C++: Streams

Pierre-Alain Fayolle

Table of contents

Introduction

- streams are C++'s way of formatting input and output to and from a variety of sources (console, file, string)
- streams will be used in several places in this course; mostly for writing data to the standard output (the console)
- this section is a simplified presentation on what streams are and how to use them but this will be sufficient for our purposes

Introduction

- streams are part of the standard library; as such they are in the namespace std
- when using streams, you need to:
  - include the corresponding header: <iostream> (IO streams), <fstream> (File streams) or <sstream> (String streams)
  - either prefix symbols by std:: or use the directive 'using namespace std;'
- example:

```
std::cout << "write to stdout" << std::endl;
```

- or:

```
// e.g. at the beginning of the file
using namespace std;
//...
cout << "write to stdout" << endl;
```

IO streams

- the C++ standard library provides IO streams objects for reading data from the standard input and writing data to the standard output (i.e. the console)
- IO stream types are part of the standard library, in the namespace std
- all IO streams related symbols are available by including the header <iostream>

Output streams

- the standard library provides a stream object called cout (for character output) for writing formatted data to the standard output
- example:

```
cout << ‘‘Programming in C++’’ << endl;
```

- Here $\ll$ operator is called the stream insertion operator and is used to push data into a stream (in this case the standard output stream)
- endl is a function (a stream manipulator) that adds an end of line symbol (“$\backslash n$”) in the stream

Input streams

- the standard library provides a symmetric stream object for reading values from the standard input: cin (for character input)
- to read a value from cin, the stream extraction operator $\gg$ should be used
- example:

```
cout << ‘‘Enter a number:’’ << endl;
int number;
cin >> number;
```

Note that when using cin, there is no endl at the end. The following is illegal:

```
cin >> number >> endl; // Error
```

IO streams

- it is possible to read or write several values by chaining extraction or insertion operators
- example:

```
cout << "Enter your name and a number:" << endl;
string name;
int number;
cin >> name >> number;
cout << "Mr " << name << " your number is: "
      << number << endl;
```

File streams

- the standard library provides file streams that are used to write and read from files
- File streams and IO streams are manipulated the same way; i.e. with stream objects and extraction or insertion operators
- the header file fstream (File STREAM) exports the stream types: ifstream and ofstream
- ifstream is the stream type for reading data from a file
- ofstream is the stream type for writing data to a file

Input file streams

- ifstream is the stream type for reading from a file
- ifstream stands for Input File STREAM
- to create an ifstream object in order to read from a file either:

```
ifstream input("file.txt");
```

- or:

```
ifstream input;
input.open("file.txt");
```

- after using close the stream with the method .close()

Example

```
ifstream input("file.txt");
if (!input.is_open()) {
  cerr << "can not open file" << endl;
  exit(1);
}
int number;
input >> number;
input.close();
```

Input file streams

- first line creates a new ifstream object named input that reads from file.txt
- in order to read from the file, the file should exist (and has at least read permission on the file system)
- use the method is_open() on an ifstream object (see line 2) to check if a file could be opened properly
- note that the error is reported to cerr, which is like cout but designed for error reporting and sometimes handled differently by the operating system
- a number (an integer) is read from the file with the extraction stream operator; note the similarity with reading from cin above
- finally, the input stream is closed.

Output file streams

- the counterpart of ifstream is ofstream
- example:

```
ofstream output("save.txt");
```

- if the file save.txt does not exist, it will be created
- if the file save.txt already exists, it will be overwritten
- if you want to append data to an already existing file, the file should be opened with the proper openmode. e.g.:

```
ofstream output("save.txt", ios_base::app);
```

- after using output file streams, they should be closed with the method .close()

File streams

- in C++ programming, it is preferrable to manipulate string object than char* (the type string is part of the C++ standard library and declared in the header <string>)
- file stream constructors (and the open methods) however take as arguments C-style strings (char *); you can convert a string object to a C-style string with the string method .c_str()
- idiomatic example:

```
string fileName = "file.txt";
ifstream input(fileName.c_str());
```

String streams

- String streams are the last type of C++ streams that we will see
- it allows to manipulate string objects like any other streams
- the type stringstream is exported in the header <sstream>
- it is part of the standard library; in the namespace std
- stringstream objects are useful for constructing a string object composed of plain text and data (e.g. numbers)

Example of utilisation

- ideally we would like to form a string like this:

```
string data;
data = data + "append some data " + 122;
cout << data << endl;
```

- while this would work in Java , it is illegal in C++
- instead we need to create a stringstream object and insert data in it:

```
stringstream ssdata;
ssdata << "append some data " << 122;
cout << ssdata.str() << endl;
```

- to convert a stringstream object back to a string object, the method .str() is used (as illustrated above)
- stringstream objects are generally used as temporary string buffers

Stream manipulators

- a stream manipulator is a function or an object that can be inserted into a stream to change its property
- endl is an example of stream manipulator that inserts a end of line symbol in the stream
- example:

```
cout << "C++ programming" << endl;
```

- is equivalent to:

```
cout << "C++ programming\n";
```

- stream manipulators are in the namespace std, and declared in the header <iomanip>

Stream manipulators

- boolalpha: prints a boolean (true or false) instead of a digit (1 or 0)

```
cout << true << endl; // 1
cout << boolalpha << true << endl; // true
```

- hex, dec, oct: encode numbers in the stream in hexadecimal, decimal or octal form (default is decimal)

```
cout << 10 << endl; // 10
cout << hex << 10 << endl; // a
cout << oct << 10 << endl; // 12
```

Stream manipulators

- setw(n): padds with spaces so that the stream takes at least n characters

```
cout << "hello" << endl; // hello
cout << setw(10) << "hello" << endl; //      hello
```

- setfill(c): fills the stream with the character c instead of spaces (used with setw())

```
cout << setfill('0') << setw(5)
     << 11 << endl; // 00011
```