ПРАКТИЧЕСКАЯ РАБОТА № 8
РАБОТА С ФАЙЛАМИ В ЯЗЫКЕ С++

ЦЕЛЬ ПРАКТИЧЕСКОЙ РАБОТЫ:

Целью данной практической работы является приобретение практических навыков по работе с файлами на языке программирования языке С++.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ:

Большинство компьютерных программ работают с файлами, и поэтому возникает необходимость создавать, удалять, записывать читать, открывать файлы. Что же такое файл? 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 операторам cin и cout.

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

- создать объект класса ofstream;
- связать объект класса с файлом, в который будет производиться запись;
- записать строку в файл;
- 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.
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.

```
fout.open("text.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 C++"; // запись строки в
файл
```

Используя операцию передачи в поток совместно с объектом fout строка Работа с файлами в C++ 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.

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

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

Объединим весь код и получим следующую программу.

```
1) #include "stdafx.h"
2) #include <fstream>
3) using namespace std;
4)
5) int main(int argc, char* argv[])
6) {
7)     ofstream fout("text.txt"); // создаём объект
   класса ofstream для записи и связываем его с файлом
   text.txt
8)     fout << "Работа с файлами в C++"; // запись строки
   в файл
9)     fout.close(); // закрываем файл
10)      system("pause");
11)      return 0;
12)  }
```

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

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, из которого будет производиться считывание;
- прочитать файл;
- закрыть файл.

```
1) #include "stdafx.h"
2) #include <fstream>
3) #include <iostream>
4) using namespace std;
5)
6) int main(int argc, char* argv[])
7) {
8)     setlocale(LC_ALL, "rus"); // корректное
   отображение Кириллицы
9)     char buff[50]; // буфер промежуточного хранения
   считываемого из файла текста
10)     ifstream fin("text.txt"); // открыли файл для
   чтения
11)
12)     fin >> buff; // считали первое слово из файла
13)     cout << buff << endl; // напечатали это слово
14)     fin.getline(buff, 50); // считали строку из
   файла
15)     fin.close(); // закрываем файл
16)     cout << buff << endl; // напечатали эту строку
17)
18)     system("pause");
19)     return 0;
20) }
```

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.

Программа работает правильно, но не всегда так бывает, даже в том случае, если с кодом всё впорядке. 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 не возможно. 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 C++ 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.

```
1) // file_read.cpp: определяет точку входа для
   консольного приложения.
2)
3) #include "stdafx.h"
4) #include <fstream>
5) #include <iostream>
6) using namespace std;
7)
8) int main(int argc, char* argv[])
9) {
10)      setlocale(LC_ALL, "rus"); // корректное
   отображение Кириллицы
11)      char buff[50]; // буфер промежуточного хранения
   считываемого из файла текста
12)      ifstream fin("text.doc"); // (ВВЕЛИ НЕ
   КОРРЕКТНОЕ ИМЯ ФАЙЛА)
13)
14)      if (!fin.is_open()) // если файл не открыт
15)          cout << "Файл не может быть открыт!\n"; //
   сообщить об этом
16)      else
17)      {
18)      fin >> buff; // считали первое слово из файла
19)      cout << buff << endl; // напечатали это слово
20)
21)      fin.getline(buff, 50); // считали строку из
   файла
22)      fin.close(); // закрываем файл
23)      cout << buff << endl; // напечатали эту строку
24)      }
25)      system("pause");
26)      return 0;
27)  }
```

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

Константа	Описание
ios_base::in	открыть файл для чтения

Константа	Описание
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("text.txt", ios_base::app); // открываем файл для добавления информации к концу файла
fout.open("text.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 ifstream связываясь 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 перечислены далеко не все флаги, но для начала этих должно хватить.

Обратите внимание на то, что флаги ate и app по 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.

ВАРИАНТЫ ЗАДАНИЙ

1. Реализуйте программу, считывающую текст из файла и выводящую каждое слово с новой строки.
2. Реализуйте программу, считывающую текст с клавиатуры и записывающую его в файл.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Н.В. Зорина Курс лекций по Объектно-ориентированному программированию, Москва, МИРЭА, 2016
2. М. Эллис, Б. Строуструп. Справочное руководство по языку C++ с комментариями: Пер. с англ. - Москва: Мир, 1992. 445с.
3. Стенли Б. Липпман. C++ для начинающих: Пер. с англ. 2тт. - Москва: Унитех; Рязань: Гэлион, 1992, 304-345сс.
4. В.В. Подбельский. Язык C++: Учебное пособие. - Москва: Финансы и статистика, 1995. 560с.
5. Х. Дейтел, П. Дейтел. Как программировать на C++: Пер. с англ. - Москва: ЗАО "Издательство БИНОМ", 1998. 1024с.