**Лабораторная работа №14.**

**Потоки ввода-вывода. Файловые операции**

## 1 Цель и порядок работы

Цель работы – ознакомиться с возможностями ввода-вывода языка C++, освоить основные операции работы с файлами.

Порядок выполнения работы:

* ознакомиться с описанием лабораторной работы;
* получить задание у преподавателя, согласно своему варианту;
* написать программу и отладить ее на ЭВМ;
* оформить отчет.

## 2 Краткая теория

Текстовыми называют файлы, состоящие из любых символов. Они организуются  *по* строкам, каждая из которых заканчивается символом "конец строки". Конец самого файла обозначается символом "*конец 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.

1. Для работы с файлами используются специальные типы данных, называемые потоками. *Поток* ifstream служит для работы с файлами в режиме чтения.
2. *Поток* ofstream служит для работы с файлами в режиме записи.
3. Для работы с файлами в режиме как чтения, так и записи служит *поток* iofstream.

В программах на C++ при работе с текстовыми файлами необходимо подключать библиотеки **iostream** и **fstream**.

Для того, чтобы записать данные в текстовый *файл*, необходимо:

1. Описать переменную типа **ofstream**.
2. Открыть файл с помощью функции **open**.
3. Вывести информацию в файл.
4. Закрыть файл.

Для того, чтобы считать данные из текстового файла, необходимо:

1. Описать переменную типа **ifstream**.
2. Открыть файл с помощью функции **open**.
3. Считать информацию из файла, при считывании каждой порции данных необходимо проверять достигнут ли конец файла.
4. Закрыть файл.

**Запись информации в текстовый файл**

Для того, чтобы начать работать с текстовым файлом, необходимо описать переменную типа **ofstream**. Например, с помощью оператора

**ofstream F;**

будет создана переменная **F** для записи информации в файл. На следующем этапе файл необходимо открыть для записи. В общем случае оператор открытия потока будет иметь вид:

**F.open("file", mode);**

Здесь F — переменная, описанная как **ofstream**, **file** — имя файла на диске, **mode** — режим работы с открываемым файлом.

*Файл может быть открыт в одном из следующих режимов:*

* **ios::in** — открыть файл в режиме чтения данных, *этот режим является режимом по умолчанию для потоков* **ifstream**;
* **ios::out** — открыть файл в режиме записи данных (при этом информация в существующем файле уничтожается*), этот режим является режимом по умолчанию для потоков* **ofstream**;
* **ios::app** — открыть файл в режиме записи данных в конец файла;
* **ios::ate** — передвинуться в конец уже открытого файла;
* **ios::trunc** — очистить файл, это же происходит в режиме **ios::out**;
* **ios::nocreate** — не выполнять операцию открытия файла, если он не существует6;
* **ios::noreplace** — не открывать существующий файл.

Параметр mode может отсутствовать, в этом случае файл открывается в режиме по умолчанию для данного потока.

После удачного открытия файла (в любом режиме) в переменной **F** будет храниться **true**, в противном случае **false**. Это позволит проверять корректность операции открытия файла.

Открыть файл (в качестве примера возьмём файл **abc.txt**) в режиме записи можно одним из следующих способов:

//Первый способ.

**ofstream f;**

**f.open ( " abc.txt ", ios::out );**

//Второй способ,

//режим **ios::out** является режимом по умолчанию для потока **ofstream**

**ofstream f;**

**f.open ( " abc.txt " )**

//Третий способ объединяет описание переменной типа поток

//и открытие файла в одном операторе

**ofstream f( " abc.txt ", ios::out );**

После открытия файла в режиме записи будет создан пустой файл, в который можно будет записывать информацию. Если нужно открыть файл, чтобы в него что-либо дописать, то в качестве режима следует использовать значение **ios::app**.

### Закрытие файла

В классах файловых потоков имеется функция **close()**, которая сбрасывает содержимое потока и закрывает ассоциированный с ним файл.

Кроме того, деструктор потока автоматически закрывает файл при уничтожении объекта потока.

При ошибке закрытия файла устанавливается флаг **failbit**.

### Чтение символов и строк

Для чтения одиночных символов, а также строк применяется функция **get** класса **istream**. Эта функция перегружена следующим образом:

int get();

istream &get(char &c) ;

istream &get(char \*buf, long len, char t = '\n');

Две первые формы функции предназначены для извлечения из потока одиночного символа. Функция **int get()** возвращает символ в качестве своего значения. Функция **get (char &c)** передает символ в параметре и возвращает ссылку на свой поток.

**Пример 1.** Блок посимвольного копирования файлов

ifstream ifs("infile.dat");

ofstream ofs("outfile.dat");

// put (char) передает в поток

// одиночный символ.

while (ifs & ofs)

ofs.put(ifs.get());

Последняя форма функции **get()** извлекает из потока последовательность символов. Символы читаются в буфер **buf**, пока не произойдет одно из следующих событий:

* будет встречен ограничивающий символ **t** (по умолчанию **'\n'**);
* будет встречен конец файла;
* в буфер будет записано **len** символов, включая завершающий строку **0**.

Имеется еще функция **getline()**, во всем аналогичная этой форме **get()**:

Функция **getline(is, str, delim)** имеет три параметра:

**is** – Входной поток, из которого извлекается строка.

**str –** количество символов новой строки илиимя строки, в которую считываются символы из входного потока.

**delim ­–** Разделитель строк, по умолчанию **'\n'**.

istream &getline(char \*buf, long len, char t = '\n');

Разница между этими двумя функциями состоит в том, что **getline()** извлекает из потока ограничивающий символ, в то время как **get()** этого не делает. Ни та, ни другая функция не записывает ограничивающий символ в буфер.

Пример 2. Блок чтения строк с помощью **get()**

#include <iostream>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

char name[9], ext[4];

cout << "Введите имя файла: ";

cin.get(name, 9, '.');

cin.ignore (80, '.'); // Удалить все оставшиеся до точки символы

cin.get(ext, 4) ;

cin.ignore(80, '\n'); // Удалить все, что осталось в строке

cout<< "Имя: "<< name << " Расширение: " << ext << endl;

return 0;

}

## Примеры программ

#include <fstream>

#include <iostream>

using namespace std;

int main()

{

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;

}

**Пример 1. Поиск подстроки**

Написать программу, которая определяет, сколько раз встречается в заданном текстовом файле заданная последовательность символов. Длина строки текста не превышает 80 символов, текст не содержит переносов слов, последовательность не содержит пробельных символов.

**Алгоритм решения задачи.**

1. Построчно считывать текст из файла.

2. Для каждой строки проверять, содержится ли в ней заданная последовательность.

3. Если да, напечатать сообщение о наличии заданной последовательности и завершить программу.

4. При нормальном выходе из цикла напечатать сообщение об отсутствии заданной последовательности и завершить программу.

#include <fstream>

#include <iostream>

#include <cstring>

#include <windows.h>

using namespace std;

int main()

{

setlocale(LC\_ALL, "rus");

const int len = 81;

char word1[len], word2[len], line[len];

cout << " Введите слово для поиска:: ";

cin >> word1;

int k = 0;

OemToCharA(word1, word2); // Переводит строку из набора символов, определенного //OEM, в ANSI или строку с широким символом.

ifstream fin("cppstudio.txt");

if (!fin)

{

cout << " Ошибка открытия файла." << endl;

return 1;

}

while (fin.getline(line, len))

{

cout << line << endl;

if (strstr(line, word2)) k++;

}

if (k)

cout << " слово встречается " << k << " раз" << endl;

else cout << "слово отсутствует!" << endl;

fin.close();

system("pause");

return 0;

}

**Пример 2.** Из файла dat.txt считываются элементы одномерного массива не более 50 и выводятся на экран.

Из массива (файл dat.txt) выбираются все отрицательные элементы и записываются в другой файл rez.txt.

#include <fstream>

#include <iostream>

#include <cstring>

#include <windows.h>

using namespace std;

int main() {

setlocale(LC\_ALL, "rus");

float x[50];

int n, i = 0;

ifstream fin("dat.txt"); // 3

if (!fin)

{

cout << "файл открыть невозможно!" << endl;

return 1;

}

ofstream fout("rez.txt");

if (!fout)

{

cout << " файл открыть невозможно!" << endl;

return 1;

}

while (!fin.eof())

{

fin >> x[i]; cout << x[i] << " "; i++;

}

n = i;

for (i = 0; i<n; i++)

if (x[i]<0)

fout << x[i] << " " << endl;

fin.close();

fout.close();

system("pause");

return 0;

}

**Пример 3. Поиск вхождения слова в файле**

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{

setlocale(LC\_ALL, "Russian");

const int len = 81;

char word[len], line[len], end\_word[] = "done";

//пересоздадим файл и откроем для чтения/записи

fstream f("lab11.txt", ios::in | ios::out | ios::trunc);

if (!f)

{

cout << "Ошибка открытия файла" << endl;

return -1;

}

//введем несколько строк для дальнейшего поиска в них

do

{

cin >> line;

//с файловым потоком можно работать как со стандартными cin и cout

f << line << endl;

}

//продолжаем пока не введем done

while (strcmp(line, end\_word));

//сбросим файловый поток на начало

f.seekg(0);

cout << "Содержимое файла: " << endl;

//содержимое файла на экран

//выполняем пока не достигнем конца файла

while (!f.eof())

{

f >> line;

cout << line << endl;

}

cout << "Введите слово для поиска: ";

cin >> word;

//сбрасываем бит-признак конца файла eofbit

f.clear();

//сбросим файловый поток на начало

f.seekg(0);

//определим длину искомого слова

size\_t l\_word = strlen(word);

//счетчик вхождения слова

int wc = 0;

//читаем построчно и ищем слово word в строке

while (f.getline(line, len))

{

//поместим указатель p на начало строки

char \*p = line;

//strstr возвращает указатель

// на элемент из строки p с которого начинается word

while (p = strstr(p, word))

{

//используем указатель не текущую позицию в строке

char \*c = p;

//переместим p на символ сразу за концом слова

p += l\_word;

//проверим стоит ли наше слово отдельно

//или это просто подстрока в другом (большем) слове

//проверим совпадает ли начало слова с началом строки

if (c != line)

//проверим символ перед началом слова

//на принадлежность к разделителям

if ( !ispunct(\*(c-1)) && !isspace(\*(c-1)) )

//начинается не с начала строки и

//не с разделителя => ищем дальше

continue;

//символы перед началом слова подходят

//проверяем символы за окончанием слова

//если это пробелы, символы пунктуации

//или конец строки => увеличиваем счетчик слов

if (ispunct(\*p) || isspace(\*p) || (\*p == '\0'))

{

wc++;

cout << "Слово найдено" <<endl;

}

}

}

cout << "Количество вхождений слова: "<< wc << endl;

return 0;

}

*Замечание.* Когда эта программа открывает уже существующий файл, он усекается до нулевой длины (т.е. все его данные теряются). Если необходимо работать с уже имеющимися в файле данными, нужно убрать бит ios::trunc из режима открытия потока.

**Задания на текстовые файлы**

Все задания в основном направлены на освоение техники работы с файлами. Программы должны обрабатывать файлы, созданные в редакторе «Блокнот» (Windows) или файлы Word, сохраненные как текст DOS.

1. Дано два файла. Получить третий, содержащий поочередно строки исходных двух файлов.
2. Написать программу, которая считывает из текстового файла предложения и записывает в другой файл только те предложений, которые заканчиваются на вопросительный или восклицательный знак.
3. Написать программу, которая считывает текст из файла и записывает в другой файл только предложения, содержащие введенное с клавиатуры слово.
4. Написать программу, которая считывает текст из файла и записывает в другой файл только числа, встречающиеся в тексте.
5. Написать программу, которая считывает английский текст из файла и записывает в другой файл только предложения, начинающиеся с гласных букв.
6. Написать программу, которая считывает текст из файла и записывает в другой файл только предложения, не содержащие запятых.
7. Написать программу, которая считывает текст из файла и записывает в другой файл только цитаты, то есть предложения, заключенные в кавычки.
8. Дан файл, содержащий текст. Сформировать другой файл, содержащий нечетные строки первого файла.
9. Дан файл. Записать в другой файл только те строки, которые начинаются с заданной буквы.
10. Написать программу, которая считывает текст из файла и записывает в другой файл только предложения, состоящие из заданного количества слов.
11. Написать программу, которая считывает английский текст из файла и записывает в другой файл только слова, начинающиеся и оканчивающиеся на гласные буквы.
12. Написать программу, которая считывает английский текст из файла и записывает его в другой файл, заменив каждую первую букву слов, начинающихся с гласной буквы, на прописную.
13. Написать программу, которая считывает текст из файла и записывает его в другой файл, заменив цифры от 0 до 9 на слова «ноль», «один», ..., «девять», начиная каждое предложение с новой строки.
14. Написать программу, которая считывает текст из файла и записывает в другой файл сначала вопросительные, а затем восклицательные предложения.
15. Дан файл, содержащий случайные целые числа. Записать из него в другой файл числа, кратные трем.
16. Дан файл, состоящий из строк, которые могут начинаться либо с русских, либо с английских слов. Записать в другой файл, только те строки, которые начинаются с русских слов.
17. Дан файл, строки которого могут содержать и числа, и слова. Сформировать из него другой файл, содержащий только числа.
18. Дан файл, содержащий целые числа. Записать их в другие два файла, при этом в первый записать четные, а во второй – нечетные числа.
19. Даны два файла, содержащих одинаковое количество целых чисел. Перезаписать первый файл следующим образом: если число из второго файла больше соответствующего числа в первом, то число из второго файла записывается в первый файл на место меньшего.
20. Дан файл, строки которого содержат по четыре числа. Эти числа представляют собой коэффициенты уравнений двух прямых. Перезаписать в другой файл только те строки, в которых данные соответствуют параллельным прямым.
21. Дан файл. Записать из него в другой файл только те строки, которые начинаются с заглавной буквы.
22. Дан файл. Получить из него другой, содержанием которого являются строки первого файла, записанные в обратном порядке.