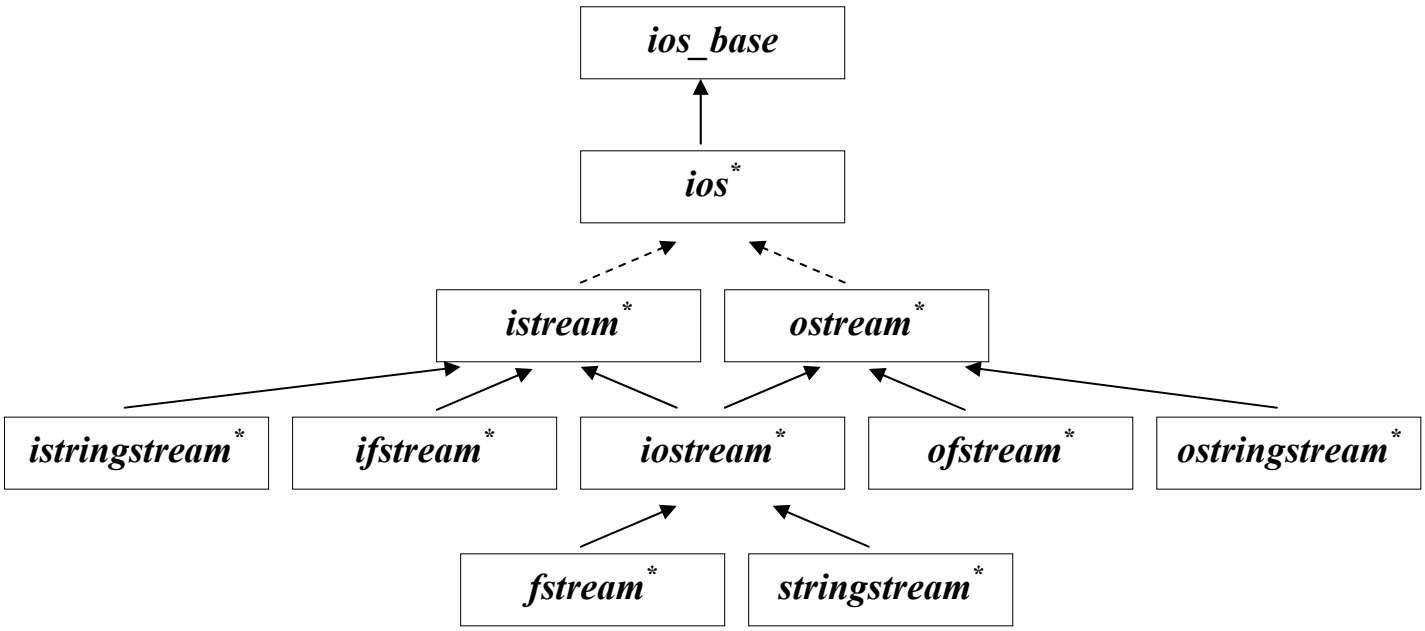
Текстовые файлы. Стиль C++

В программе на C++ стандартные потоки ввода-вывода ***cout***, ***cerr***, ***clog*** и ***cin*** с их эквивалентами для расширенных символов доступны по умолчанию, а их связь с устройствами ввода-вывода или файлами определяется “системой”. Кроме того можно создавать свои собственные потоки. В этом случае необходимо указать, к чему их следует прикрепить. Зачастую поток прикрепляют к файлу или к строке ***string***, и поэтому такие действия поддерживаются стандартной библиотекой.

Представим иерархию классов стандартных потоков:



Пунктирная линия означает виртуализацию базовых классов, а * – классы-шаблоны, параметры которых имеют символьный тип, а их имена начинаются с префикса ***basic_***.

Файлы и строки – это примеры контейнеров, из которых можно считывать и в которые можно записывать. Следовательно, можно завести поток, поддерживающий как оператор вывода <<, так и оператор ввода >>.Такой поток называется ***iostream***, он определён в пространстве имён ***std*** и представлен в ***<iostream>***:

```
template <class Ch, class Tr = char_traits<Ch>>
class basic_iostream : public basic_istream<Ch, Tr>, public basic_ostream<Ch, Tr>
{
   public:
      explicit basic_iostream(basic_streambuf<Ch, Tr>* sb);
      virtual ~basic_iosream();
   // …
};
typedef basic_iostream<char> iostream;
typedef basic_iostream<wchar_t> wiostream;
```

Управление чтением и записью осуществляется через две операции над буфером ***streambuf*** потока ***iostream*** – “записать в буфер” и “прочесть из буфера”.

Класс ***basic_ofstream*** объявлен в ***<fstream>***:

```
template <class Ch, class Tr = char_traits<Ch>>
class basic_ofstream : public basic_ostream<Ch, Tr>
{
```

```
    public:
        basic_ofstream();
        explicit basic_ofstream(const char* p, openmode m = out|trunc);
        // …
        void open(const char* p, openmode m = out|trunc);
        void close();
        // …
};
typedef basic_ofstream<char> ofstream;
typedef basic_ofstream<wchar_t> wofstream;
```

Класс ***basic_ifstream*** похож на класс ***basic_ofstream*** за исключением того, что он является производным от класса ***basic_istream*** и предназначен для чтения.

Как обычно, доступны определения типов с помощью ***typedef***:

```
typedef basic_ifstream<char> ifstream;
typedef basic_ifstream<wchar_t> wifstream;
```

Стандартная библиотека также предлагает класс ***basic_fstream***, похожий на классы ***basic_ifstream*** и ***basic_ofstream*** за исключением того, что он является производным от класса ***basic_iostream*** и предназначен для чтения и записи.

Как обычно, доступны определения типов с помощью ***typedef***:

```
typedef basic_fstream<char> fstream;
typedef basic_fstream<wchar_t> wfstream;
```

Второй аргумент в конструкторах файловых потоков указывает, как открывается файл:

```
class ios_base
{
    public:
        // …
        static openmode app,        // добавить в конец файла
                        ate,        // открыть и искать конец файла
                        binary,     // ввод-вывод в двоичном виде
                        in,         // открыть для чтения
                        out,        // открыть для записи
                        trunc;      // урезать файл до нулевой длины
        // …
};
```

Класс ***basic_ostream*** определяет оператор << (“записать в”) для управления выводом объектов встроенных типов:

```
template <class Ch, class Tr = char_traits<Ch>>
class basic_ostream : virtual public basic_ios<Ch, Tr>
{
    public:
        // …
        basic_ostream& operator<<(short int n);
        basic_ostream& operator<<(int n);
        basic_ostream& operator<<(long int n);
        basic_ostream& operator<<(unsigned short int n);
        basic_ostream& operator<<(unsigned int n);
```

```
        basic_ostream& operator<<(unsigned long int n);
        basic_ostream& operator<<(float f);
        basic_ostream& operator<<(double f);
        basic_ostream& operator<<(long double f);
        basic_ostream& operator<<(bool n);
        basic_ostream& operator<<(const void* p);
        // …
        basic_ostream& put(Ch c);
        basic_ostream& write(const Ch* p, streamsize n);
        // …
};
```

Класс ***basic_istream*** определяет оператор >> (“прочесть из”) для управления вводом объектов встроенных типов:

```
template <class Ch, class Tr = char_traits<Ch>>
class basic_istream : virtual public basic_ios<Ch, Tr>
{
   public:
        // …
        basic_istream& operator>>(short int& n);
        basic_istream& operator>>(int& n);
        basic_istream& operator>>(long int& n);
        basic_istream& operator>>(unsigned short int& n);
        basic_istream& operator>>(unsigned int& n);
        basic_istream& operator>>(unsigned long int& n);
        basic_istream& operator>>(float& f);
        basic_istream& operator>>(double& f);
        basic_istream& operator>>(long double& f);
        basic_istream& operator>>(bool& n);
        basic_istream& operator>>(void*& p);
        // …
        int_type get();
        basic_istream& get(Ch& c);
        basic_istream& get(Ch* p, streamsize n);
        basic_istream& get(Ch* p, streamsize n, Ch term);
        basic_istream& getline(Ch* p, streamsize n);
        basic_istream& ignore(streamsize n = 1, int_type t = Tr::eof());
        basic_istream& read(const Ch* p, streamsize n);
        // …
};
```

Примечание. Оператор >> пропускает символы-разделители (пробел, табуляция, новая строка, новая страница и возврат каретки).

Создание пустого и непустого текстовых файлов. Версия 1

```
// Programming in C++ Style

#include <iostream>
#include <fstream>

int main()
{
  using namespace std;
  ofstream outFile;
  outFile.open("empty.txt");
  if (!outFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
    outFile.close();
  outFile.open("text.txt");
  if (!outFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
  {
    outFile.write("Hi!\n", 4);
    outFile.write("So long!\n", 9);
  }
  return 0;
}
```

empty.txt 0 КБ

text.txt 1 КБ

```
Hi!
So long!
```

Создание пустого и непустого текстовых файлов. Версия 2

```
// Programming in C++ Style

#include <iostream>
#include <fstream>

int main()
{
  using namespace std;
  ofstream outFile;
  outFile.open("empty.txt");
  if (!outFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
    outFile.close();
  outFile.open("text.txt");
  if (!outFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
  {
    outFile << "Hi!\n";
    outFile << "So long!\n";
  }
  return 0;
}
```

Пустой текстовый файл
Исследование механизма доступа

```
// Programming in C++ Style

#include <iostream>
#include <fstream>

int main()
{
  using namespace std;
  const int n(80);
  char line[n];
  ifstream inFile("empty.txt");
  if (!inFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
    while(!inFile.eof())
      if (!inFile.getline(line, n))
      {
        cout << "Invalid reading file!\n";
        if (inFile.eof()) cout << "The End-Of-File is set!\n";
      }
  return 0;
}
```

Результат работы программы:

```
Invalid opening file!
```

Результат работы программы:

```
Invalid reading file!
The End-Of-File is set!
```

Текстовый файл
Исследование механизма доступа

```
// Programming in C++ Style

#include <iostream>
#include <fstream>

int main()
{
  using namespace std;
  const int n(80);
  char line[n];
  ifstream inFile("text.txt");
  if (!inFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
    while(!inFile.eof())
      if (!inFile.getline(line, n))
      {
        cout << "Invalid reading file!\n";
        break;
      }
      else
        cout << line << endl;
  return 0;
}
```

Результат работы программы:

```
Invalid opening file!
```

Результат работы программы:

```
Hi!
So long!
Invalid reading file!
```

Текстовый файл
Исследование механизма доступа

```
// Programming in C++ Style

#include <iostream>
#include <fstream>

int main()
{
  using namespace std;
  const int n(80);
  char line[n];
  ifstream inFile("text.txt");
  if (!inFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
    while(!inFile.eof())
      if (inFile.getline(line, n)) cout << line << endl;
  return 0;
}
```

Результат работы программы:

```
Invalid opening file!
```

Результат работы программы:

```
Hi!
So long!
```

Текстовый файл
Исследование механизма доступа

```
// Programming in C++ Style

#include <iostream>
#include <fstream>

int main()
{
  using namespace std;
  const int n(80);
  char line[n];
  ifstream inFile("text.txt");
  if (!inFile)
  {
    cout << "Invalid opening file!\n";
    return -1;
  }
  else
    while(inFile.getline(line, n)) cout << line << endl;
  return 0;
}
```

Результат работы программы:

```
Invalid opening file!
```

Результат работы программы:

```
Hi!
So long!
```