Seminar 18.

20 марта 2023 г.

1 Как организована иерархия классов потоков в библиотеке IOStream?

Иерархия классов потоков в библиотеке IOStream (все классы, кроме "родоначального"ios_base имеют специализации для char и wchar_t, обозначенные с использованием typedef; в то же время на данный момент отсутствуют специализации для типов char16_t и char32_t):

- Существует базовый класс, ios_base, определяющий общие возможности и функциональность всех потомков: он задаёт состояние потока и флаги потока.
- Напрямую от ios_base наследуется шаблонный класс basic_ios<>, предоставляющий некоторые средства интерфейса для взаимодействия с объектами.
- Наследниками (динамический полиморфизм) basic_ios<> являются basic_istream<> и basic_ostream<>, которые обеспечивают поддержку высокоуровневых операций ввода и вывода (соответственно) над символьными потоками: поддерживаемые операции включают форматированный и неформатированный ввод и вывод.
- От этих двух классов наследуется basic_iostream<>, объединяющий многие возможности родителей.
- От классов basic_istream<>, basic_ostream<> и basic_iostream<> наследуются basic_istringstrea basic_ostringstream<> и basic_stringstream<> соответственно для работы со строками, а также basic_ifstream<>, basic_ofstream<> и basic_fstream<> соответственно для работы с файлами.

Кроме того, важной частью библиотеки IOStream является класс basic_streambuf<> и его наследники basic_stringbuf<> и basic_filebuf<>. Класс basic_streambuf управляет вводом и выводом последовательностей символов: обеспечивает доступ к управляемой последовательности символов, называемой буфером, которая может содержать входную последовательность для буферизации операций ввода и/или выходную последовательность для буферизации операций вывода; а также обеспечивает доступ к связанной последовательности символов, называемой источником (для ввода) или приёмником (для вывода). Это может быть объект, доступ к которому осуществляется через API OS (например, файл) или это может быть объект (std::vector, массив, строковый литерал), который можно интерпретировать как источник или накопитель символов. Объекты потока ввода-вывода basic_istream и basic_ostream, а также все производные от них объекты полностью реализованы в терминах basic_streambuf.

Более удобными для восприятия иерархии являются рисунки fig1 и fig2.

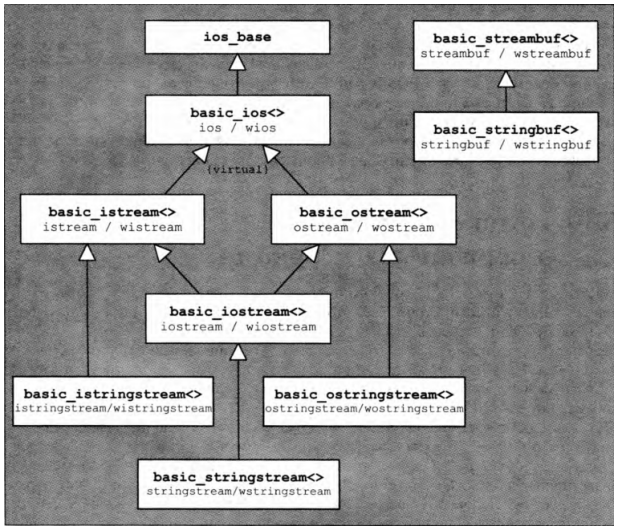

Рис. 1: Иерархия классов потоков в библиотеке IOStream.

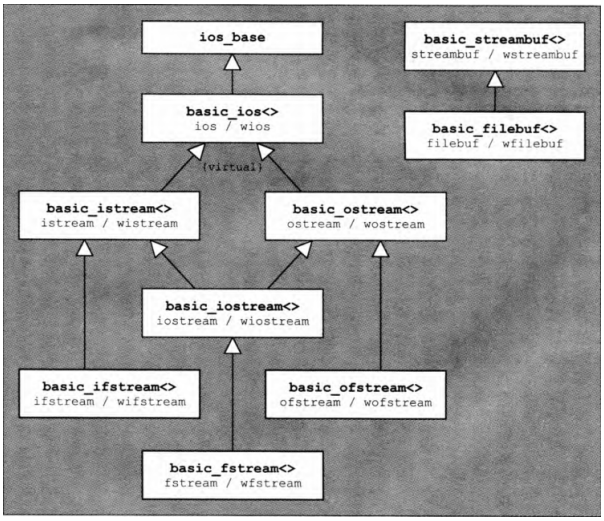

Рис. 2: Иерархия классов потоков в библиотеке IOStream.

2 Какие состояния потоков реализованы в базовом классе basic_ios?

Состояния потоков, реализованые в базовом классе basic_ios:

- good bit – всё в порядке;
- eofbit – конец файла;
- failbit – сбой при вводе/выводе (не фатальная ошибка);
- badbit – фатальная ошибка.

Стоит отметить, что в случая сбоя в потоке его состояние нужно сбрасывать вручную, с помощью clear.

3 В чём разница между манипуляторами и флагами форматирования?

Манипуляторы – это вспомогательные функции, управляющие потоком данных обычно локально (т.е. в рамках одной команды, в месте использования operator»/operator«). Флаги форматирования же позволяют делать глобальные настройки для каких-либо потоков (т.е. эти настройки будут сохраняться для данного потока на протяжении выполенния всей программы после настройки/до новых изменений флагов). Т.о., флаги и манипуляторы выполняют одну и туже задачу – задают определённый формат ввода-вывода информации в потоках, но используются по-разному.

4 Из каких основных элементов состоят пути в файловой системе?

Путь – набор символов, показывающий расположение файла или каталога в файловой системе. Пути могут быть абсолютными или относительными в зависимости от наличия тех или иных компонент. Основные элементы пути:

- Названия корня (в зависимости от типа пути), директории/ий (каталоги) и файла.
- Разделительный знак. В OS UNIX разделительным знаком при записи пути является (/), в Windows – (\) (обычно). Эти знаки служат для разделения названия каталогов, составляющих путь к файлу. Кроме того, в случае Windows может быть также использован разделитель томов (:) для формирования абсолютного пути к файлу из корня диска. Также в Windows существуют разделители (.) и (..).
- Кодировка имени файла. В Unix/Linux имя файла представляет собой последовательность любых байтов, кроме косой черты или NUL. Современные среды Unix/Linux прекрасно обрабатывают имена файлов в кодировке UTF-8. В Windows для этих целей используется UTF-16. Это определяет использование типов char и wchar_t в Linux и Windows соответственно.

5 Зачем нужны форматы обмена данными, такие как JSON и XML?

Форматы обмена данными удобны для чтения и написания как человеком, так и компьютером. С их помощью успешно создаётся разметка в документах и текстах, где доля разнотипных символьных данных велика, а доля разметки мала. Более того, форматы обмена данными не зависят от языка программирования и часто основываются на универсальных структурах данных, поддерживаемых многими современными языками программирования в какой-либо форме. В частностти, они позволяют лаконично производить сериализацию и десериализацию (при передаче объектов по сети и для сохранения их в файлы).

Список литературы

[1] https://en.cppreference.com/w/

[2] https://www.json.org/json-en.html