**Critical Thinking Assignment Module Five Assignment**

**User Input Program**

Victor Enogwe

Computer Science, Colorado State University - Global Campus

CSC450-1: Programming III

Reginald Haseltine

May 20, 2024

In this module, we’ve looked at concurrency, its vulnerabilities and file IO mechanisms.

I created a program that reads the content of a file “CSC450\_CT5\_mod5.txt” into a string, asks the user to enter an input string and appends the user input to the file content. The program also takes the contents of the file “CSC450\_CT5\_mod5.txt” and reverses it. The reversed content is stored in another file “CSC450\_CT5\_mod5-reverse.txt” and its content is always overwritten.

**Program Pseudocode**:

PROGRAM: User Input Read/Write Program  
- This program takes a string input from a user and stores it in a file "CSC450\_CT5\_mod5.txt".  
 The program reverses the values of the contents of the file "CSC450\_CT5\_mod5.txt" and stores it in "CSC450\_CT5\_mod5-reverse.txt".  
  
BEGIN  
 BEGIN  
 declare a "read\_file" function that takes an absolute file path parameter and returns the content of the file as a string  
 declare a "write\_file" function that takes a value(the data to be written), an absolute file path and a file write mode as parameters.  
 The function will write the data to the file in the given mode(append or overwrite).  
 declare a "str\_reverse" function that takes a string parameter and returns the reversed string. Make sure the initial string is not mutated.  
 declare a "get\_input" function that prompts a user to enter a string and returns the string.  
 The function takes one argument: a string "instruction" to tell the user what to enter.  
 END  
  
 declare a "save\_input" function that  
 takes a user's input using the "get\_input" function  
 reads the contents of the "CSC450\_CT5\_mod5.txt" file using the "read\_file" function  
 appends the input to the file using the "write\_file" function  
 declare a "save\_reversed\_file\_content" function that  
 reads the contents of the "CSC450\_CT5\_mod5.txt" file using the "read\_file" function.  
 reverses the file contents.  
 saves the reversed string to the "CSC450\_CT5\_mod5-reverse.txt" file using the "write\_file" function.  
 the contents of the "CSC450\_CT5\_mod5-reverse.txt" file should be overwritten.  
 declare a main function to run the program.  
 inside the main function:  
 - call the "save\_input" function  
 - call the "save\_reversed\_file\_content" function  
END

**String.h:**

/\*  
 \* User Input Read/Write Program  
 \* String.h  
 \* string formatter  
 \* Created by Victor Enogwe on 20/05/2024.  
 \*/  
  
#ifndef STRING\_H  
#define STRING\_H  
  
#include <string>  
  
using std::string;  
  
string str\_trim(const string& value);  
string str\_reverse(const string& value);  
  
#endif //STRING\_H

**String.cpp:**

/\*  
 \* User Input Read/Write Program  
 \* String.cpp  
 \* string formatter  
 \* Created by Victor Enogwe on 20/05/2024.  
 \*/  
  
#include <string>  
#include <regex>  
  
using std::string;  
using std::reverse;  
using std::regex;  
using std::regex\_replace;  
  
/\*  
 \* str\_trim  
 \* trims the leading and trailing whitespaces in a string  
 \* avoids mutating original string  
 \*  
 \* @Param const string& {value} the original string  
 \*  
 \* @Return string the trimmed string  
 \*/  
string str\_trim(const string& value) {  
 string s;  
  
 s.assign(value);  
  
 return regex\_replace(  
 regex\_replace(s, regex( "^\\s+$" ), ""),  
 regex( "\\s+$" ),  
 ""  
 );  
}  
  
/\*  
 \* str\_reverse  
 \* reverses a string  
 \* avoids mutating original string  
 \*  
 \* @Param const string& {value} the original string  
 \*  
 \* @Return string the reversed string  
 \*/  
string str\_reverse(const string& value) {  
 string s;  
  
 s.assign(value);  
  
 reverse(s.begin(), s.end());  
  
 return s;  
}

**File.h:**

/\*  
 \* User Input Read/Write Program  
 \* File.h  
 \* reads and writes values to a file  
 \* Created by Victor Enogwe on 20/05/2024.  
 \*/  
  
#ifndef FILE\_H  
#define FILE\_H  
  
#include <string>  
#include <filesystem>  
  
using std::string;  
using std::filesystem::path;  
using std::filesystem::canonical;  
using std::filesystem::current\_path;  
using std::ios;  
  
const path DEFAULT\_DIRECTORY = current\_path();  
const path FILE\_NAME("CSC450\_CT5\_mod5.txt");  
const path REVERSE\_FILE\_NAME("CSC450-mod5-reverse.txt");  
const path FILE\_PATH = DEFAULT\_DIRECTORY / FILE\_NAME;  
const path REVERSE\_FILE\_PATH = DEFAULT\_DIRECTORY / REVERSE\_FILE\_NAME;  
const path FILE\_ABSOLUTE\_PATH(canonical(FILE\_PATH));  
const path REVERSE\_FILE\_ABSOLUTE\_PATH(canonical(REVERSE\_FILE\_PATH));  
  
string read\_file(const path& file\_path = FILE\_ABSOLUTE\_PATH);  
void write\_file(const string& value, const path& file\_path = FILE\_ABSOLUTE\_PATH, unsigned int mode = ios::*app*);  
  
#endif //FILE\_H

**File.cpp:**

/\*  
 \* User Input Read/Write Program  
 \* File.cpp  
 \* reads and writes values to a file  
 \* Created by Victor Enogwe on 20/05/2024.  
 \*/  
  
#include <iostream>  
#include <sstream>  
#include <fstream>  
#include <string>  
#include <filesystem>  
  
using std::cout;  
using std::runtime\_error;  
using std::string;  
using std::filesystem::path;  
using std::ifstream;  
using std::ofstream;  
using std::ostringstream;  
using std::ios;  
  
/\*  
 \* read\_file  
 \* reads a file from disk  
 \*  
 \* @Param const path& {file\_path} optional the absolute path to the file  
 \*  
 \* @Return string the file content  
 \*/  
string read\_file(const path& file\_path) {  
 try {  
 /\*  
 \* Reserve string space upfront  
 \* and read entire file.  
 \*  
 \* @Note An improvement may be to limit filesize to read  
 \* to get the file size the pointer on open can be set to the end of the file  
 \* "tellg" call can then be used to check - note that this maybe inacccurate.  
 \* https://stackoverflow.com/questions/22984956/tellg-function-give-wrong-size-of-file/22986486#22986486  
 \* suggests some remedies to using "tellg" for file size check.  
 \*/  
  
 cout << "Reading file(" << file\_path << ") from disk\n";  
  
 ifstream input\_file(file\_path);  
 ostringstream buffer;  
  
 buffer << input\_file.rdbuf();  
  
 const string value = buffer.str();  
  
 input\_file.close();  
  
 return value;  
 } catch (const runtime\_error& error) {  
 cout << "Error reading file(" << file\_path << ") from disk\n";  
  
 throw error;  
 }  
}  
  
/\*  
 \* write\_file  
 \* appends string content to a file on disk  
 \*  
 \* @Param const string& {value} the data to write  
 \* @Param const path& {file\_path} optional the absolute path to the file  
 \*  
 \* @Returns void  
 \*/  
void write\_file(const string& value, const path& file\_path, const unsigned int mode) {  
 try {  
 cout << "Writing value to file(" << file\_path << ") on disk\n";  
  
 // open file - create if not exists  
 ofstream output\_file;  
  
 output\_file.open(file\_path, mode);  
  
 output\_file << value;  
  
 output\_file.close();  
 } catch (const runtime\_error& error) {  
 cout << "Error writing to file(" << file\_path << ") on disk\n";  
  
 throw error;  
 }  
}

**Input.h:**

/\*  
 \* User Input Read/Write Program  
 \* Input.h  
 \* reads and formats user input  
 \* Created by Victor Enogwe on 20/05/2024.  
 \*/  
  
#ifndef INPUT\_H  
#define INPUT\_H  
  
#include <string>  
  
using std::string;  
  
string get\_input(const string& instruction);  
  
#endif //INPUT\_H

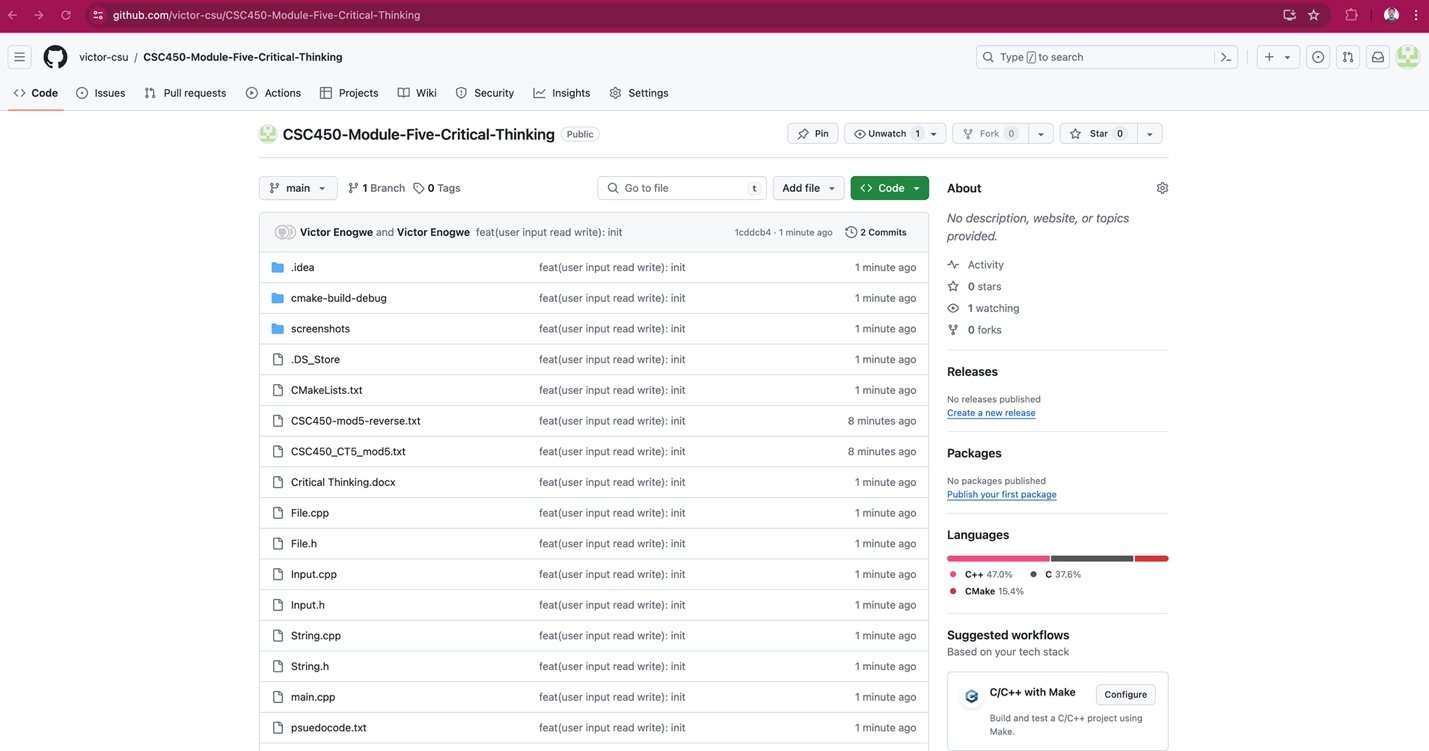
**Input.cpp:**

/\*  
 \* User Input Read/Write Program  
 \* Input.cpp  
 \* reads and formats user input  
 \* Created by Victor Enogwe on 20/05/2024.  
 \*/  
  
#include <iostream>  
#include <string>  
#include <regex>  
#include "String.h"  
  
using namespace std::regex\_constants;  
  
using std::cout;  
using std::cin;  
using std::getline;  
using std::string;  
using std::regex;  
using std::regex\_replace;  
using std::regex\_match;  
using std::runtime\_error;  
using std::stoi;  
  
/\*  
 \* get\_input  
 \* asks a user to enter an integer  
 \*  
 \* @Param const string& {instruction} user instruction  
 \*  
 \* @Return string the input string  
 \*/  
string get\_input(const string& instruction) {  
 try {  
 cout << instruction;  
  
 string input;  
  
 getline(cin, input);  
  
 return input;  
 } catch (const runtime\_error& error) {  
 cout << error.what();  
  
 throw error;  
 }  
}

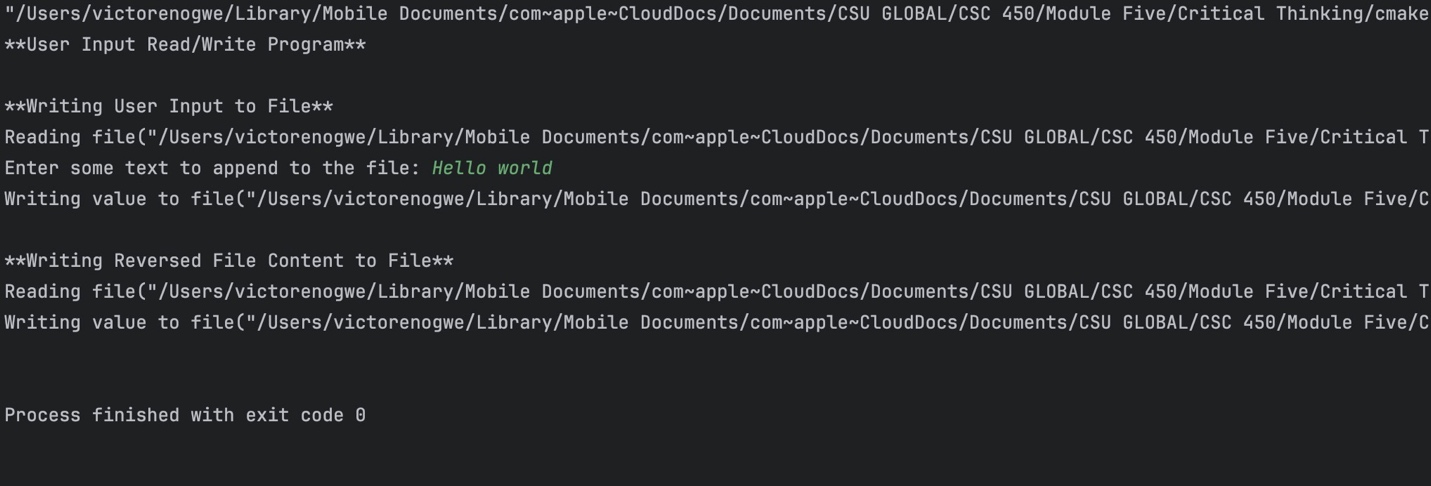
**Main.cpp:**

#include <iostream>  
/\*  
 \* Program: User Input Read/Write Program  
 \* a program that will take a string input from a user and store it in a file "CSC450\_CT5\_mod5.txt".  
 \* The program reverses the values of the contents of the file "CSC450\_CT5\_mod5.txt" and stores it in "CSC450\_CT5\_mod5-reverse.txt".  
 \*/  
  
#include <fstream>  
#include <string>  
#include <filesystem>  
  
#include "File.h"  
#include "Input.h"  
#include "String.h"  
  
using std::cout;  
using std::runtime\_error;  
using std::string;  
using std::filesystem::path;  
using std::filesystem::canonical;  
using std::filesystem::current\_path;  
using std::ifstream;  
using std::ofstream;  
using std::ios;  
using std::istreambuf\_iterator;  
  
void save\_input() {  
 cout << "\*\*Writing User Input to File\*\*\n";  
  
 const string file\_contents = read\_file();  
 const string input = get\_input("Enter some text to append to the file: ");  
  
 write\_file(input);  
  
 cout << "\n";  
}  
  
void save\_reversed\_file\_content() {  
 cout << "\*\*Writing Reversed File Content to File\*\*\n";  
  
 const string file\_contents = read\_file();  
 const string reversed\_file\_content = str\_reverse(file\_contents);  
  
 write\_file(reversed\_file\_content, REVERSE\_FILE\_ABSOLUTE\_PATH, ios::*trunc*);  
  
 cout << "\n";  
}  
  
int main() {  
 cout << "\*\*User Input Read/Write Program\*\*\n\n";  
  
 save\_input();  
 save\_reversed\_file\_content();  
  
 return 0;  
}

**Git Repository Image: Git Branch = Main** **-** **<https://github.com/victor-csu/CSC450-Module-Five-Critical-Thinking>**

****

**Happy Path Execution Screenshot – Fictional Person - CSC450\_CT5\_mod5-execution-output:**



References

*Tellg() function give wrong size of file?* (n.d.). Stack Overflow. <https://stackoverflow.com/questions/22984956/tellg-function-give-wrong-size-of-file/22986486#22986486>