C++ course – Exercises Set 5

Wouter Verkerke (Jan 2023)

Exercise 5.1 – Reading a file

The goal of this exercise is to read a text file with integers all the way to the end.

- Write a program that opens the file `ex5.1/data1.txt` using the class `ifstream`.

- Add code that verifies that the file was opened OK. Check that this code works by temporarily changing the name of the file to be opened to a different – non-existent – one, e.g. `data2.txt`. (Hint: use `operator!()`)

- Look at the contents of file `data1.txt`. You will see that it contains only integers, separated by spaces and/or newlines. Write a loop that reads in one integer at a time until you reach the end of the file.

  Print each integer as you read it in. Think about how you will know when you have reached the end of the file.

- Change the loop so that it introduces an *additional* condition to stop reading: when you have read an integer with value zero.

Exercise 5.2 – Word counting (optional)

The goal of this exercise is to count the number of lines, the number of words and the number of characters in a text file, similar to the functionality of the UNIX utility command `wc`.

Step 1 - Write a program that counts the *lines* of a text file

- Write a program that takes a filename as argument when it is invoked (e.g. ‘`myProgram theFile.txt`’), opens that file, and checks if the file was opened successfully. (Hint: use `main(int argc, char* argv[])`. You can use the file `ex5.2/example.txt` as input.

- Write a loop thats read the file line-by-line using `ifstream::getline(char* buf, int buflen) ;`.
  Think about how you will know in the loop when you are at the end of the file.

- In the loop, count the number of lines you read and report the line count to standard output at the end of the program.

Run your program on `example.txt` and cross check your result with the unix utility `wc -l <file>` (only if you run on a Unix or MacOS platform)

Step 2 - Augment the program to also count the number of *characters*.

- Add code in the reading loop that counts the number of characters in the line being processed, and use that number to calculate the total number of characters in the file.

- Cross check your result with '`wc -c example.txt`' (only if you run on a Unix or MacOS platform)

Step 3 – Modify the program such that also counts the number of *words* in the file

- Add code in the loop that reads and processes each line as follows:

  Construct an `istringstream` object (declared in `<sstream>`) passing the `char[]` line buffer as constructor argument.

  This creates an *input stream* representation of the line you are processing and allows you re-process the same line using the C++ streamer operators (>>).

- Read a `char[]` string from the `istringstream`. The read will stop as soon as it encounters a white in the character stream.

  Repeat the reading procedure until you are at the end of the stream
  (how do you know that you are at the end of a stream?).

  The number of successful reads is the number of words on the line.

- Use the number of words per line to calculate the total number of words in the file. Report the total number of words at the end of your program and cross check your result with '`wc example.txt`' (only if you run on a Unix or MacOS platform)