**Лабораторная работа №16**

**«Текстовые файлы»**

**Теория**

Файлы позволяют пользователю считывать большие объемы данных непосредственно с диска, не вводя их с клавиатуры. Существуют два основных типа файлов: **текстовые и двоичные**.

*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.

В *двоичных* файлах информация считывается и записывается в виде блоков определенного размера, в которых могут храниться данные любого вида и структуры.

**Работа на языке С:**

Стандартная библиотека <stdio.h> содержит следующие функции:

FILE \*fopen(const char \*path, const char \*mode) - открытие файла,

где path –имя файла, mode:

"r" – открытие текстового файла для чтения;

"w" – открытие (создание) текстового файла для записи;

"a" - открытие текстового файла для добавления в конец файла;

int fclose(FILE \*f) – закрытие файла

int fflush(FILE \*f) - записывает все данные, которые еще оставались в буфере вывода, и не проводит закрытие файла

int feof(FILE\*f) - определяет достижение конце файла можно использовать функцию

int fputc(int ch, FILE \*f) – запись символа в файл, где ch – записываемый символ.

int fgetc(FILE \*f) - чтение символа из файла

int fputs(const char \*st, FILE \*f) - запись строки в файл, где st – записываемая строка.

char \*fgets(const char \*st, int len, FILE \*f) - чтение строки файла выполняет функци

где st – указатель на символьный массив, в который помещается вводимая строка, len - число, ограничивающее количество считываемых символов.

void rewind(FILE \*f) - установка курсора файла на его начало

выполняется функцией

int remove(const char \*st) - удаление файла, где st - имя файла.

int rename(const char \*oldfname, const char \*newfname) - переименование файла,

где oldfname - старое имя файла, newfname - новое имя файла.

int fprintf(FILE \*f, const char \*format,... ) - выводит в файл значения аргументов из списка, обозначенного здесь как многоточие, в соответствии со строкой формата format.

int fscanf (FILE \*f,const char \*format,... ) - для чтения из файла

**Работа с текстовыми файлами в С++**

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

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

Для того чтобы записывать данные в текстовый файл, необходимо:

1. описать переменную типа **ofstream**.
2. открыть файл с помощью функции **open**.
3. вывести информацию в файл.
4. обязательно закрыть файл.

Для считывания данных 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

Как было сказано ранее, для того чтобы начать работать с текстовым файлом, необходимо описать переменную типа **ofstream**. Например, так:

**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** — полное имя файла на диске, **mode** — режим работы с открываемым файлом. Обратите внимание на то, что при указании полного имени файла нужно ставить двойной слеш.

Файл может быть открыт в одном из следующих режимов:

* *ios::in* — открыть файл в режиме чтения данных; режим является режимом по умолчанию для потоков **ifstream**;
* *ios::out* — открыть файл в режиме записи данных (при этом информация о существующем файле уничтожается); 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 существует;
* *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 проверить корректность операции открытия файла.

Открыть файл (в качестве примера возьмем файл **D:\\sites\\accounts.txt**) в режиме записи можно одним из следующих способов:

*//первый способ*

ofstream F;

F.open("D:\\sites\<strong>\</strong>accounts<strong>.</strong>txt", ios::out);  
*//второй способ, режим ios::out является режимом по умолчанию*

*//для потока ofstream*

ofstream F;

F.open("D:\\game\\noobs.txt");  
*//третий способ объединяет описание переменной и типа поток*

*//и открытие файла в одном операторе*

ofstream F ("D:\\game\\noobs.txt", ios::out);

После открытия файла в режиме записи будет создан пустой файл, в который можно будет записывать информацию.

Если вы хотите открыть существующий файл в режиме дозаписи, то в качестве режима следует использовать значение **ios::app**.

После открытия файла в режиме записи, в него можно писать точно так же, как и на экран, только вместо стандартного устройства вывода **cout** необходимо указать имя открытого файла.

Например, для записи в поток **F**переменной **a**, оператор вывода будет иметь вид:

**F<<a;**

Для последовательного вывода в поток **G** переменных **b**, **c**, **d** оператор вывода станет таким:

**G<<b<<c<<d;**

Закрытие потока осуществляется с помощью оператора:

**F.close();**

Чтение информации из текстового файла

Для того чтобы прочитать информацию из текстового файла, необходимо описать переменную типа **ifstream**. После этого нужно открыть файл для чтения с помощью оператора **open**. Если переменную назвать **F**, то первые два оператора будут такими:

ifstream F;

F.open("D:\\sites\\accounts.txt", ios::in);

После открытия файла в режиме чтения из него можно считывать информацию точно так же, как и с клавиатуры, только вместо **cin** нужно указать имя потока, из которого будет происходить чтение данных.

Например, для чтения данных из потока **F** в переменную**a**, оператор ввода будет выглядеть так:

**F>>a;**

Два числа в текстовом редакторе считаются разделенными, если между ними есть хотя бы один из символов: пробел, табуляция, символ конца строки. Хорошо, когда программисту заранее известно, сколько и какие значения хранятся в текстовом файле. Однако часто известен лишь тип значений, хранящихся в файле, при этом их количество может быть различным. Для решения данной проблемы необходимо считывать значения из файла поочередно, а перед каждым считыванием проверять, достигнут ли конец файла. А поможет сделать это функция **F.eof()**. Здесь **F** - имя потока функция возвращает логическое значение: **true** или **false**, в зависимости от того достигнут ли конец файла.

Следовательно, цикл для чтения содержимого всего файла можно записать так:

//организуем для чтения значений из файла, выполнение

//цикла прервется, когда достигнем конец файла,

//в этом случае F.eof() вернет истину

while (!F.eof())

{

//чтение очередного значения из потока F в переменную a

F>>a;

//далее идет обработка значения переменной a

}}

**Практика**

1. Создать текстовый файл **D:\\accounts.txt** и записать в него **n** вещественных чисел **(lab16\_01.cpp).**
2. В текстовом файле **D:\\accounts.txt** хранятся вещественные числа, вывести их на экран и вычислить их количество **(lab16\_02.cpp).**
3. Пример демонстрирует создание текстового файла. Введём несколько слов, а потом рассмотрим результат. В созданном файле введённые слова и разделители **(lab16\_1.cpp).**
4. Пример производит чтение из существующего текстового файла. Перепишем файл, созданный в первом примере для чтения во втором. Для нормальной интерпретации введённых текстовых строк, при вводе нужно заменять разделитель с кодом 10 на 0 **(lab16\_2.cpp).**
5. Написать программу, которая создаёт в цикле 100 случайных чисел, переводит их в текстовый вид, используя функцию itoa(n,t,10); где n -сформированное число, t-принимающее текстовое поле, 10- основание системы счисления. И выводит сформированный текст в текстовый файл с именем «text.txt» **(lab16\_3.cpp).**
6. **Задание.** Новая программа генерирует 100 случайных чисел и пишет их в текстовый файл, как в первом примере. После этого закрывает введённый файл, открывает его для чтения, читает записи и переводит их в числовой формат, используя функцию n=atoi(s). Конец файла определяется при помощи функции feof(f)
7. **Пример.** Создать первый текстовый файл из произвольных строк. Переписать строки первого файла во второй текстовый файл так, чтобы в нем не было повторяющихся строк.

Алгоритм создания второго файла состоит в следующем: необходимо считывать очередную строку из первого файла и проверять ее со всеми строками второго файла. Если равная строка найдена во втором файле, то цикл проверки следует закончить и эту строку не записывать во второй файл. Если же строка первого файла не будет обнаружена во втором файле, то ее следует записать во второй файл. **(lab16\_4.cpp).**

1. **Пример.** Создать файл с информацией о студентах. Каждая строка файла должна иметь следующую структуру (в скобках указаны длины полей): номер студента (4), пробел, группа (8), пробел, фамилия (12), оценка 1, оценка 2, оценка 3, средний балл(5). Отсортировать список студентов по фамилиям в порядке, обратном алфавитному (использовать массив), а также найти трех худших студентов по сумме баллов и записать информацию о них во второй файл. Сортировку требуется выполнить при вводе данных из файла, т.е. помещать элемент в массив сразу на требуемое место. При вводе данных вычислить средний балл и занести его сразу в файл.

Для удобства обработки данных представим в программе информацию о каждом студенте в виде структуры. Для записи и чтения данных в файл целесообразно использовать форматный ввод/вывод. Поскольку заранее объем файла не известен, то удобно использовать динамический массив для хранения данных, считанных из файла. При помещении в массив элемента, считанного из файла, поиск места в массиве выполним в цикле, сравнивая считанную фамилию с очередной фамилией, хранящейся в массиве структур. Если считанная фамилия больше текущей, то место вставки найдено: необходимо вставить перед текущим элементом массива. Все элементы массива, начиная с текущего, смещаются на одну позицию в сторону конца массива.

Поиск трех худших студентов можно выполнить, используя тот же алгоритм, что и в предыдущем пункте, но сравнение структур вести по полю средний балл и располагать элементы по возрастанию среднего балла, а не по убыванию **(lab16\_5.cpp).**

1. **Пример.** Отсортировать текстовый файл. Сортировка текстового файла выполняется с использованием дополнительного файла. Сначала можно подсчитать общее количество строк исходного файла, затем в цикле искать минимальную строку исходного файла. Минимальная строка записывается в результирующий файл. Затем содержимое исходного файла, за исключением минимальной строки, переписывается в промежуточный файл. Исходный файл уничтожается. Промежуточный файл переименовывается, при этом он получает имя исходного файла. Данная процедура повторяется в цикле столько раз, каков объем файла. В конце исходный файл уничтожается, а результирующий файл получает имя исходного файла **(lab16\_6.cpp).**
2. **Задание.** Программа принимает на вход файл City.txt, содержащий список центральных городов России. Названия городов читаются из массива, отображаются на монитор, после этого создаётся новый файл, куда выбранные названия пишутся по возрастанию.
3. **Задание.** Написать программу, которая содержит функцию, которая принимает на вход целое трехзначное число, а на выход выдаёт текстовое выражение этого числа. (121=сто двадцать один).
4. **Задание.** Сформировать текстовый файл, содержащий информацию о марке компьютера, его тактовой частоте, объеме оперативной и внешней памяти. Создать второй файл, записав в него сведения о компьютерах, удовлетворяющих заданным требованиям (вводятся с клавиатуры).