **fflush()**, **fseek()**, **rewind()** аннулирует действие **ungetc()** и сбрасывает символ (**ch**). Попытка вернуть в поток ввода EOF игнорируется. Обращение к функции **ungetc()** очищает признак конца файла, связанного с данным потоком. Значение указателя текущей позиции файла для текстового потока до тех пор, пока не прочитаны все возвращаемые символы, остается таким, как до первого вызова **ungetc()**. При работе с бинарными файлами каждый вызов функции уменьшает указатель текущей позиции файла.

Функции **vprintf**, **vfprintf** – их действие аналогично функциям **printf** и **fprintf**, но список аргументов заменяется указателем на список аргументов.

Функции **vscanf**, **vfscanf** – их действие аналогично функциям **scanf** и **fscanf**, но список аргументов заменяется указателем на список аргументов.

## Стандартные потоки

Открываются автоматически в начале выполнения программы на C / C++ стандартные потоки можно перенаправить:

```
freopen ("output.txt", "w+", stdout);
```

C \C++

## Подпрограммы для работы с файлами

Функции ввода/вывода языка C собраны в библиотеке (заголовке) `<stdio.h>`

### Некоторые типы данных, макросы и константы заголовка `<stdio.h>`

Тип данных `FILE` – содержат информацию для работы с файлом.

Структура этого типа зависит от конкретной реализации, приведем один из вариантов:

```
typedef struct {
    int level;           // флаг состояния буфера
    unsigned flag;      // флаг состояния файла
    char fd;            // префикс файла
    unsigned char hold; // переданный символ
    int bsize;          // размер внутреннего буфера
    unsigned char_FAR *buffer; /* значение указателя для доступа внутрь буфера, задает (в зависимости от
    режима буферизации) начало буфера, начало строки или текущее значение указателя внутри буфера */
    unsigned char_FAR *curp; /* текущее значение указателя для доступа внутрь буфера, задает текущую
    позицию в буфере для обмена с программой */
    unsigned istemp;     // флаг временного файла
    short token;         // маркер действительности файла
} FILE;
```

Тип данных `fpos_t` – используется функциями `fgetpos()` и `fsetpos()` для хранения текущей позиции в файле (целое без знака).

Тип данных `size_t` – используется для представления результат операции `sizeof` (целое без знака. не длиннее `unsigned long`).

Константа `BUFSIZ` – используется для задания в байтах размера буфера потока данных функциями `fopen()`, `freopen()`, `setbuf()`. При открытии потока данных к нему автоматически присоединяется буфер длиной `BUFSIZ`. Минимальный размер `BUFSIZ` – 255 байт.

Константа `EOF` – используется для сообщения об окончании файла (отрицательное целое число).

C \ C++

## Подпрограммы для работы с файлами

Функции ввода/вывода языка C собраны в библиотеке (заголовке) `<stdio.h>`

### Некоторые типы данных, макросы и константы заголовка `<stdio.h>`

Константа `FOPEN_MAX` – задает максимальное число открытых потоков, оно должно быть  $\leq 8$ , при этом `stdin`, `stdout` и `stderr` – открывающиеся автоматически входят в это число. Программы, открывающие более 5 потоков одновременно, должны проверить доступно ли это количество потоков.

Константы `_IOFBF`, `_IOLBF`, `_IONBF` – применяются для определения режима буферизации в функции `setvbuf()`. Где `_IOFBF` – 0 – полная буферизация, `_IOLBF` – 1 – построчная буферизация текстового файла, `_IONBF` – 2 – отмена буферизации.

Константа `L_tmpnam` – задает минимальную длину строки имени файла, генерируемого функцией `tmpnam()`.

Константы `SEEK_SET`, `SEEK_CUR`, `SEEK_END` – это аргументы функции `fseek()`. Здесь `SEEK_SET` – 0 – сдвиг выполняется от начала файла, `SEEK_CUR` – 1 – сдвиг выполняется от текущей позиции указателя записи/чтения файла, `SEEK_END` – 2 – сдвиг выполняется от конца файла.

Константа `TMP_MAX` – задает максимальное количество различных временных имен файла, генерируемых функцией `tmpnam()` в течение одного сеанса работы программы. Значение этой константы не может быть меньше 25.

# ФАЙЛЫ

C \C++

## Практические занятия

**Задание 1. Создать на диске Z:\ файл numbers.txt, записать в него 5 введенных с клавиатуры целых чисел.**

*/\* Создает на диске Z:\ файл numbers.txt и записывает в него 5 целых чисел введенных пользователем с клавиатуры \*/*

```
#include <stdio.h>
```

```
#define FNAME "e:\\edu2\\numbers.txt\0" // имя создаваемого файл
```

```
#define N 5 // количество вводимых в файл чисел
```

```
void main()
```

```
{ char fname[20] = FNAME;
```

```
FILE *out; // файл чисел
```

```
int n; // вводимое число
```

```
int i; // счетчик цикла for
```

```
puts ("\nСоздание файла");
```

```
// Открыть файлы режиме записи текста - wt
```

```
if (( out = fopen(fname, "wt")) == NULL)
```

```
{ printf ("Ошибка при открытии файла для записи");
```

```
getch();
```

```
return; }
```

```
printf ("Вводимые числа записываются в файл");
```

```
printf ("%s\n", fname);
```

```
puts ("Введите 5 целых чисел, нажимая Enter после каждого числа");
```

```
for (i = 0; i < N; i++)
```

```
{ printf ("->");
```

```
scanf ("%i", &n); // читать числа с клавиатуры
```

```
fprintf (out, "%i\n", n); // записать числа в файл }
```

```
fclose (out); // закрыть файл
```

```
printf ("Введенные числа записаны в файл %s\n", fname);
```

```
puts ("\nДля завершения нажмите Enter");
```

```
}
```

# ФАЙЛЫ

**C \C++**

## Практические занятия

**Задание 2. Вывести на экран содержимое файла numbers.txt.**

```
// Выводит на экран файл z:\numbers.txt
#include <stdio.h>
// имя создаваемого файл
#define FNAME "e:\\edu2\\numbers.txt\0"

void main()
{ char fname[20] = FNAME;
  FILE *in;    // файл чисел
  char st[80]; // строка из файла
  printf ("\nСодержимое файла %s\n", fname);
  puts ("-----");
// Открыть файл в режиме чтения текста - rt
  if (( in = fopen(fname, "rt")) == NULL)
    { printf ("Ошибка при открытии файла для чтения");
      getch();
      return; }
  while (!feof(in))
    { fscanf (in, "%s", &st); // читать число из файла
      printf ("%s\n", st);    // вывести число на экран
    }
  fclose (in); // закрыть файл
  puts ("-----");
  puts ("\nДля завершения нажмите Enter");
}
```