# CS 202 Homework 003

Kelby Hubbard

February 16, 2020

- Repository Link: `https://github.com/krhubbard2/CS202`
- Git Commits: `https://github.com/krhubbard2/CS202/commits`
- This homework took approximately 12 hours to complete.

## 1 Design

The design I followed for this program was to sectionalize the functions. The overall purpose of this program was that so we could use these functions later on and build off of it. By following the guidelines and creating separate functions for different tasks I was able to complete the program step by step and hopefully be able to cleanly and easily implement the functions in the future.

## 2 Post Mortem

I think the hardest part of this program as understanding exactly what was needed and the overall goal of the program. I didn't quite understand what the directions were asking–or rather the specifics on exactly how it was supposed to be accomplished. I think overall the program was fairly simple to put together, just understanding what was being asked and where took quite a bit of thought.

## 3 Answers to Questions

In this section, you will write the answers to the questions in the homework assignment.

1. To write std::string to a binary file using an ofstream by using std::ofstream save("filename.dat", std::ios::binary); and then from there you could implement save.write(string, string.size()); The biggest downside for doing this method is reading from the file could be particularly hard if you don't have white spaces or some other way of searching the file.
2. A trade off for text vs binary file is that text files are human readable. This could be beneficial or the opposite. If it is human readable it is possible that the user could go in and change some things that they shouldn't be touching since they can read it. On the other hand you, as a developer could possibly look back at your work and understand it easier. Binary is typically much faster also.
3. For a binary file with 128 bytes the 10th record is located at
4. Your operating system uses your hard drive to store files by creating separate file systems and partitions. It allocates a certain amount of bytes for each section so when it is searching or saving it knows which bytes and section to look into and adjust accordingly.
5. A magic number is a constant value used to identify a file format. For example a NES ROM file has magic number of NES.

## 4 Simple Tokenizer

### 4.1 Sample Output

Listing 1: Tokenizer Sample Output

```
./Tokenizer --help
To tokenize a .txt file use the command
"--tokenize filenamehere.txt".
Ensure the file is in the same directory as the
   program.
To do line only run command
"--lineonly filenamehere.txt".
Otherwise run the program to tokenize input text.
*PROGRAM COMPLETE*


./Tokenizer --lineonly Dracula.txt
Finished job: Sun Feb 16 20:46:22 2020
Elapsed time: 0.0623403s
File size was 0.842247MB. Result is 13.5105 MB/second
*PROGRAM COMPLETE*


./tokenizer --tokenize Dracula.txt
...
Row 15973, Column 43: "about"
Row 15973, Column 24: "new"
Row 15973, Column 53: "eBooks."
Row 15974, Column 1: "Empty Line"
Finished job: Sun Feb 16 20:47:18 2020
Elapsed time: 0.697059s
File size was 0.842247MB. Result is 1.20829 MB/second
*PROGRAM COMPLETE*


Enter some text. To exit type "END".
 This is a    sample output
testing this sample
END
Row 1, Column 2: "This"
Row 1, Column 4: "is"
Row 1, Column 10: "a"
Row 1, Column 15: "sample"
Row 1, Column 22: "output"
Row 2, Column 1: "testing"
Row 2, Column 9: "this"
Row 2, Column 14: "sample"
Row 3, Column 1: "END"
*PROGRAM COMPLETE*
```

### 4.2 Program 1 Questions

To process Dracula with –lineonly it only took 0.0623403 seconds. The file size being 0.842247MB (842247 Bytes) this comes out to 13.5 MB/second. Where as to process it with tokenize and reading to console it took .697059 seconds–coming out to about 1.2 MB/second. Clearly it is much faster (over 10x) to process –lineonly.

## 4.3 Git Commit Messages

| Date | Message |
|---|---|
| 2020-02-07 | Created basis of SimpleTokenizer |
| 2020-02-07 | Implamented bool LineToTokens in main.cpp |
| 2020-02-16 | Implamented error for incorrect argv input |
| 2020-02-16 | Implamented void ReadLine in tokenizer.cpp |
| 2020-02-16 | Implamented PrintTokens in tokenizer.cpp |
| 2020-02-16 | Allows user input |
| 2020-02-16 | Finished user input. |
| 2020-02-16 | Added sample.txt and error checking for file opening |
| 2020-02-16 | Implamented tokenizing from file. Need to fix repeat bug. |
| 2020-02-16 | Added comments. Cleaned up code. Need to do final bug checks. |
| 2020-02-16 | Added stopwatch.cpp/.hpp for timing |
| 2020-02-16 | Timed reading/output and added stopwatch |
| 2020-02-16 | Implamented MB/s in –tokenize |
| 2020-02-16 | Implamented MB/s in –lineonly |
| 2020-02-16 | Fixed MB display issue |
| 2020-02-16 | Fixed megabyte formula. |
| 2020-02-16 | Fixed a duplication bug. |

## 4.4 Source Code

```cpp
// Kelby Hubbard
// CS202
// February 16, 2020
// Hw003 -- A Simple Tokenizer

#include "tokenizer.hpp"
#include "stopwatch.hpp"

int main(int argc, const char** argv)
{
  //Variables needed for both user input or file input.
  vector<string> tokens;
  vector<std::pair<int, int>> linecols;
  string line;
  int row = 0;


  //TIMER
  StopWatch timer;
```

```

//Help command
if (argc >= 2 && argv[1] == string("--help"))
{
  cout << "To tokenize a .txt file use the command"
       << " \"--tokenize filenamehere.txt\".\nEnsure the file is in the "
       << "same directory as the program.\nTo do line only run command"
       << " \"--lineonly filenamehere.txt\". \nOtherwise run the program to "
       << "tokenize input text." << endl;
}

//--lineonly doesn't print back to user
else if (argc >= 2 && argv[1] == string ("--lineonly"))
  {
    ifstream file(argv[2]);
    //Ensure file can be opened. If not reference help to user.
    if (!file)
    {
      cout << "Error. Can't locate or open file. For help type \"--help\"\n";
    }
    //If file opened--continue
    else
    {
      bool read = true;
      timer.starttimer();
      while (read)
      {
        getline(file, line);
        row++;
        ReadLine(line, tokens, linecols, row);
        if (!file)
        {
          //If end of file end loops.
          if (file.eof())
          {
            read = false;
          }
          else
          {
            read = true;
          }
        }
      }
      timer.stoptimer();
      timer.elapsed();

      ifstream size(argv[2], std::ios::binary | std::ios::ate);
      double fsize = size.tellg();

      double megabyte =  (fsize / 1024) / 1024;
      double mbpstime = megabyte / timer.mbps();
      cout << "File size was " << megabyte << "MB. Result is "
           << mbpstime << " MB/second" << endl;
    }
  }

//--tokenize command
else if (argc >= 2 && argv[1] == string ("--tokenize"))
{
  ifstream file(argv[2]);
  //Ensure file can be opened. If not reference help to user.
  if (!file)
```

```
    {
      cout << "Error. Can't locate or open file. For help type \"--help\"\n";
    }
    //If file opened--continue
    else
    {
      bool read = true;
      timer.starttimer();
      while (read)
      {
        getline(file, line);
        row++;
        ReadLine(line, tokens, linecols, row);
        if (!file)
        {
          //If end of file end loops.
          if (file.eof())
          {
            read = false;
          }
          else
          {
            read = true;
          }
        }
      }

      //Print tokens back to user.
      PrintTokens(tokens, linecols);

      timer.stoptimer();
      timer.elapsed();

      ifstream size(argv[2], std::ios::binary | std::ios::ate);
      double fsize = size.tellg();

      double megabyte =  (fsize / 1024) / 1024;
      double mbpstime = megabyte / timer.mbps();
      cout << "File size was " << megabyte << "MB. Result is "
           << mbpstime << " MB/second" << endl;
    }
  }


  //Incorrect argv input
  else if (argc >= 2)
    {
      cout << "Command not understood. For assistance please type"
           << "\"--help\"" << endl;
    }

  //If no commands were passed when running program
  else if (argc <= 2)
    {
    cout << "Enter some text. To exit type \"END\"." << endl;
    //User Input
    bool loop = true;
    while (loop)
    {
      getline(cin, line);
      if (line == "END" || line == "end" || line == "End")
      {
        loop = false;
      }
      row++;
      ReadLine(line, tokens, linecols, row);


    }

    PrintTokens(tokens, linecols);
  }



  return 0;
}
```

## 4.5 Tokenizer Header

```
// Kelby Hubbard
// CS202
// February 16, 2020
// Hw003 -- A Simple Tokenizer

#ifndef TOKENIZER_HPP_
#define TOKENIZER_HPP_

/////////////////////////////////////////////////////////////////////////
// U S I N G ////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////

#include <iostream>
using std::cin;
using std::cout;
using std::endl;
#include <string>
using std::string;
#include <vector>
using std::vector;
#include <sstream>
using std::istringstream;
#include <fstream>
using std::ifstream;




/////////////////////////////////////////////////////////////////////////
// P R O T O T Y P E S //////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////////////


void ReadLine(const std::string& line, std::vector<std::string>& tokens,
          std::vector<std::pair<int, int>>& linecols, const int& row);

void PrintTokens(const std::vector<std::string>& tokens,
          const std::vector<std::pair<int, int>>& linecols);








#endif
```

## 4.6 Tokenizer Source

```cpp
// Kelby Hubbard
// CS202
// February 16, 2020
// Hw003 -- A Simple Tokenizer

#include "tokenizer.hpp"

//Reads a getline, converts to tokens reading both row and column
void ReadLine(const std::string& line, std::vector<std::string>& tokens,
          std::vector<std::pair<int, int>>& linecols, const int& row)
{
  string str;
  istringstream iss(line);

  //If line is not empty
  if (!line.empty())
  {
    while (!iss.eof())
    {
      iss >> str;
      if (iss)
      {
        tokens.push_back(str);
        linecols.push_back(std::make_pair(row, line.find(str) + 1));
      }
    }
  }

  //If line is empty
  else
  {
    tokens.push_back("Empty Line");
    linecols.push_back(std::make_pair(row, 1));
  }
}

void PrintTokens(const std::vector<std::string>& tokens,
          const std::vector<std::pair<int, int>>& linecols)
{
  int r = 0;
 for (auto i : tokens)
 {
   cout << "Row " << linecols[r].first << ", Column " << linecols[r].second
        << ": \"" << i << "\"" << endl;
   r++;
 }
}
```

## 4.7 Stopwatch Header

```
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It II

#ifndef STOPWATCH_HPP_
#define STOPWATCH_HPP_


#include <chrono>
#include <ctime>
#include <iostream>
using std::cout;
using std::endl;
#include <random>

class StopWatch
{
public:

  std::chrono::system_clock::time_point _start;
  std::chrono::system_clock::time_point _end;

void starttimer();
void stoptimer();
void elapsed();
double mbps();
};






#endif
```

## 4.8 Stopwatch Source

```
// Kelby Hubbard
// CS202
// Jan. 26, 2020
// HW001 -- Time It II

#include "stopwatch.hpp"


void StopWatch::starttimer()
{
  _start = std::chrono::system_clock::now();
}

void StopWatch::stoptimer()
{
  _end = std::chrono::system_clock::now();
}

void StopWatch::elapsed()
{
  std::chrono::duration<double> elapsed_seconds = _end-_start;
  std::time_t end_time = std::chrono::system_clock::to_time_t(_end);

  std::cout << "Finished job: " << std::ctime(&end_time)
       << "Elapsed time: " << elapsed_seconds.count() << "s\n";
}

double StopWatch::mbps()
{
  std::chrono::duration<double> elapsed_seconds = _end-_start;
  std::time_t end_time = std::chrono::system_clock::to_time_t(_end);

  double passed = elapsed_seconds.count();

  return passed;
}
```

# 5 Pretty Book

## 5.1 Sample Output / Screenshot

Listing 2: Tokenizer Sample Output

```
./PrettyBook book.txt 40
I  felt I uttered my explanations
awkwardly; the master frowned. 'It is
nothing, is it, Ellen Dean?' he said
sternly. 'You shall account more
clearly for keeping me ignorant of
this!' And he took his wife in his
arms, and looked at her with anguish.


 At first she gave him no glance of
recognition: he was invisible to her
abstracted gaze. The delirium was not
fixed, however; having weaned her eyes
from contemplating the outer darkness,
by degrees she centred her attention on
him, and discovered who it was that
held her.
```

## 5.2 Git Commit Messages

| Date | Message |
|---|---|
| 2020-02-16 | Implemented lineToTokens in main.cpp |
| 2020-02-16 | Fixed lineToTokens |
| 2020-02-16 | Implamented reading and argv. Need to char wrap |
| 2020-02-16 | Finished char wrapping. Need to do error checking. |
| 2020-02-16 | Finished |

## 5.3 Source Code

```
// Kelby Hubbard
// CS202
// February 16, 2020
// Hw003 -- Book Pretty Printer

#include <iostream>
using std::cout;
using std::endl;
#include <fstream>
using std::ifstream;
#include <string>
using std::string;
#include <vector>
using std::vector;
#include <sstream>
using std::istringstream;
#include <algorithm>

bool lineToTokens(const std::string& line, std::vector<std::string>& tokens)
{
  string str;
  for (auto a : line)
  {
    if (a == ' ')
    {
      tokens.push_back(str);
      str = "";
    }
    else
    {
      str += a;
    }
  }
    if (str.length() != 0)
    {
      tokens.push_back(str);
    }
    return true;
  }

```

```
int main(int argc, char* argv[])
{
  vector<string> paragraphs;
  string line;
  int wrap;

  if (argc < 2)
  {
    cout << "Error. Please try format: \"filename.txt 40\" or "
         << "\"filename.txt --html\". Thank you. \n";
         return 0;
  }
  istringstream iss(argv[2]);
  iss >> wrap;
  if (!iss)
  {
    cout << "Error. Improper entry.";
  }
  else if (argc >= 2  && iss)
  {
      ifstream ifile(argv[1]);
      if (!ifile)
      {
        cout << "Error. Couldn't read file." << endl;
      }

      while (getline(ifile, line))
      {
        if (line == "")
        {
          paragraphs.push_back("\n\n");
        }
        lineToTokens(line, paragraphs);
      }

        string tempstr;
        for (auto r : paragraphs)
        {
            if (tempstr.length() + r.length() >= wrap)
            {
              cout << tempstr << endl;
              tempstr = r;
            }
            else
            {
              tempstr += r;
            }

            if (tempstr.size() != wrap)
            {
              tempstr += " ";
            }
        }
        cout << tempstr;


      cout << endl;

  }
  return 1;
}
```