НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Файлы

- Что такое файл?
- Файл – именованный набор байтов, который может быть сохранен на некотором накопителе.
- Другими словами, под файлом понимается последовательность байтов, записанных на диск, которая имеет своё, уникальное имя, например **`file.txt`**.

Файлы

- В одном каталоге не могут находиться файлы с одинаковыми именами.
- Под именем файла понимается не только его название, но и расширение, например: file.txt и file.dat - разные файлы, хоть и имеют одинаковые названия.
- Существует такое понятие, как полное имя файлов – это полный путь к каталогу файла с указанием имени файла, например: **`D:\docs\file.txt`**.

Файлы

- Для работы с файлами необходимо подключить заголовочный файл <fstream>.
- В <fstream> определены несколько классов и подключены заголовочные файлы
  - <ifstream> - файловый ввод и
  - <ofstream> - файловый вывод.

Файлы

- Файловый ввод/вывод аналогичен стандартному вводу/выводу, единственное отличие – это то, что ввод/вывод выполнятся не на экран, а в файл.
- Если ввод/вывод на стандартные устройства выполняется с помощью объектов cin и cout, то для организации файлового ввода/вывода достаточно создать собственные объекты, которые можно использовать аналогично тому, как использовались cin и cout.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Файлы

- Например, необходимо создать текстовый файл и записать в него строку ”Работа с файлами в C++”.
- Для этого необходимо проделать следующие шаги:

1. создать объект класса ofstream;
2. связать объект класса с файлом, в который будет производиться запись;
3. записать строку в файл;
4. закрыть файл.

Файлы

```
// создаём объект для записи в файл
ofstream /*имя объекта*/; // объект класса ofstream
```

Назовём объект – fout:

```
ofstream fout;
```

Для чего нужен объект?

- Объект необходим, чтобы можно было выполнять запись в файл.
- Уже объект создан, но не связан с файлом, в который нужно записать строку.

```
fout.open("example.txt");
// связываем объект с файлом
```

Файлы

```
ofstream fout;
fout.open("example.txt");
```

- Через операцию **точка** получаем доступ к методу класса open(), в круглых скобочках которого указываем имя файла.
- Указанный файл будет создан в текущей директории с программой.
- Если файл с таким именем существует, то существующий файл будет заменен новым.

```
fout << "Работа с файлами в C++";
// запись строки в файл
```

Используя операцию << с объектом fout, строка “Работа с файлами в C++” записывается в файл.

Файлы

```
ofstream fout;
fout.open("example.txt");
fout << "Работа с файлами в C++";
```

Так как больше нет необходимости изменять содержимое файла, его нужно закрыть, то есть отделить объект от файла.

```
fout.close(); // закрываем файл
```

Итог – создан файл со строкой “Работа с файлами в C++”.

Шаги 1 и 2 можно объединить, то есть в одной строке создать объект и связать его с файлом:

```
ofstream fout("example.txt");
```

Файлы

В итоге получим такую программу:

```
#include <fstream>
using namespace std;

int main()
{
    ofstream fout("example.txt");
    fout << "Работа с файлами в C++";
    fout.close();
    return 0;
}
```

Осталось проверить правильность работы программы, а для этого открываем файл example.txt:

```
Работа с файлами в C++
```

Файлы

Для того чтобы прочитать файл понадобится выполнить те же шаги, что и при записи в файл с небольшими изменениями:

- создать объект класса **ifstream** и связать его с файлом, в который будет производиться запись;
- прочитать файл;
- закрыть файл.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Файлы

```
#include <fstream>
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleCP(1251); SetConsoleOutputCP(1251);
    char buff[50];
    ifstream fin("example.txt");
    fin >> buff;
    cout << buff << endl;
    fin.getline(buff, 50);
    fin.close();
    cout << buff << endl;
    return 0;
}
```

Файлы

- Программа сработала правильно, но не всегда так бывает, даже в том случае, если с кодом всё в порядке.
- Например, в программу передано имя несуществующего файла или в имени допущена ошибка.
- В этом случае ничего не произойдёт вообще.
- Файл не будет найден, а значит и прочитать его не возможно.
- Поэтому компилятор проигнорирует строки, где выполняется работа с файлом.
- В результате корректно завершится работа программы, но на экране ничего показано не будет.

Файлы

- Простому пользователю не будет понятно, в чём дело и почему на экране не появилась строка из файла.
- Чтобы отреагировать в такой ситуации, в C++ предусмотрена специальная функция - is_open(), которая возвращает целые значения:
  - 1 — если файл был успешно открыт,
  - 0 — если файл открыт не был.
- Доработаем программу с открытием файла, таким образом, что если файл не открыт выводилось соответствующее сообщение.

Файлы

```
#include <fstream>
#include <iostream>
#include <windows.h>

using namespace std;

int main()
{
    SetConsoleCP(1251); SetConsoleOutputCP(1251);
    char buff[50];
    ifstream fin("example.txt");
    if (!fin.is_open()) // если файл не открыт
        cout << "Файл не может быть открыт!\n";
    else {
        fin >> buff;
        cout << buff << endl;
        fin.getline(buff, 50);
        fin.close();
        cout << buff << endl;
    }
    return 0;
}
```

Режимы открытия файлов

Константа	Описание
ios_base::in	открыть файл для чтения
ios_base::out	открыть файл для записи
ios_base::ate	при открытии переместить указатель в конец файла
ios_base::app	открыть файл для записи в конец файла
ios_base::trunc	удалить содержимое файла, если он существует
ios_base::binary	открытие файла в двоичном режиме

Режимы открытия файлов

Режимы открытия файлов можно устанавливать при создании объекта или при вызове функции open()

```
// открываем файл для добавления информации
// в конец файла
ofstream fout("example.txt", ios_base::app);
fout.open("example.txt", ios_base::app);
```

Режимы открытия файлов можно комбинировать с помощью поразрядной логической операции «или» - |, например: `ios_base::out | ios_base::trunc` - открытие файла для записи, предварительно очистив его.

Режимы по умолчанию

- Объекты класса **ofstream**, при связке с файлами по умолчанию содержат режимы открытия файлов **ios_base::out** | **ios_base::trunc**.
- То есть файл будет создан, если не существует.
- Если же файл существует, то его содержимое будет удалено, а сам файл будет готов к записи.

- Объекты класса **ifstream** связываясь с файлом, имеют по умолчанию режим открытия файла **ios_base::in** - файл открыт только для чтения.
- Режим открытия файла ещё называют — “флаг”.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Синтаксис С

- Для управления потоками используются потоковый указатель , который является указателем на структуру, содержащую информацию о соответствующем файле.

```
FILE *<имя потокового указателя>
```

где <имя потокового указателя> - это имя указателя на структуру FILE.
Описание типа FILE дается в файле stdio.h;

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Синтаксис С

Структура определения файла для потока, определенная в stdio.h:

```
typedef struct
 {
    short level;
    usisigned flag;
    . . .
 } FILE;
```

В определении типа FILE указывается, например, адрес буфера, с которым будет работать данный файл; адрес текущего символа, который будет передаваться (приниматься) в прикладную программу.

Отметим среди этих описаний еще следующие:

```
#define EOF -1 /*Константа, возвращаемая при обнаружении конца
                 файла*/
               /*Для текстовых файлов EOF есть код символа [Ctr/Z] */
#define NULL 0 /*Константа с нулевым значением, часто используется
как   значение пустого указателя. Не путать с нулевым знаком  '\0'*/
```

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Открытие файла (потока)

FILE *fopen(const char *физ_имя_файла, const char *режим);
Режим указывает на строку содержащую режим открытия файла.
Возможны следующие режимы:

режим	Действие
“r”	Открывает файл для чтения
“w”	Создает файл для записи
“a”	Открывает файл для дозаписи в конец файла
“rb”	Открывает двоичный файл для чтения
“wb”	Открывает двоичный файл для записи
“ab”	Открывает двоичный файл для дозаписи в конец файла
“r+”	Открывает текстовый файл для чтения/записи
“w+”	Создает текстовый файл для чтения/записи
“a+”	Открывает или создает текстовый файл для чтения/записи

Открытие файла (потока)

“r+b” Открывает двоичный файл для чтения/записи
“w+b” Открывает двоичный файл для чтения/записи
“a+b” Открывает или создает двоичный файл для чтения/записи
“rt” Открывает текстовый файл для чтения
“wt” Открывает текстовый файл для записи
“at” Открывает текстовый файл для дозаписи в конец файла
“r+t” Открывает текстовый файл для чтения
“w+t” Открывает текстовый файл для чтения/записи
“a+t” Открывает или создает текстовый файл для чтения/записи

В случае если спецификация файла задана неверно, то fopen() возвращает указатель NULL.
Функция fclose() разрывает связь указателя с файлом и закрывает его к дальнейшему использованию, до тех пор пока он вновь не будет используя функцию fopen().

Пример. Открытия файла с именем rezult.txt для записи.

```
FILE *fptr;  /* Это - определение потока. fptr - указатель на FILE */
fptr = fopen("rezult.txt","w");
if (fptr = =NULL)
{ /* Файл не открылся. Напечатать его имя и режим обработки и аварийно
завершить программу */
   }
else
 { /* Файл открылся. Провести обработку файла */
}
```

Очистка потока

Прототип функции fflush (в stdio.h):

int fflush(FILE *stream);

Функция fflush вызывает запись в stream содержимого буферов, связанных с открытыми потоками вывода, и чистит содержимое буфера, если stream - открытый поток ввода (чаще всего для очистки буфера ввода и используется). Поток stream остается открытым.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Очистка потока

Пример.

```
scanf(“%d”, &n);
fflush(stdin);      /* Очистка вх. потока stdin в случае неправильного ввода */
scanf (“%d”, &i);
fflush(stdin); /* Очистка вх. потока stdin в случае неправильного ввода и для последующего нормального выполнения функции gets (надо "убрать" символ конец строки, оставшийся от scanf, чтобы gets нормально проработала). */
gets (str);
```

Чтение символа из потока

Чтение. Прототип функции:

```
int fgetc(FILE *stream);
```

Прототип макроса:

```
int getc(FILE *stream);
```

Вводит следующий символ из потока, указанного в stream. В случае успеха fgetc и getc возвращают прочитанный символ после преобразования его в целые без знака. При конце файла или ошибке возвращают EOF.

запись символа в поток

Запись. Прототип функции:

int fputc(int ch, FILE *sream);

Прототип макроса:

int putc(int ch, FILE *stream);

Выводит заданный символ ch в заданный поток stream. В случае успеха fputc и putc возвращают символ ch, в случае ошибки - EOF.

По действию соответствующие функции и макросы эквивалентны. Функция работает медленнее, чем макрос, но на каждый вызов требует меньше памяти. Кроме того, существуют ситуации (например, при работе с указателями), когда необходимо использовать именно функцию, а не макрос. Так, нельзя использовать функцию, указывающую на макрос.

запись символа в поток

Запись. Прототип функции:

int fputc(int ch, FILE *sream);

Прототип макроса:

int putc(int ch, FILE *stream);

Выводит заданный символ ch в заданный поток stream. В случае успеха fputc и putc возвращают символ ch, в случае ошибки - EOF.

По действию соответствующие функции и макросы эквивалентны. Функция работает медленнее, чем макрос, но на каждый вызов требует меньше памяти. Кроме того, существуют ситуации (например, при работе с указателями), когда необходимо использовать именно функцию, а не макрос. Так, нельзя использовать функцию, указывающую на макрос.

Форматированный ввод (вывод) в (из) потока

Ввод. Прототип функции:

```
int  fscanf(FILE *stream, char * format [, argument, . . .] );
```

Вывод. Прототип функции:

```
int  fprintf(FILE *stream, char * format [, argument, . . .] );
```

Функция fscanf читает данные из заданного входного потока. Функция fprintf записывает данные в заданный выходной поток. Имя потока задано параметром stream. Параметр format задает строку формата, в соответствии с которой производится форматированный ввод или вывод переменного числа величин. Функция fscanf возвращает количество успешно введенных и сохраненных входных аргументов. Если при чтении входного потока функция встретила конец файла, то возвращает EOF. Функция fprintf возвращает количество выведенных байт. В случае ошибки возвращает EOF.

Чтение (запись) строки из(в) потока

Чтение. Прототип функции:

```
char  *fgets(char *string, int n, FILE *stream);
```

Запись. Прототип функции:

```
char  *fputs(char *string, FILE *stream);
```

Функция fgets читает символы из потока stream в строку string. Функция заканчивает чтение, когда она либо прочтет n-1 символ, либо встретит символ новой строки. Последним символом, записанным в string, будет нулевой символ. Возвращаемое значение: при успехе - строка string, переданная как аргумент, а при ошибке или конце файла - NULL.

Функция fputs копирует строку string, оканчивающуюся нулевым символом, в выходной поток stream; символ новой строки не добавляется. Возвращаемое значение: при успехе - последний записанный символ; в противном случае - EOF.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ КОРАБЛЕБУДУВАННЯ ІМЕНІ АДМІРАЛА МАКАРОВА

Установка указателя файла в потоке

Прототипы функций:

```
int  fseek(FILE *stream, long offset, int fromwhere);
long rewind(FILE *stream);
long ftell(FILE *stream);
```

Функция fseek устанавливает указатель файла, связанного со stream, на новую позицию, которая отстоит от места в файле, заданного параметром fromwhere, на количество байт, указанных в offset (с учетом знака).

Параметр fromwhere должен быть одной из величин: 0, 1, 2. Эти величины можно представлять тремя константами, определенными в stdio.h (см. таблицу ниже).

Установка указателя файла в потоке

Fromwhere		Размещение в файле
Имя константы	значение	
SEEK_SET	0	Начало файла
SEEK_CUR	1	Текущее положение указателя файла
SEEK_END	2	Конец файла

Функция rewind устанавливает указатель на начало файла, то есть вызов функции rewind(stream) эквивалентен по своему действию вызову функции fseek(stream, 0L, 0).

После вызова функции fseek или rewind следующей операцией с файлом, открытым для модификации, может быть как ввод, так и вывод.

Возвращаемое значение: fseek и rewind возвращают 0, если указатель успешно перемещен, и ненулевое значение при ошибке.

Функция ftell возвращает текущее положение (смещение) указателя файла, указанного в stream. Смещение измеряется в байтах, считая от начала файла.