CS 202 Homework 3

Bryan Beus

February 17, 2020

Source Code Link: `https://github.com/siddhartha-crypto/cs202/tree/master/hw3`

1 Design

1.1 Main

The program does not appear difficult to create.

There is one point of clarity that I make here.

"Time the file with LineToTokens and with ReadLine/PrintTokens. Have your program determine which choice to use by checking the command line."

This seems to imply that we need to write two separate types of methods to read in the tokens and columns – one for the line-to-tokens method, and one for read-line, then send this to tokens.

Yet, we also have the following.

"Test your program with a large Project Gutenberg book (such as Moby Dick or Wuthering Heights) and determine how long it takes to process it. What is the difference in time if you choose to skip the output"

This implies that we set one timer to read in the lines only, and not print them out, except for a report of the time involved.

This makes more sense to me, and so I am going to assume that the "–line-only" command in the instructions implies that only the lines should be read and timed.

1.2 Pretty Print

These projects are enjoyable, and I like learning how to process text.

For this specific project, I intend to have the optional `<h1> Chapter</h1>` aspect only apply to the "Stories of the Wagner Opera" file.

2 Post Mortem

2.1 Main

I really enjoyed both of these projects.

The only real struggle in the main project was the aspect of getting my iterators to properly iterate over and assemble each paragraph.

2.2 Pretty Print

There is a great satisfaction I feel over seeing the words manipulated automatically on the screen.

Would that I could go back to my twenties and learn this art form earlier! So much time wasted on repetitive tasks...

Having the initial aspect of the layout complete, from Main, made this relatively easy.

However, I did have to spend a long time trying to figure out why my `ofstream` was not outputting properly. I spent probably two hours reading through documentation on this one problem alone. Then I discovered that `ofstream` needs an `endl` to close, apparently. We probably discussed this in class, but I forgot.

3 Commit History

3.1 Main

```
2020-02-15 & \text{initiate hw3} \\
2020-02-15 & \text{clean hw3 dirs} \\
2020-02-15 & \text{add existing content into hw3 main} \\
2020-02-15 & \text{initial layout complete, begin compiling debug process in main in hw3} \\
2020-02-16 & \text{complete compilation debug process in main in hw3} \\
2020-02-16 & \text{debug line tokenizer function in main in hw3} \\
2020-02-16 & \text{debug linecols first value in main in hw3} \\
2020-02-16 & \text{debug blank line in LineToTokens() in main in hw3} \\
2020-02-16 & \text{debug TokenizerFunctions.cpp} \\
2020-02-16 & \text{calculate MB/sec size in main in hw3} \\
2020-02-16 & \text{debug console output in main in hw3} \\
2020-02-16 & \text{debug whitespace} \\
2020-02-16 & \text{debug deletion of report.txt in main in hw3} \\
2020-02-16 & \text{comment main() in main in hw3} \\
2020-02-16 & \text{restructure TokenizerFunctions.cpp in hw3} \\
2020-02-16 & \text{comment TokenizerFunctions in hw3} \\
```

3.2 Pretty Print

```
2020-02-16 & \text{initiate pretty print} \\
2020-02-16 & \text{debug os stream in PrettyPrint() in pretty print in hw3} \\
2020-02-16 & \text{debug ofstream in pretty print in hw3} \\
2020-02-16 & \text{debug ofstream in while loop in pretty print in hw3} \\
2020-02-16 & \text{complete logic for pretty print in hw3} \\
2020-02-16 & \text{complete pretty print in hw3} \\
2020-02-16 & \text{add word wrapping to pretty print in hw3} \\
2020-02-16 & \text{finish pretty print project} \\
```

4 Answers to Questions

  - `ofstream fout("data.dat", std::ios::out | std::ios::binary );`
  - (error checking)
  - `fout.write((char*)mydata, sizeof(myDataSize))`

- Writing a text file allows the file to be read simply by common readers. Binary files, on the other hand, may require special processing. The advantage to binary is speed of processing.
- `128 * 4 * 10`, `128 * 4 * n`
- The operating system has embedded firmware that loads first, and contains all the information necessary to read the hard drive. The hard drive is divided into partitions, then sectors, and so on down to the individual magnet/electric bits. Each available bit of memory has an address, and these store ones and zeros. The operating system access these when storing memory.
- We use ”Magic” numbers at work. Each instance of a specific software blockchain generates its own magic number based on the specific parameters that make up that individual blockchain. This allows instances of the blockchain on the network to identify each other before attempting to process signals on the network; this prevents confusion when the human-readable names of different blockchains are similar.
- The core philosophies of the two products, Microsoft and Linux, are open-source versus closed-source, and these essentially inform all aspects of their products, including coding style. I do not feel that either one is superior to the other; they are different tools for different purposes. The styles in Microsoft derive from large teams of paid individuals, whereas the Linux style derives often from volunteers, with

a tight network of individuals typically funded by a firm that sells close-source software. They are Yin (Linux) and Yang (Microsoft).

5 Sample Output

5.1 Main - Console Output

```
           File size is: 0.226231 Mbs.
   Processing only time: 0.0505597
                Speed is: 4.47452 Mbs/sec.

With report output time: 0.116311
        Time difference: 0.0657515
                Speed is: 1.94505 Mbs/sec.
          Speed diff is: 2.52948 Mbs/sec.
```

5.2 Main - Report Output

```
Line     1, Column       1: "The"
Line     2, Column       4: "Project"
Line     3, Column      11: "Gutenberg"
Line     4, Column      20: "EBook"
Line     5, Column      25: "of"
Line     6, Column      27: "Stories"
Line     7, Column      34: "of"
Line     8, Column      36: "the"
Line     9, Column      39: "Wagner"
Line    10, Column      45: "Opera,"
Line    11, Column      51: "by"
Line    12, Column      53: "H."
Line    13, Column      55: "A."
Line    14, Column      57: "Guerber"
Line    14, Column      57: blank line
Line    14, Column      57: "This"
Line    15, Column      61: "eBook"
Line    16, Column      66: "is"
Line    17, Column      68: "for"
Line    18, Column      71: "the"
Line    19, Column      74: "use"
Line    20, Column      77: "of"
```

```
Line    21, Column      79: "anyone"
Line    22, Column      85: "anywhere"
Line    23, Column      93: "at"
```

5.3 Pretty Print

```
The Project Gutenberg EBook of Stories of the
Wagner Opera, by H. A. Guerber

This eBook is for the use of anyone anywhere at
no cost and with almost no restrictions
whatsoever. You may copy it, give it away or
re-use it under the terms of the Project
Gutenberg License included with this eBook or
online at www.gutenberg.net

Title: Stories of the Wagner Opera

Author: H. A. Guerber

Release Date: October 9, 2005 [EBook #16840]

Language: English

*** START OF THIS PROJECT GUTENBERG EBOOK STORIES
OF THE WAGNER OPERA ***

Produced by Juliet Sutherland, Daniel Emerson
Griffith and the Online Distributed Proofreading
Team at http://www.pgdp.net

[Illustration: RICHARD WAGNER.]

STORIES OF THE WAGNER OPERA.

H.A. GUERBER,

Author of

"MYTHS OF GREECE AND ROME," "MYTHS OF NORTHERN
LANDS," "CONTES ET LÉGENDS," etc.

NEW YORK: DODD, MEAD, AND COMPANY. 1905.

_Copyright 1895_, BY DODD, MEAD AND COMPANY.

University Press: JOHN WILSON AND SON, CAMBRIDGE,
U.S.A.

Dedicated to my Friend, M.A. McC.

```

```
These short sketches, which can be read in a few
moments' time, are intended to give the reader as
clear as possible an outline of the great
dramatist-composer's work.

The author is deeply indebted to Professor G.T.
Dippold, to Messrs. Forman, Jackman, and Corder,
and to the Oliver Ditson Company, for the
poetical quotations scattered throughout the
text.

CONTENTS. Page

Rienzi, the Last of the Tribunes 7 The Flying
Dutchman 23 Tannhäuser 38 Lohengrin 56 Tristan
and Ysolde 72 The Master-Singers of Nuremberg 88
The Nibelung's Ring.--Rheingold 105 The Walkyrie
120 Siegfried 138 Dusk of the Gods 154 Parsifal
172

ILLUSTRATIONS. Page

...
...
```

6 My Programs

6.1 Main

```
/*
 * main.cpp
 * CS 202
 * February 14, 2020
 * Bryan Beus
 * Main file for main project
 */

#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
#include <fstream>
#include <utility>
#include <bits/stdc++.h>
#include <stdio.h>

#include "Miscellaneous.hpp"
#include "Value.hpp"
```

```
#include "StopWatch.hpp"
#include "TokenizerFunctions.hpp"

using std::cout;
using std::cin;
using std::endl;
using std::vector;
using std::string;
using std::pair;
using std::ifstream;
using std::ofstream;
using std::make_pair;

// Check whether the user has indicated to only do the processing and not the report
bool getLineInput(int &argv, char** argc) {

    bool res = false;

    if (argv < 3) {
        return res;
    }

    string isLineOnly(argc[2]);

    cout << endl;

    if (isLineOnly == "--line-only") {
        res = true;
    } else {
        cout << "The value provided {" << isLineOnly << "} is not a valid input." << endl;
        exit(0);
    }

    return res;

}

int main(int argv, char** argc) {

    // Check user has input the correct number of arguments
    if (argv < 2 || argv > 3) {
        cout << "Input error" << endl;
        exit(0);
    }

    // Discover whether user desires to skip the report creation
    bool isLineOnly = getLineInput(argv, argc);

    // Declare file to read
    string fileToRead(argc[1]);
    ifstream fin(fileToRead);

```

```
72      // Check that the file is valid
73      if (!fin) {
74          cout << "Error reading input file." << endl;
75          exit(0);
76      }
77
78      // Create a report file name
79      // Be sure to not overwrite an existing report if the user
        ↪  indicated --line-only
80      string outputname;
81      if (!isLineOnly) {
82          outputname = "report.txt";
83      } else {
84          outputname = "temporary_report_file.txt";
85      }
86
87      // Create an output stream for the report
88      ofstream fout(outputname);
89
90      if (!fout) {
91          cout << "Error creating report." << endl;
92          exit(0);
93      }
94
95      // Initiate variables
96      vector<string> tokens;
97      vector<pair<int, int>> linecols;
98      linecols.push_back(make_pair(1, 1));
99
100     // There are two tests:
101     // 1 for processing
102     // 1 for both processing and outputting
103     // (The second test must start, but if the user has indicated
104     // --line-only, it will be ignored later)
105     StopWatch processing_only;
106     StopWatch and_outputting;
107
108     // Read the lines from the file
109     bool linesRead = ReadLine(fin, tokens, linecols);
110
111     // Stop the first StopWatch
112     processing_only.captureFinishTime();
113
114     if (!linesRead) {
115         cout << "File is not readable." << endl;
116         exit(0);
117     }
118
119     // If the user has not indicated --line-only, create the
        ↪  report
120     if (!isLineOnly) {
121         PrintTokens(fout, tokens, linecols);
122     }
```

```
123
124     // Stop the second StopWatch
125     and_outputting.captureFinishTime();
126
127     fin.close();
128
129     // Discover input file size
130     ifstream f_size(fileToRead, std::ifstream::ate |
        ↪  std::ifstream::binary);
131     int byte_size = (int)(f_size.tellg());
132     float MB_size = (float)(byte_size) / (float)1048576;
133
134     // Discover speed of processing for first StopWatch
135     float MB_sec_process_only = (float)MB_size /
        ↪  (float)(processing_only.reportFinishTime());
136
137
138     // Print information to console
139     cout << setw(25) << "File size is: " << MB_size << " Mbs." <<
        ↪  endl;
140     cout << setw(25) << "Processing only time: " <<
        ↪  processing_only.reportFinishTime() << endl;
141     cout << setw(25) << "Speed is: " << MB_sec_process_only << "
        ↪  Mbs/sec." << endl;
142
143     if (!isLineOnly) { float MB_sec_with_proc = (float)MB_size /
        ↪  (float)(and_outputting.reportFinishTime()); cout << endl;
144         cout << setw(25) << "With report output time: " <<
            ↪  and_outputting.reportFinishTime() << endl;
145         cout << setw(25) << "Time difference: " <<
            ↪  (and_outputting.reportFinishTime() -
            ↪  processing_only.reportFinishTime()) << endl;
146         cout << setw(25) << "Speed is: " << MB_sec_with_proc << "
            ↪  Mbs/sec." << endl;
147         cout << setw(25) << "Speed diff is: " <<
            ↪  (MB_sec_process_only - MB_sec_with_proc) << "
            ↪  Mbs/sec." << endl;
148     }
149
150     // Delete the temporary report file, if needed
151     if (isLineOnly) {
152         remove("temporary_report_file.txt");
153     }
154
155     return 0;
156 }
```

6.2 Pretty Print

```
/*
 * main.cpp
 * CS 202
 * February 14, 2020
 * Bryan Beus
 * Main file for Pretty Print project in hw3
 */

#include <iostream>
#include <iomanip>
#include <string>
#include <vector>
#include <fstream>
#include <utility>
#include <bits/stdc++.h>
#include <stdio.h>

#include "Miscellaneous.hpp"
#include "Value.hpp"
#include "StopWatch.hpp"
#include "TokenizerFunctions.hpp"

using std::cout;
using std::cin;
using std::endl;
using std::vector;
using std::string;
using std::pair;
using std::ifstream;
using std::ofstream;
using std::make_pair;
using std::isdigit;

// Check if the provided input is a digit
// (Content influenced by Stackoverlflow
bool isDigit(const string &tempStr) {

    string::const_iterator it = tempStr.begin();
    while (it != tempStr.end() && isdigit(*it)) ++it;
    return !tempStr.empty() && it == tempStr.end();
}

// Check whether the user has indicated to only do the processing
 ↪  and not the report
void getLineInput(int &argv, char** argc, bool &isHtml, int
 ↪  &wrapCount) {

    // Discover whether the second input param is an int
    string tempStr(argc[2]);
    bool isInt = isDigit(tempStr);
```

```
49      if (isInt) {
50          wrapCount = stoi(tempStr);
51      }
52
53      // Check if isHtml should be true
54      // Check to make sure any int value is appropriate
55      if (tempStr == "--html") {
56          isHtml = true;
57      } else if (wrapCount != 0 && wrapCount < 30) {
58          cout << "Wrap count should be at least 30 columns." <<
            ↪  endl;
59          exit(0);
60      } else if (wrapCount != 0) {
61          isHtml = false;
62      } else {
63          cout << "The value provided {" << tempStr << "} is not a
            ↪  valid input." << endl;
64          exit(0);
65      }
66
67  }
68
69  int main(int argv, char** argc) {
70
71      // Check user has input the correct number of arguments
72      if (argv != 3) {
73          cout << "Input error" << endl;
74          exit(0);
75      }
76
77      // Discover whether user desires to skip the report creation
78      int wrapCount = 0;
79      bool isHtml;
80      getLineInput(argv, argc, isHtml, wrapCount);
81
82      // Declare file to read
83      string fileToRead(argc[1]);
84      ifstream fin(fileToRead);
85
86      // Check that the file is valid
87      if (!fin) {
88          cout << "Error reading input file." << endl;
89          exit(0);
90      }
91
92      // Initiate variables
93      vector<string> tokens;
94      vector<pair<int, int>> linecols;
95      linecols.push_back(make_pair(1, 1));
96
97      // Read the lines from the file
98      bool linesRead = ReadLine(fin, tokens, linecols);
99
```

```
100     if (!linesRead) {
101         cout << "File is not readable." << endl;
102         exit(0);
103     }
104 
105     fin.close();
106 
107     // Create new ostream
108     string prettyFilename = "pretty-print.txt";
109     ofstream fout;
110     fout.open(prettyFilename);
111 
112     if (!(fout.is_open())) {
113         cout << "Error creating pretty output file" << endl;
114         exit(0);
115     }
116 
117     // Pretty print the files
118     PrettyPrint(fout, tokens, isHtml, wrapCount);
119 
120     fout.close();
121 
122     return 0;
123 }
```