CSCI-511 Object-Oriented Programming

String substitution

Summary

For this project we'll focus on combining algorithm development within an object-oriented framework. The task is string substitution within a file while minimizing memory usage.

Problem description

The completed application will be invoked as follows:

```
bash$ stringsub oldstring newstring filename
```

stringsub The name of the executable

oldstring The string to search for within the file

newstring The string which should replace oldstring wherever oldstring is found within the file

filename The file which is searched/replaced.

Application requirements:

- The string substitution must happen inline. That is, no temporary files. Each byte is read once from the file and (if required) written back to the file.
- The algorithm must work in a single pass, reading from the beginning of the file to the end of the file.
- The file can only be opened one time. Both reading and writing must be done on the open file using the same fstream object.
- The algorithm must minimize the use of memory. That is, you cannot simply read the file into memory, manipulate it, and write it all out again. By default, a single byte buffer is allowed (if desired). A larger buffer is only allowed if newstring is longer than oldstring. The buffer is only allowed to grow when writing out a larger newstring requires it.
- The only time where a buffer is needed is when newstring is larger than oldstring. Specifically: (length(newstring)-length(oldstring))*numberOfSubstitutions. However, this is worst-case performance. A space-optimized algorithm can (in files requiring many substitutions) utilize a smaller buffer than the worst case. For example, when replacing **a** with **XYZ** in this file: **aaaaa**, in the worst case your buffer is allowed to as large as 10 characters. However, a better solution will (in this example) only require a buffer of size 3.
- Like native stream classes, on a "bad" read a single character may be pushed back onto the buffer. That is, at the point oldstring is partially-matched with the with file (or buffer) contents and fails, the failed character may be pushed back onto the buffer.
- There is a suite of test cases provided to help you develop your algorithm. One of the more difficult tests to get correct is partial embedded matches. For example, finding ***ab*** within **a*ab*** or finding ***abcd*** in **abc*abcd***.