C++ Библиотека ввода/вывода

В C++ имеются две библиотеки ввода/вывода

- ☞ **«старая», в C-стиле:** `<cstdio>` с функциями `printf(), scanf() ...`
- ☞ **«новая», объектно-ориентированная:** `<iostream>` из библиотеки C++

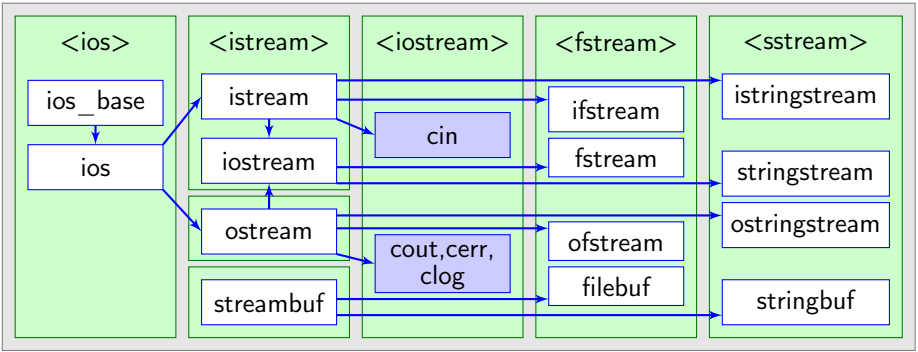
Преимущества <iostream>

- ✓ **Расширяемость:** легко добавить операторы $\ll$ и $\gg$ для пользовательских классов
- ✓ **Наследуемость:** стандартный набор классов для потоков можно наследовать и строить собственные классы

Преимущество <cstdio>

- ✓ **Эффективность:** ввод-вывод в `cstdio` работает быстрее чем в `iostream`

Цепочка наследования классов в IOStream



- `<ios_base>` — базовые классы
- `<istream>, <iostream>` — основные устройства ввода/вывода
- `<fstream>` — ввод/вывод в файлы
- `<sstream>` — ввод/вывод в C++ `string`

Основные сведения о потоках

Stream (поток): в простейшем случае последовательность символов

- Класс **istream** определяет поток для чтения
- Класс **ostream** определяет поток для записи

Глобальные потоковые объекты

- `cin` – поток ввода; аналог `stdin` в C, буфуризованный
- `cout` – поток вывода; аналог `stdout` в C, буферизованный
- `cerr` – поток сообщений об ошибках; аналог `stderr` в C, небуферизованный
- `clog` – буферизованный вариант `cerr`

Операторы вывода: перегружаются операторы побитового сдвига

- `<<` вывод в поток (inserter)
- `>>` ввод (extractor)

Манипуляторы

- ☞ Специальные потоковые объекты для управления вводом/выводом

основные манипуляторы определены в `<iostream>`

`endl` — (end of line) выводит `'\n'` и очищает буфер

`flush` — очистка буфера

`ends` — (end of string) выводит `'\0'`

`ws` — (white space) читает «пробелы» и пропускает их

Манипуляторы

- **некоторые манипуляторы из заголовочного файла** `<ios>`
  - `dec,hex,oct` – переключатели системы счисления для целых
  - `showbase, noshowbase` – переключатель для вывода префикса системы счисления
  - `fixed,scientific` – переключатели для чисел с плавающей точкой
  - `showpos, noshowpos` – переключатель для вывода знака +

- **некоторые манипуляторы из заголовочного файла** `<iomanip>`
  - `setw()` – ширина следующего вывода или ввода
  - `setfill()` – меняет символ заполнитель
  - `setprecision()` – количество знаков для чисел с плавающей точкой
  - `setbase()` – изменяет базу системы счисления для целых чисел

Пользоваться манипуляторами не слишком удобно:

```
cout << hex << setfill('0') << setw(8) << i << dec << endl;
printf("0x%08x\n", i);

cout << scientific << setprecision(3) << setw(10) << x << fixed
     << setprecision(15) << setw(20) << y << setprecision(6) << endl;
printf("%10.3e %20.15f\n",x,y);
```

☞ Многие манипуляторы (`hex`,`setprecision()`, `...`) меняют поведение всего потока и возврат в значение по умолчанию требует повторного вызова

Файловые потоки

Чтение из файла и запись в файл

```
#include <fstream>                // file I/O
ifstream file_in("input.dat"); // открыть файл для чтения
int r;
file_in >> r;
cout << "r= " << r << endl;
ofstream file_out("output.out"); // открыть файл для звписи
file_out << (r+1) << endl;
```

Возможные ошибки

1. Что будет если файл `input.dat` отсутствует?
2. Что случится если в этом файле ошибочный формат данных?
3. Как проверить, что программа не перепишет уже существующий файл `output.out`?

Работа над ошибками

1. нет файла `input.dat`

☛ Output: r= 0

Проверка, что из потока можно читать

```
const char* input = "input.dat";
ifstream file_in(input);
if( !file_in ) { // check the stream is in good condition
     cerr << "can not open file: " << input << endl;
     exit(EXIT_FAILURE);
}
```

☞ `if( !file_in )` — проверка состояния потока

☞ `if( !file_in.is_open() )` — проверка связан ли поток с файлом

2 файл input.dat пустой или «неправильный»

☛ Output: r= 0

```
file_in >> r;
ios::iostate state = file_in.rdstate(); // проверка ошибок в потоке
if( state ) {
  if( state & ios::eofbit ) { // true если конец файла
      cerr << " end of file " << endl;
      exit(EXIT_SUCCESS);
   } else {  // другие ошибки чтения
      cerr << "error reading file" << endl;
      exit(EXIT_FAILURE);
   }
}
```

- eofbit – конец файла, больше читать нечего
- badbit – поток больше не функционален (read/writing error)
- failbit – сбой чтения возможно из-за неожиданных символов (logical error)

3. **файл `output.dat` уже существует**

☛ файл `output.dat` перепишется

Проверка, что файл существует для UNIX подобных систем

```
#include <sys/stat.h> // get file status: see man 3 stat
string output("output.out");
struct stat buff;
if( stat(output.c_str(), &buff) == 0 ) { // not C++, but POSIX
   cerr << " file " << output << " already exist" << endl;
   exit(EXIT_FAILURE);
}
ofstream file_out(output);
```

☞ В C++17 появилась библиотека файловой системы