

(6 – 2) Streams and File Processing in C++

Instructor - Andrew S. O'Fallon
CptS 122 (February 16, 2018)
Washington State University

What is a Stream? A Refined Definition

- A *sequence* of objects (generally just considered bytes) that flow from a device to memory or from memory to a device
- For *input* operations, the bytes flow from the device (i.e. keyboard, network connection, disk, etc.) to main memory
- For *output* operations, the bytes flow from main memory to the device (screen, printer, etc.)



Analogy for a Stream

- A conveyer belt
 - You can place an item in sequence on the belt, i.e. into the stream (insertion or output operation)
 - You can remove an item in sequence from the belt, i.e. take from the stream (extraction or input operation)



Classic Streams vs. Standard Streams

- The *classic* input/output streams for C++ supported byte-sized `chars`, which represented the ASCII characters
- Many alphabets require *more* characters than can be represented by a *byte* and the ASCII character set does not provide the characters
 - The *Unicode* character set provides these ones
- C++ provides *standard* stream libraries to process Unicode characters (`wchar_t`)



Standard Streams in C++ (1)

- For *standard* input/output streams, include `<iostream>`
 - `cin` is a predefined *object* of class `istream` and is connected to the standard input device (i.e. keyboard)
 - `cin >> var // cin applying stream extraction operator - stops at whitespace for strings`
 - `cout` is a predefined *object* of class `ostream` and is connected to the standard output device (i.e. screen)
 - `cout << var // cout applying stream insertion operator`



Standard Streams in C++ (2)

- *Member* function `getline()` will read a line from the stream
 - Inserts a null character at the end of the array of characters, removes and discards the '\n' from the stream (i.e. stored as a C string)



Recall the File Processing Algorithm!

- Step 1: open the desired file
 - Opening is based on filename and permissions (read, write, or append)
 - Associates a file with a stream
- Step 2: process the file
 - Read data from the file
 - Does not affect file
 - Write data to the file
 - Completely overwrites existing file
 - Add data to the end of the file
 - Retains previous information in file
- Step 3: close the file
 - Disassociates a file from a stream



Files Streams in C++ (1)

- For input/output streams to work with *files*, include `<fstream>`
 - `ifstream` objects enable input from a file
 - `ofstream` objects enable output to a file
 - `fstream` objects for input from and output to a file
- Associate file with a file stream either during construction (applying the constructor or by calling `open()`)
 - `fstream fstr("filename.txt")` // an instantiation of `fstream` object **or** `fstr.open("filename.txt")` // after instantiation



Files Streams in C++ (2)

- Read from files using:

- `fstr >> var; //` applying the stream extraction operator - stops at whitespace for strings
- `fstr.getline () //` to read entire line into a character array

- Stored as a C string

- Write to files using:

- `fstr << var; //` applying the stream insertion operator



Files Streams in C++ (3)

- Each file ends with an end-of-file marker (EOF)
 - check if at end of file using `fstr.eof()`
- Close a file using:
 - `fstr.close()` ;



Closing Thoughts on Files

- Files are required for many applications
- Files may be created and manipulated in any manner appropriate for an application



References

- P.J. Deitel & H.M. Deitel, *C++: How to Program* (9th ed.), Prentice Hall, 2014
- J.R. Hanly & E.B. Koffman, *Problem Solving and Program Design in C* (8th Ed.), Addison-Wesley, 2016



Collaborators

- Jack Hagemeister

