Строковый ввод-вывод. Работа с текстовыми файлами

Выполнил Полудницын Константин
РИС-17-1б

Раздел 1. *Ввод-вывод текстовых файлов. Работа с файлами. Хранение текстовой информации на внешних носителях*

- Файлы позволяют пользователю считывать большие объемы данных непосредственно с диска, не вводя их с клавиатуры.
- *Текстовыми* называются файлы, состоящие из любых символов. Они организуются по строкам, каждая из которых заканчивается символом «*конца строки*». Конец самого файла обозначается символом «*конца файла*». При записи информации в текстовый файл, просмотреть который можно с помощью любого текстового редактора, все данные преобразуются к символьному типу и хранятся в символьном виде.

Заголовочный файл и объекты

- Для того, чтобы в C++ работать с файлами, необходимо подключить заголовочный файл **<fstream>.**
- После этого можно объявлять объекты, привязанные к файлам: для чтения данных из файла используются объекты типа **ifstream** (аббревиатура от ***input file stream***), для записи данных в файл используются объекты типа **ofstream** (***output file stream***)
- Например:
- **ifstream in; //Поток in будем использовать для чтения**
- **ofstream out; //Поток out будем использовать для записи**

Открытие и закрытие файла (open and close)

- Open: Открывает файл, идентифицированный именем файла аргумента, связав его с объектом потока, так что операции ввода / вывода выполняются над его содержимым. Режим аргумента указывает режим открытия.
- ! Если поток уже связан с файлом (т. е. он уже открыт), вызов этой функции завершается с ошибкой.
- Close: Закрывает файл, связанный с объектом, отключая его от потока. Любая ожидающая выходная последовательность записывается в файл. Если поток в настоящее время не связан ни с каким файлом (то есть, файл с ним не был успешно открыт), вызов этой функции завершается с ошибкой.
- ! Обратите внимание, что любой открытый файл автоматически закрывается, когда объект ifstream уничтожается.

Чтобы привязать тот или иной поток к файлу (открыть файл для чтения или для записи) используется метод open, которому необходимо передать параметр – текстовую строку, содержащую имя открываемого файла.

```
in.open("input.txt");
out.open("output.txt");
```

После открытия файлов и привязки их к файловым потокам, работать с файлами можно так же, как со стандартными потоками ввода-вывода cin и cout. Например, чтобы вывести значение переменной n в поток out используются следующая операция:

```
out<<n;
```

А чтобы считать значение переменной из потока in:

```
in>>n;
```

Для закрытия ранее открытого файла используется метод close() без аргументов:

```
in.close();
out.close();
```

Пример 1: Необходимо создать текстовый файл и записать в него строку “Добрый день, ребята!”

1.Создать объект класса ofstream;
2.Связать объект класса с файлом, в который будет производиться запись;
3.Записать строку в файл;
4.Закрыть файл.

```cpp
#include "stdafx.h"
#include <fstream>
using namespace std;

int main()
{
    ofstream fout;
    fout.open("studio.txt");// создаём объект класса ofstream для записи и связываем его с файлом cppstudio.txt
    fout << "Добрый день, ребята!"; // запись строки в файл
    fout.close(); // закрываем файл
    return 0;
}
```

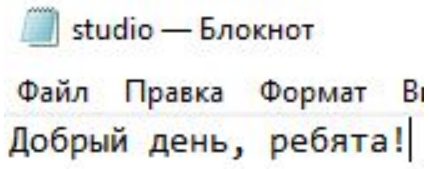
```cpp
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    char name_of_file[50];
    cout << "Название файла: ";
    cin >> name_of_file;
    ofstream fout;
    fout.open(name_of_file);
    fout << "Добрый день, ребята!";
    fout.close();
    return 0;
}
```

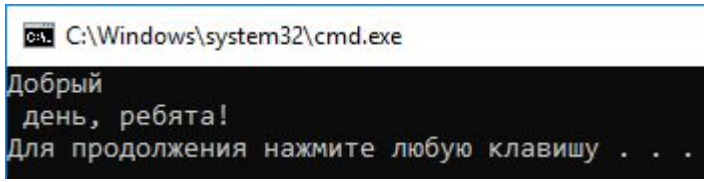
Пример 2: Прочитать тот же файл.

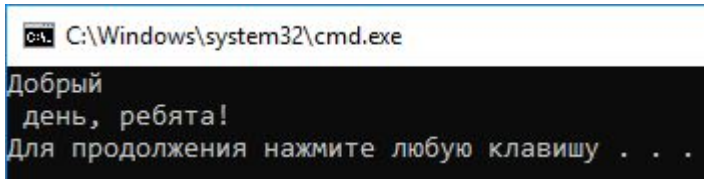
- 1. Создать объект класса ifstream и связать его с файлом, из которого будет производиться считывание;
- 2. Прочитать файл;
- 3. Закрыть файл.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "rus"); // корректное отображение Кириллицы
    char s[50];
    ifstream fin("studio.txt"); // открыли файл для чтения

    fin >> s; // считали первое слово из файла
    cout << s << endl; // напечатали это слово

    fin.getline(s, 50); // считали строку из файла
    fin.close(); // закрываем файл
    cout << s << endl; // напечатали эту строку

    system("pause");
    return 0;
}
```

Выводы


studio — Блокнот
Файл Правка Формат Вы
Добрый день, ребята!


C:\Windows\system32\cmd.exe
Добрый
день, ребята!
Для продолжения нажмите любую клавишу . . .

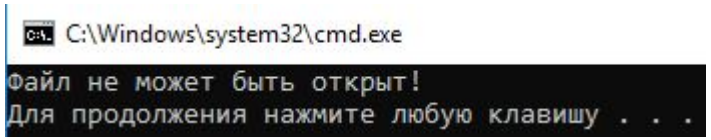


- В примере 2 были показаны два способа чтения из файла, первый – используя операцию передачи в поток, второй – используя функцию getline(). В первом случае считывается только первое слово, как и при обычном выводе cin, а во втором целая строка, но так как в строке осталось меньше 50 символов, то считаются до последнего включительно. Считывание строки начинается со второго слова, так как первое уже было записано через операцию передачи в поток.

is_open()

- Если файл не находится из-за ошибки ввода названия файла или передачи несуществующего файла, то можно использовать функцию is_open(), где 1 – файл открыт, а 0 – файл не был открыт. Это нужно для того, чтобы пользователю было понятно, почему не выводит содержимое файла.

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "rus");
    char s[50];
    ifstream fin("studio.doc"); // (ВВЕЛИ НЕ КОРРЕКТНОЕ ИМЯ ФАЙЛА)

    if (!fin.is_open()) // если файл не открыт
        cout << "Файл не может быть открыт!"<<endl;
    else
    {
        fin >> s;
        cout << s << endl;

        fin.getline(s, 50);
        fin.close();
        cout << s << endl;
    }
    return 0;
}
```

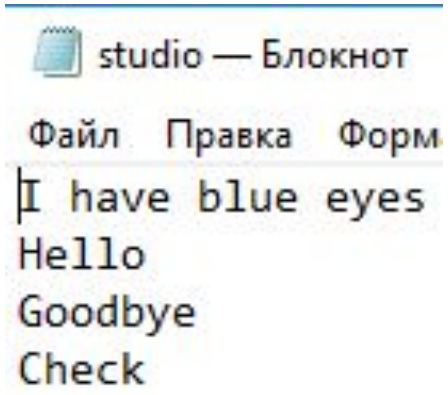


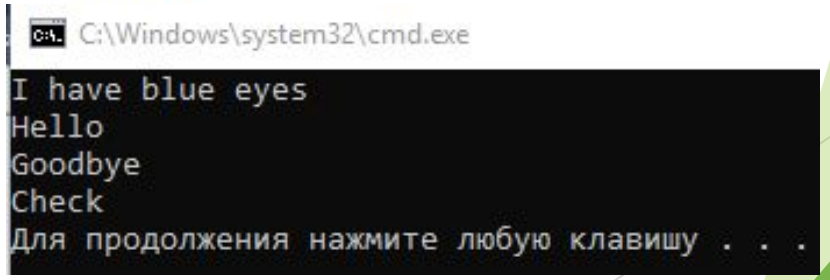
eof(end of file)

Пример 3: Вывести построчно содержимое из файла, пока не будет достигнут его конец

- Функция eof() показывает, достигнут ли конец файла. 1 – достигнут конец, 0 – в противном случае.

```
#include "stdafx.h"
#include <iostream>
#include <string> // подключаем строки
#include <fstream> // подключаем файлы

using namespace std;

int main() {
    string s; // сюда кладём считанные строки
    ifstream file("studio.txt");

    while (!file.eof()) // пока не достигнут конец файла
    {
        getline(file, s); //класть очередную строку в переменную(s)
        cout << s << endl;
    }

    file.close();

    return 0;
}
```





Режимы открытия файлов

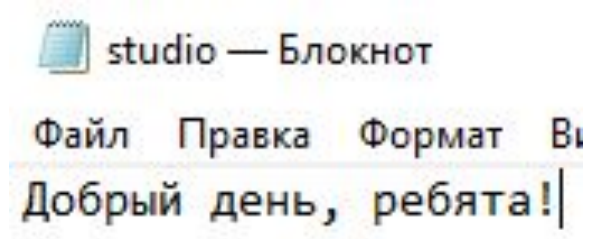
- Режимы открытия файлов устанавливают характер использования файлов. Для установки режима в классе ios_base предусмотрены константы, которые определяют режим открытия файлов.

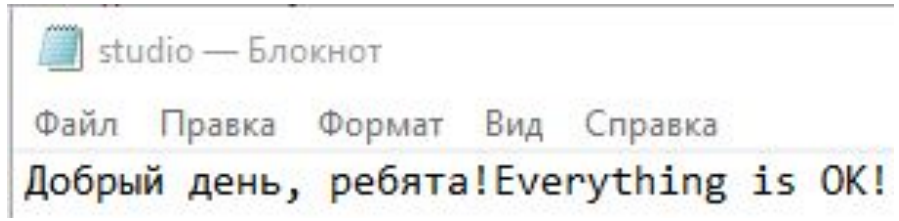
Таблица 1 — режимы открытия файлов

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

Пример 4: Добавить в конец файла строку

```cpp
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    char s[50];
    ofstream fin("studio.txt", ios_base::app);
    if (fin.is_open())
    {
        fin << "Everything is OK!";
        fin.close();
    }

    else
    {
        cout << "Error! File is not found";
    }
    return 0;
}
```

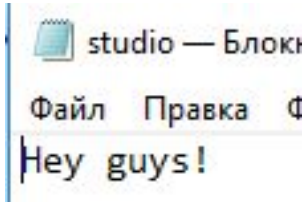


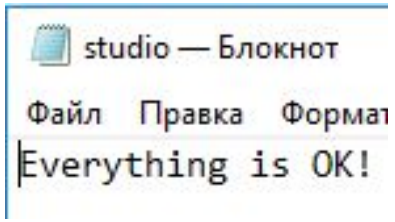


Пример 5: Скомбинировать удаление текста, если он есть, и запись нового текста

- Также можно комбинировать с помощью логической операции ИЛИ (|)

```
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;

int main()
{
    char s[50];
    fstream fin;
    fin.open("studio.txt", ios_base::out | ios_base::trunc);
    if (fin.is_open())
    {
        fin << "Everything is OK!";
        fin.close();
    }

    else
    {
        cout << "Error! File is not found";
    }
    return 0;
}
```





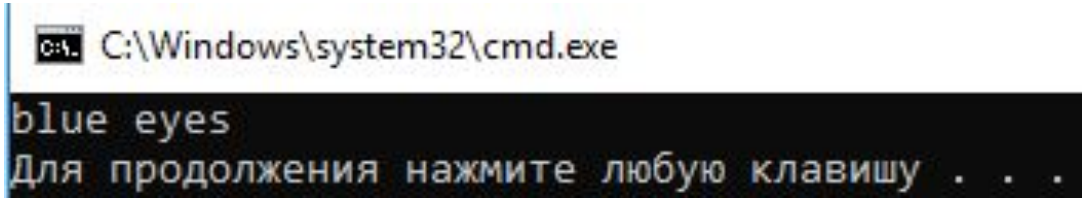
Произвольный доступ к файлу

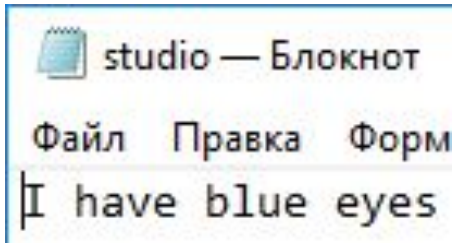
Позиция	Значение
ios::beg	Начало файла
ios::cur	Текущее положение
ios::end	Конец файла

- C++ позволяет осуществлять произвольный доступ к файлу:
- **ifstream &seekg(Смещение, Позиция);** - определяет, где будет произвводиться следующая операция ввода (выведение результата в консоль)
- **ofstream &seekp(Смещение, Позиция);** - определяет, где будет производиться следующая операция вывода (выведение результата в файл)
- ***Смещение*** определяет область значений в пределах файла (long int).

Пример 6: Написать строку в файл, заменить с определённой позиции слова, отобразить полученную строку в файле, закрыть файл. Открыть тот же файл, с определённой позиции, вывести в консоль получившуюся строку

```cpp
#include "stdafx.h"
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    char s[50];
    fstream fin;
    fin.open("studio.txt", ios::out);
    fin << "I have 3 dogs!" << endl;
    fin.seekp(7, ios::beg);
    fin << "blue eyes";
    fin.close();
    fin.open("studio.txt", ios::in);
    fin.seekg(-9, ios::end);
    fin.getline(s, 50);
    fin.close();
    cout << s << endl;
    return 0;
}
```





Краткие итоги:

- Файловые потоки нужны для обмена информацией с файлами на внешних носителях данных (например, на магнитном диске).
- Работу с объемными данными и организацию долговременного хранения данных в языках программирования осуществляют с помощью файлов, расположенных на внешних носителях.
- *Обмен данными* с *внешними устройствами* осуществляется с помощью организации потоков – общего понятия, которое достаточно наглядно демонстрирует направленную передачу данных *по* специально организованным каналам. При этом под *внешними устройствами* следует понимать *устройства ввода-вывода* данных, к которым также можно отнести и файлы.
- Поток – это абстрактное понятие, относящееся к любому переносу данных от источника к приемнику.
- Чтение данных из потока называется извлечением, вывод в поток – помещением, или включением.
- Поток определяется как последовательность байтов и не зависит от конкретного устройства, с которым производится обмен (оперативная память, файл на диске, клавиатура или принтер).

Раздел 2. *Функции для построчного ввода-вывода*

- Перед тем как выполнять эти операции, надо открыть файл и получить доступ к нему.

- В языке программирования С указатель на файл имеет тип FILE и его объявление выглядит так:

  FILE *myfile;

fopen()

- Функция **fopen()** открывает файл по указанному в качестве первого аргумента адресу в режиме чтения ("r"), записи ("w") или добавления ("a") и возвращает в программу указатель на него. Поэтому процесс открытия файла и подключения его к программе выглядит примерно так:
- **myfile = fopen ("hello.txt", "r");**
- При чтении или записи данных в файл обращение к нему осуществляется посредством файлового указателя (в данном случае, myfile).

fclose()

- После того, как работа с файлом закончена, принято его закрывать, чтобы освободить буфер от данных и по другим причинам. Это особенно важно, если после работы с файлом программа продолжает выполняться. Разрыв связи между внешним файлом и указателем на него из программы выполняется с помощью функции **fclose()**. В качестве параметра ей передается указатель на файл:
- **fclose(myfile);**
- В программе может быть открыт не один файл. В таком случае каждый файл должен быть связан со своим файловым указателем. Однако если программа сначала работает с одним файлом, потом закрывает его, то указатель можно использовать для открытия второго файла.

Режимы открытия для fopen()

Символ	Значение
r	Открыть для чтения
w	Удалить содержимое и открыть для записи
a	Данные будут добавлены
r+	Помимо чтения разрешить также запись
w+	Удалить содержимое и открыть для записи и чтения
b	Двоичный файл (преобразование символов окончания строк отсутствует)

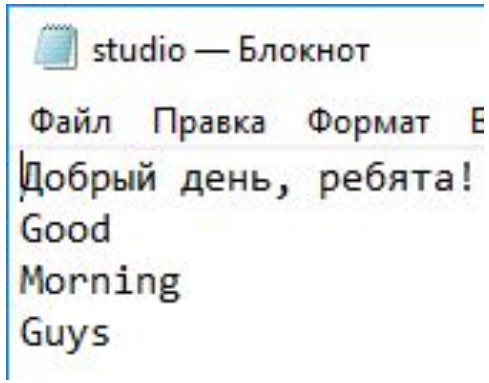
fgets()

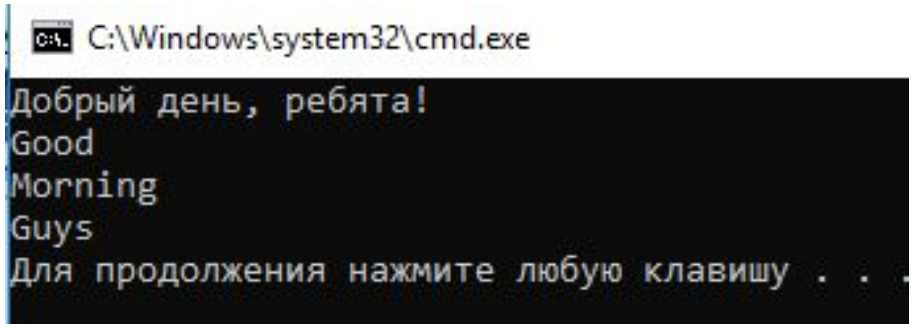
- Функция fgets() аналогична функции gets() и осуществляет построчный ввод из файла. Один вызов fgets() позволят прочитать одну строку.
- **fgets (массив_символов, количество_считываемых_символов, указатель_на_файл);**
- **fgets (s, 50, myfile);**
- Данный вызов прочитает из файла, связанного с указателем myfile, одну строку текста полностью, если ее длина меньше 50 символов вместе с ‘\n’, что также хранится в массиве. Последним будет символ ‘\0’(нудь-терминатор). Если строка будет длиннее, то функция будет читать 49 символов и в конце будет нуль-терминатор.

Пример 7: Считать данные строка за строкой в массив s[50]

```
#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");
    FILE *myfile;
    char s[50];
    myfile = fopen("studio.txt", "r");

    while (fgets(s, 50, myfile) != NULL)//возвращает NULL, если не может прочитать строку
    {
        cout << s;
    }
    cout << endl;
    fclose(myfile);

    return 0;
}
```

Выводы


studio — Блокнот
Файл Правка Формат
Добрый день, ребята!
Good
Morning
Guys

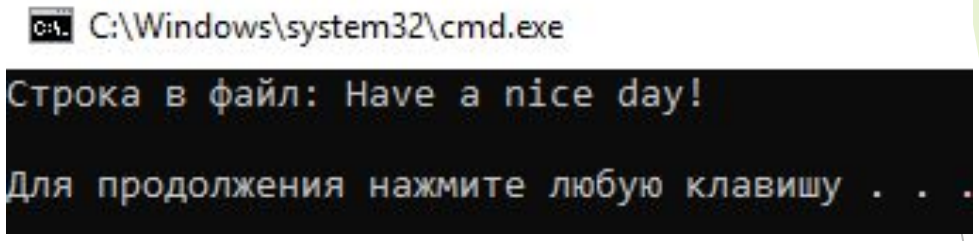

C:\Windows\system32\cmd.exe
Добрый день, ребята!
Good
Morning
Guys
Для продолжения нажмите любую клавишу . . .

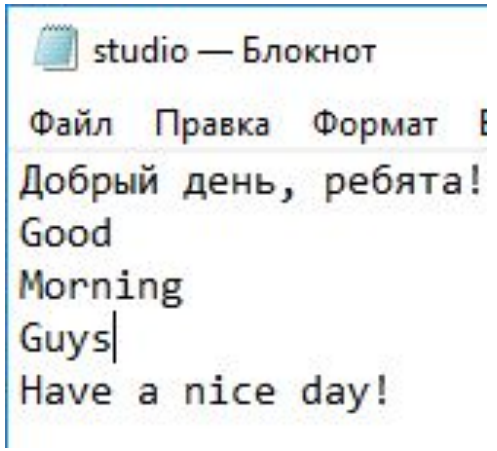
fputs()

- Функция построчного вывода.
- **fputs (строка, файловый_указатель).**
- Функция начинает копирование с адреса, указанного в строке, пока не будет достигнут нулевой символ . Этот нулевой символ не копируется в поток.

Пример 8: Ввести строку для добавления в файл и вывести эту строку в файл

```
#include "stdafx.h"
#include <iostream>
#include <cstdio>
using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");
    FILE *myfile;
    char s[256];
    myfile = fopen("studio.txt", "a");

    // записать строку из стандартного потока ввода в символьный массив
    cout << "Строка в файл: ";
    fgets(s, 255, stdin); // stdin - стандартный поток для ввода

    fputs(s, myfile); // добавить строку в файл
    fclose(myfile);
    cout << endl;
    return 0;
}
```





Пример 9: Копирование файла in в файл out

- Алгоритм:

1. Делаем указатель на исходный и принимающий файл
2. Создаем строку
3. Открываем исходный файл для чтения и принимающий для записи
4. Читаем символы из файла в строку
5. Записываем символы из строки в принимающий файл
6. Закрываем оба файла

```cpp
#include "stdafx.h"
#include <iostream>
using namespace std;

int main()
{
    setlocale(LC_ALL, "Russian");

    FILE *in, *out;

    char buf[255];
    in = fopen("studio.txt", "r");
    out = fopen("studio1.txt", "w");

    while (fgets(buf, 255, in) != 0)
    fputs(buf, out);

    fclose(in);
    fclose(out);
    remove("studio.txt");
    return 0;
}
```