# Работа с файлами

8 июня 2017 г.

# Класс fstream

- за работу с файлами отвечает класс fstream:
  - класс ifstream – файловый ввод
  - класс ofstream – файловый вывод
  - наследуется от класса iostream

```
#include <fstream>
```

# Файловый вывод

- создать объект класса ofstream
- связать объект с файлом для записи (open)
- записать данные в файл
- закрыть файл (close)

```
#include <fstream>

std::ofstream fileout;
fileout.open("myfile.txt");
// запись данных
fileout.close();
```

# Файловый ввод

- создать объект класса ifstream
- связать объект с файлом для записи (open)
- проверить, открыт ли файл (is_open)
- прочитать данные из файла
- закрыть файл (close)

```
#include <fstream>

std::ifstream filein;
filein.open("myfile.txt");
if ( !filein.is_open() ) { return; }
// чтение данных
filein.close();
```

# Режимы открытия файла

| Флаг | Значение |
|---|---|
| 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 файла

- флаги указываются вторым аргументом в методе open
- можно задавать несколько флагов через |
- ifstream – по умолчанию std::ios_base::in
- ofstream – std::ios_base::out | std::ios_base::trunc

```
#include <fstream>

std::ifstream filein;
filein.open("myfile.txt", std::ios_base::in|std::ios_base::binary);
if ( !filein.is_open() ) { return; }
// чтение данных
filein.close();
```

# Открытие нескольких файлов

**одновременно**

- для каждого файла создается отдельный поток

**последовательно**

- создать один поток и по очереди ассоциировать его с каждый новым файлом
- после обработки каждого следующего файла закрыть файл (close), вызвать метод clear и потом открыть следующий файл

# Последовательное открытие нескольких файлов (пример)

```
#include <fstream>

void f() {
    std::ifstream filein;
    filein.open("myfile1.txt");
    if ( !filein.is_open() ) { return; }
    // чтение данных
    filein.close();

    filein.clear();  // некоторым компиляторам не нужен

    filein.open("myfile2.txt");
    if ( !filein.is_open() ) { return; }
    // чтение данных
    filein.close();
}
```

# Состояние потока

| Метод | Описание |
|---|---|
| good() | поток в нормальном состоянии |
| eof() | достигнут конец файла |
| bad() | ошибка при работе с файлом (файл не открыт, нет свободного места и т. п.) |
| fail() | ошибка при работе с файлом (те же случаи, что и для bad(), а также ошибка с форматом данных |

- метод clear() сбрасывает флаг состояния

# Проверка текущей позиции

- метод teelg – для входных потоков
- метод teelp – для выходных потоков
- методы возвращают значение типа streampos (текущая позиция в байтах, измеренная от начала файла)

# Произвольный доступ

- метод seekg – перемещает в заданную позицию файла указатель ввода
- метод seekp – перемещает в заданную позицию файла указатель вывода
- аргументы:
  - 1) смещение в байтах от начала файла (streampos)

    или
  - 1) смещение в байтах; 2) стартовая позиция (std::ios_base::beg – начало, std::ios_base::cur – текущая позиция, std::ios_base::end – конец файла)

# Произвольный доступ (пример 1)

```
#include <fstream>

void f() {
	std::ifstream filein;
	filein.open("myfile1.txt");
	if ( !filein.is_open() ) { return; }
	filein.seekg(0, std::ios_base::end);
	// чтение данных
	filein.seekg(0, std::ios_base::beg);
	filein.close();
}
```

# Произвольный доступ (пример 2)

```
#include <fstream>
#include <iostream>


void f() {
	std::streampos begin, end;
	std::ifstream filein("myfile1.txt");
	if ( !filein.is_open() ) { return; }
	begin = filein.tellg();
	filein.seekg(0, std::ios_base::end);
	end = filein.tellg();
	filein.close();
	std::cout << "size is: " << (end - begin) << "bytes.\n";
}
```

# Ввод-вывод

| Метод | Описание |
|---|---|
| getline(s, num) | читает до num-1 символов;<br>ограничитель – новая строка (с включением) или конец файла |
| read(s, num) | читает num символов; ограничитель – конец файла |
| get(s, num) | читает до num-1 символов;<br>ограничитель – новая строка (без включения) или конец файла |
| put(char c) | записывает символ в поток |
| >> и << | ввод и вывод последовательности символов; ограничитель – пробел или новая строка |

# Вопросы?