Работа с файлами в С++

Большинство компьютерных программ работают с файлами, и поэтому возникает необходимость создавать, удалять, записывать читать, открывать файлы.

Файл – именованный набор байтов, который может быть сохранен на некотором накопителе. Ну, теперь ясно, что под файлом понимается некоторая последовательность байтов, которая имеет своё, уникальное имя, например файл.txt. В одной директории 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 адрес к директории файла с указанием имени файла, например: D:\docs\file.txt. Важно понимать эти базовые понятия, иначе сложно будет работать с файлами.

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.

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 объектам cin иcout.

Например, необходимо создать текстовый файл и записать в него строку «Работа с файлами в С++». 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.

Почему необходимо создавать объект класса ofstream, а не класса ifstream? Потому, что нужно сделать запись в файл, а если бы нужно было считать данные из файла, то создавался бы объект класса ifstream.

// создаём объект для записи в файл

ofstream /\*имя объекта\*/; // объект класса ofstream

Назовём объект – fout, Вот что получится:

ofstream fout;

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

out.open("cppstudio.txt"); // связываем объект с файлом

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. Итак, файл открыт, осталось записать в него нужную строку. Делается это так:

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

Используя операцию передачи в поток совместно с объектом fout строка Работа с файлами в С++ записывается в файл. 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 С++.

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

ofstream fout("cppstudio.txt"); // создаём объект класса ofstream и связываем его с файлом cppstudio.txt

Объединим весь код и получим следующую программу.

// file.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"

#include <fstream>

using namespace std;

int main(int argc, char\* argv[]) {

    ofstream fout("cppstudio.txt"); // создаём объект класса ofstream для записи и связываем его с файлом cppstudio.txt

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

    fout.close(); // закрываем файл

    system("pause");

    return 0;

}

Осталось проверить правильность работы программы, а для этого открываем файл cppstudio.txt и смотрим его содержимое, должно быть — Работа с файлами в С++.

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

1. создать объект класса ifstream и связать его с файлом, из которого будет производиться считывание;
2. прочитать файл;
3. закрыть файл.

// file\_read.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"

#include <fstream>

#include <iostream>

using namespace std;

int main(int argc, char\* argv[]) {

    setlocale(LC\_ALL, "rus"); // корректное отображение Кириллицы

    char buff[50]; // буфер промежуточного хранения считываемого из файла текста

    ifstream fin("cppstudio.txt"); // открыли файл для чтения

    fin >> buff; // считали первое слово из файла

    cout << buff << endl; // напечатали это слово

    fin.getline(buff, 50); // считали строку из файла

    fin.close(); // закрываем файл

    cout << buff << endl; // напечатали эту строку

    system("pause");

    return 0;

}

В программе показаны два способа чтения из файла, первый – используя операцию передачи в поток, второй – используя функцию getline(). В первом случае считывается только первое слово, а во втором случае считывается строка, длинной 50 символов. Но так как в файле осталось меньше 50 символов, то считываются символы включительно до последнего. Обратите внимание на то, что считывание во второй раз (**строка 17**) продолжилось, после первого слова, а не с начала, так как первое слово было прочитано в **строке 14**. Результат работы программы показан на рисунке 1.

CppStudio.com

Работа   
с файлами в С++

Для продолжения нажмите любую клавишу ...

Рисунок 1 — Работа с файлами в С++

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. Что тогда? В этом случае ничего не произойдёт вообще. Файл не будет найден, а значит и прочитать его невозможно. Поэтому компилятор проигнорирует строки, где выполняется работа с файлом. В результате корректно завершится работа программы, но ничего, на экране показано не будет. Казалось бы это вполне нормальная реакции на такую ситуацию. Но простому пользователю не будет понятно, в чём дело и почему на экране не появилась строка из файла. Так вот, чтобы всё было предельно понятно в С++ предусмотрена такая 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.

// file\_read.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"

#include <fstream>

#include <iostream>

using namespace std;

int main(int argc, char\* argv[])

{

    setlocale(LC\_ALL, "rus"); // корректное отображение Кириллицы

    char buff[50]; // буфер промежуточного хранения считываемого из файла текста

    ifstream fin("cppstudio.doc"); // (ВВЕЛИ НЕ КОРРЕКТНОЕ ИМЯ ФАЙЛА)

    if (!fin.is\_open()) // если файл не открыт

        cout << "Файл не может быть открыт!\n"; // сообщить об этом

    else

    {

    fin >> buff; // считали первое слово из файла

    cout << buff << endl; // напечатали это слово

    fin.getline(buff, 50); // считали строку из файла

    fin.close(); // закрываем файл

    cout << buff << endl; // напечатали эту строку

    }

    system("pause");

    return 0;

}

Результат работы программы показан на рисунке 2.

CppStudio.com

Файл не может быть открыт!

Для продолжения нажмите любую клавишу ...

Рисунок 2 — Работа с файлами в С++

Как видно из рисунка 2, программа сообщила о невозможности открыть файл. Поэтому, если программа работает с файлами, рекомендуется использовать эту функцию, is\_open(), даже, если уверены, что файл существует.

Режимы открытия файлов

Режимы открытия файлов устанавливают характер использования файлов. Для установки режима в классе ios\_base предусмотрены константы, которые определяют режим открытия файлов (см. Таблица 1).

| 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 |

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

ofstream fout("cppstudio.txt", ios\_base::app); // открываем файл для добавления информации к концу файла

fout.open("cppstudio.txt", ios\_base::app); // открываем файл для добавления информации к концу файла

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.

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 ifstreamсвязываясь с файлом, имеют по умолчанию режим открытия файла ios\_base::in— файл открыт только для чтения. Режим открытия файла ещё называют — флаг, для удобочитаемости в дальнейшем будем использовать именно этот термин. В таблице 1 перечислены далеко не все флаги, но для начала этих должно хватить.

Обратите внимание на то, что флаги ateиappпо описанию очень похожи, они оба перемещают указатель в конец файла, но флаг app позволяет производить запись, только в конец файла, а флаг ate просто переставляет флаг в конец файла и не ограничивает места записи.

Разработаем программу, которая, используя операцию sizeof(), будет вычислять характеристики основных типов данных в С++ и записывать их в файл. Характеристики:

1. число байт, отводимое под тип данных
2. максимальное значение, которое может хранить определённый тип данных.

Запись в файл должна выполняться в таком формате:

/\*  data type      byte          max value

bool               =  1         255.00

char               =  1         255.00

short int          =  2         32767.00

unsigned short int =  2         65535.00

int                =  4         2147483647.00

unsigned int       =  4         4294967295.00

long int           =  4         2147483647.00

unsigned long int  =  4         4294967295.00

float              =  4         2147483647.00

long float         =  8         9223372036854775800.00

double             =  8         9223372036854775800.00  \*/

Такая программа уже разрабатывалась ранее в разделе [Типы данных С++](http://cppstudio.com/obuchenie_cpp/tipy_dannyh), но там вся информация о типах данных выводилась на стандартное устройство вывода, а нам необходимо программу переделать так, чтобы информация записывалась в файл. Для этого необходимо открыть файл в режиме записи, с предварительным усечением текущей информации файла (**строка 14**). Как только файл создан и успешно открыт (строки 16 — 20), вместо оператора cout, в **строке 22** используем объект fout. таким образом, вместо экрана информация о типах данных запишется в файл.

// write\_file.cpp: определяет точку входа для консольного приложения.

#include "stdafx.h"

#include <iostream>

#include <fstream> // работа с файлами

#include <iomanip> // манипуляторы ввода/вывода

using namespace std;

int main(int argc, char\* argv[]) {

    setlocale(LC\_ALL, "rus");

    // связываем объект с файлом, при этом файл открываем в режиме записи, предварительно удаляя все данные из него

    ofstream fout("data\_types.txt", ios\_base::out | ios\_base::trunc);

    if (!fout.is\_open()) // если файл небыл открыт

    {

     cout << "Файл не может быть открыт или создан\n"; // напечатать соответствующее сообщение

     return 1; // выполнить выход из программы

    }

        fout << "     data type      "   << "byte"                      << "      "    << "    max value  " << endl // заголовки столбцов

             << "bool               =  " << sizeof(bool)                << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных bool\*/                           << (pow(2,sizeof(bool) \* 8.0) - 1)               << endl

         << "char               =  " << sizeof(char)                << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных char\*/                           << (pow(2,sizeof(char) \* 8.0) - 1)               << endl

         << "short int          =  " << sizeof(short int)           << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных short int\*/                      << (pow(2,sizeof(short int) \* 8.0 - 1) - 1)      << endl

             << "unsigned short int =  " << sizeof(unsigned short int)  << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных unsigned short int\*/             << (pow(2,sizeof(unsigned short int) \* 8.0) - 1) << endl

             << "int                =  " << sizeof(int)                 << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных int\*/                            << (pow(2,sizeof(int) \* 8.0 - 1) - 1)            << endl

             << "unsigned int       =  " << sizeof(unsigned int)        << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных unsigned int\*/                   << (pow(2,sizeof(unsigned int) \* 8.0) - 1)       << endl

             << "long int           =  " << sizeof(long int)            << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных long int\*/                       << (pow(2,sizeof(long int) \* 8.0 - 1) - 1)       << endl

             << "unsigned long int  =  " << sizeof(unsigned long int)   << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных undigned long int\*/              << (pow(2,sizeof(unsigned long int) \* 8.0) - 1)  << endl

             << "float              =  " << sizeof(float)               << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных float\*/                          << (pow(2,sizeof(float) \* 8.0 - 1) - 1)          << endl

             << "long float         =  " << sizeof(long float)          << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных long float\*/                     << (pow(2,sizeof(long float) \* 8.0 - 1) - 1)     << endl

             << "double             =  " << sizeof(double)              << "         " << fixed << setprecision(2)

/\*вычисляем максимальное значение для типа данных double\*/                         << (pow(2,sizeof(double) \* 8.0 - 1) - 1)         << endl;

        fout.close(); // программа больше не использует файл, поэтому его нужно закрыть

    cout << "Данные успешно записаны в файл data\_types.txt\n";

    system("pause");

    return 0;

}

Нельзя не заметить, что изменения в программе минимальны, а всё благодаря тому, что стандартный ввод/вывод и файловый ввод/вывод используются абсолютно аналогично. В конце программы, в **строке 45** мы явно закрыли файл, хотя это и не обязательно, но считается хорошим тоном программирования. Стоит отметить, что все функции и манипуляторы, используемые для форматирования стандартного ввода/вывода, актуальны и для файлового ввода/вывода. Поэтому не возникло никаких ошибок, когда оператор coutбыл заменён объектом fout.