Пермский филиал федерального государственного автономного
образовательного учреждения высшего образования
«Национальный исследовательский университет
«Высшая школа экономики»

Факультет экономики, менеджмента и бизнес-информатики

Чепоков Елизар Сергеевич

УПРАВЛЕНИЕ ДАННЫМИ

Реферат

студента образовательной программы «Программная инженерия»
по направлению подготовки *09.03.04 Программная инженерия*

Доцент кафедры
информационных
технологий в бизнесе
_____
Л. Н. Лядова

Пермь, 2020 год

Работа с текстовыми файлами в C++

Файлом называют поименованную последовательность байтов, хранящихся на внешних запоминающих устройствах.

В C++ для работы с файлами используются специальные типы данных, называемые *потоками*.

Двоичный поток – последовательность байтов, которые однозначно соответствуют тому, что находится на внешнем устройстве.

Текстовый поток – последовательность символов.

Файловая система ОС отвечает за выполнение следующих операций над файлами:

- создание
- уничтожение
- копирование
- перемещение на новое место
- переименование
- изменение значений атрибутов (установка их или отмена)
- поиск файлов по различным признакам
- открытие файлов для последующего
- чтение
- запись данных
- закрытие файлов после выполнения операций над ними.

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

Для работы с файлами в режиме, как чтения, так и записи служит библиотека **fstream**.

Класс ifstream служит для работы с файлами в режиме чтения.

Порядок считывания данных из текстового файла:

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

4. Закрыть файл.

Методы класса:

1. Open() – служит для открытия файлов, как один из способов открыть файл.

   С применением open:

   ```
   ifstream file; //создаем объект класса
   file.open("/путь к файлу/"); //открываем
   ```

   Без:

   ```
   ifstream file("/путь к файлу/"); // открываем файл в конструкторе
   ```

2. is_open() – служит для проверки на открытый файл. Заменяет логическую проверку:

   ```
   ifstream file("/путь к файлу/");
   if (!file){
       cout << "Файл не открыт";
       return -1;
   }
   else{
       cout << "Файл открыт";
       return 1;
   }
   ```

   На метод:

   ```
   ifstream file("/путь к файлу/");
   if (file.is_open())
       cout << " Файл открыт" << endl;
   else{
       cout << "Файл не открыт" << endl;
       return -1;
   }
   ```

3. >> - указывает в какую переменную будет произведено считывание

   ```
   ifstream file("/путь к файлу/");
   file >> d >> i >> s;
   ```

4. getline() и get() – служат для получения необходимого количества символов

   ```
   file.get(buffer, n); //чтение n символов
   file.getline(buffer, n, '1'); //n символов до первого = 1
   ```

5. read() – обычное чтение, применяется для неформатированного вводаи для чтения бинарных файлов.

   ```
   int n = 10;
   char* buf = new char[n + 1]; //add buffer
   buf [n] = 0;
   ifstream file("/путь к файлу/");
   file.read(text, n);
   ```

6. close() – закрывает файл. Не обязательная процедура при чтении файла, но иногда может повредится файл.

```
ifstream file("/путь к файлу/");
file.open();
file.close();
```

7. seekg() – меняет позицию чтения на указанную.

   Ios_base::end – отсчет позиции с конца файла

   beg – отсчет позиции с начала файла

   cur – отсчет новой позиции с текущей

```
file.seekg(0, ios_base::end); //конец файла
file.seekg(10, ios_base::end); //10 байт с конца

file.seekg(0, ios_base::beg); //начало
file.seekg(30, ios_base::beg); //31-й байт от начала

file.seekg(3, ios_base::cur); //3 байта вперед от текущей позиции
file.seekg(3); //3 байта вперед от текущей позиции
```

8. tellg() – сообщает о количестве прочитанного

```
ifstream file("/путь к файлу/");
    file.seekg(0, ios_base::end); //становимся в конец файла
    cout << "Размер файла (в байтах): " << file.tellg();
```

Класс ofstream служит для работы с файлами в режиме записи.

Порядок записи данных в текстовый файл:

1. Описать переменную типа ofstream.
2. Отрыть файл с помощью функции open.
3. Вывести информацию в файл.
4. Обязательно закрыть файл.

Методы класса:

1. open() – открывает файл для записи

```
ofstream file;
file.open("/путь к файлу/");
```

2. is_open() – служит для проверки на открытый файл. Заменяет логическую проверку.

```
ifstream file("/путь к файлу/");
if (file.is_open())
    cout << " Файл открыт" << endl;
else{
    cout << "Файл не открыт" << endl;
    return -1;
}
```

3. write() – производит запись в файл

```
ofstream file;
    file.open("/путь к файлу/");
```

```
    int k = 10; //любое целое
    file.write((char*)&k, sizeof(k));
```

4. close() – закрывает файл. Обязательная процедура при записи файла

```
ofstream file;
file.open("/путь к файлу/");
file.close();
```

5. width() – указывает количество символов для выводимого значения

```
ofstream file;
file.open("/путь к файлу/");
file.width(20);
file << text;
```

6. precision() – указывает количество знаков после запятой для вещественных чисел

```
ofstream file;
file.open("/путь к файлу/");
file.width(20);
file.precision(5);
file << text;
```

7. seekp() – схож с seekg, меняет позицию чтения на указанную.
8. tellp() – получает данные о позиции от начала файла в байтах.

Обработка двоичных файлов

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

При записи в двоичный файл. В общем случае оператор открытия потока символы и числа записываются в виде последовательности байт.

Порядок работы с двоичными и текстовыми файлами аналогичен.

Для записи данных в двоичный файл необходимо:

1. Описать файловую переменную типа с помощью оператора FILE *filename;
2. Отрыть файл с помощью функции fopen.
3. Записать информацию в файл с помощью функции fwrite.
4. Закрыть файл с помощью функции fclose.

Для чтения данных из двоичного файл необходимо:

1. Описать переменную типа FILE *.
2. Отрыть файл с помощью функции fopen.

3. Считать необходимую информацию из файла с помощью функции fread, при считывании информации следить за тем, достигнут ли конец файла.
4. Закрыть файл с помощью функции fclose.

Структура использования:

1. FILE *fopen(const *filename, const char *mode); - открытие файла. После открытия файла указатель файла указывает на 0-й байт файла, а по мере чтения или записи смещается на считанное (записанное) количество байт
   a. *filename – полное имя открываемого файла,
   b. *mode – определяет режим работы с файлом, принимает следующие значения:
      - rb – открыть двоичный файл в режиме чтения;
      - wb – создать двоичный файл для записи, если файл существует, то его содержимое очищается.
      - ab – создать или открыть двоичный файл для добавления информации в конец файла;
      - rb+ – открыть существующий двоичный файл в режиме чтения и записи;
      - wb+ – открыть двоичный файл в режиме чтения и записи, существующий файл очищается;
      - ab+ – двоичный файл открывается или создается для исправления существующей информации и добавления новой в конец файла.
2. int fclose(FILE *filename) - закрытие файла. Возвращает 0 при успешном закрытии файла и NULL в противном случае.
3. int remove(const char *filename) – удаление файлов. Удаляет с диска файл с именем *filename. Удаляемый файл должен быть закрыт. Функция возвращает ненулевое значение, если файл не удалось удалить.
4. int rename(const char *oldfilename, const char *newfilename) – переименование файлов *oldfilename – старое имя файла, *newfilename – новое. Функция возвращает 0 при удачном завершении программы.
5. fread (void *ptr, size, n, FILE *filename) – чтение из двоичного файла. Функция считывает из файла в массив *ptr n элементов. Функция возвращает количество считанных элементов.

6. fwrite (const void *ptr, size, n, FILE *filename) – запись в двоичный файл. Записывает в файл filename из массива *ptr n элементов размера size.
7. int feof(FILE * filename) – сообщает о конце файла. Возвращает ненулевое значение, если достигнут конец файла.

Пример использования:

```
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>

using namespace std;

int main()
{
	FILE *file;
	int n, k;
	double a;
	double * buf = new double[];

	file = fopen("file.dat", "wb");		// Создание двоичного файла
	cout << "n="; cin >> n;
	fwrite(&n, sizeof(int), 1, file);	// Запись числа в двоичный файл

	for (int i = 0; i < n; i++)	{
		cout << "a= "; cin >> a;
		fwrite(&a, sizeof(double), 1, file); // Запись числа в двоичный файл
	}
	file = fopen("test.dat", "rb");		// Открытие файла
	fread(&k, sizeof(int), 1, file);
	cout << "n=" << k << "\n";		// Вывод целого на экран
	buf = new double[k];
	fread(buf, sizeof(double), k, file);  // Чтение k вещественных чисел в буфер
	for (int i = 0; i < k; cout << buf[i] << "\t", i++); // Вывод массива на экран
		cout << endl;

	fclose(file);				// Закрыть файл
	cin >> k;
	return 0;

}
```