

Documentation: Critical Thinking 5

Alejandro Ricciardi

Colorado State University Global

CSC450: Programming III

Professor: Reginald Haseltine

November 10, 2024

Documentation: Critical Thinking 5

This documentation is part of the Critical Thinking 5 Assignment from CSC450: Programming III at Colorado State University Global. The program's name is User Input to File in Reverse. It provides an overview of the program's functionality and testing scenarios, including console output screenshots. The program is coded in C++ 23.

The Assignment Direction:

Demonstrate an understanding of C++ programming concepts by completing the following:

1. Program: Create a C++ program that will obtain input from a user and store it into the provided [CSC450_CT5_mod5.txt](#) file.
2. Download [CSC450_CT5_mod5.txt](#) file. Your program should append it to the provided text file, without deleting the existing data:
 - a. Store the provided data in the CSC450_CT5_mod5.txt file.
 - b. Create a reversal method that will reverse all of the characters in the CSC450_CT5_mod5.txt file and store the result in a CSC450-mod5-reverse.txt file.

Compile and submit your pseudocode, source code, and screenshots of the application executing the application, the results and your GIT repository in a single document.

⚠ My notes:

- The simple C++ console application is in file **CTA-5-Input-to-file.cpp**
- The program follows the following SEI CERT C/C++ Coding Standard:
 - STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
 - STR52-CPP. Use valid references, pointers, and iterators to reference elements of a `basic_string`
 - STR53-CPP. Range check element access
 - FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
 - FIO51-CPP. Close files when they are no longer needed
 - ERR50-CPP. Do not abruptly terminate the program
 - ERR51-CPP. Handle all exceptions
 - ERR56-CPP. Guarantee exception safety

Program Description:

This program prompts the user to enter a string (sentence) and appends it to the "CSC450_CT5_mod5.txt" file without deleting existing data.

The program also validates the user input, trimming leading and trailing whitespaces from input text. It then reverses each line of "CSC450_CT5_mod5.txt" by reversing the characters in each line while maintaining the order of the lines, and stores the reversed content in "CSC450-mod5-reverse.txt"

Git Repository

I use [GitHub](#) as my Distributed Version Control System (DVCS), the following is a link to my GitHub, [Omegapy](#).

My GitHub repository that is used to store this assignment is named [My-Academics-Portfolio](#).

The link to this specific assignment is:

<https://github.com/Omegapy/My-Academics-Portfolio/tree/main/Programming-3-CSC450/Critical-Thinking-5>

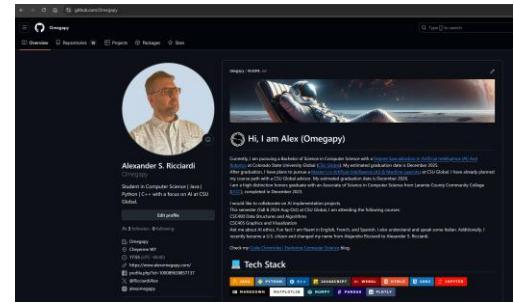
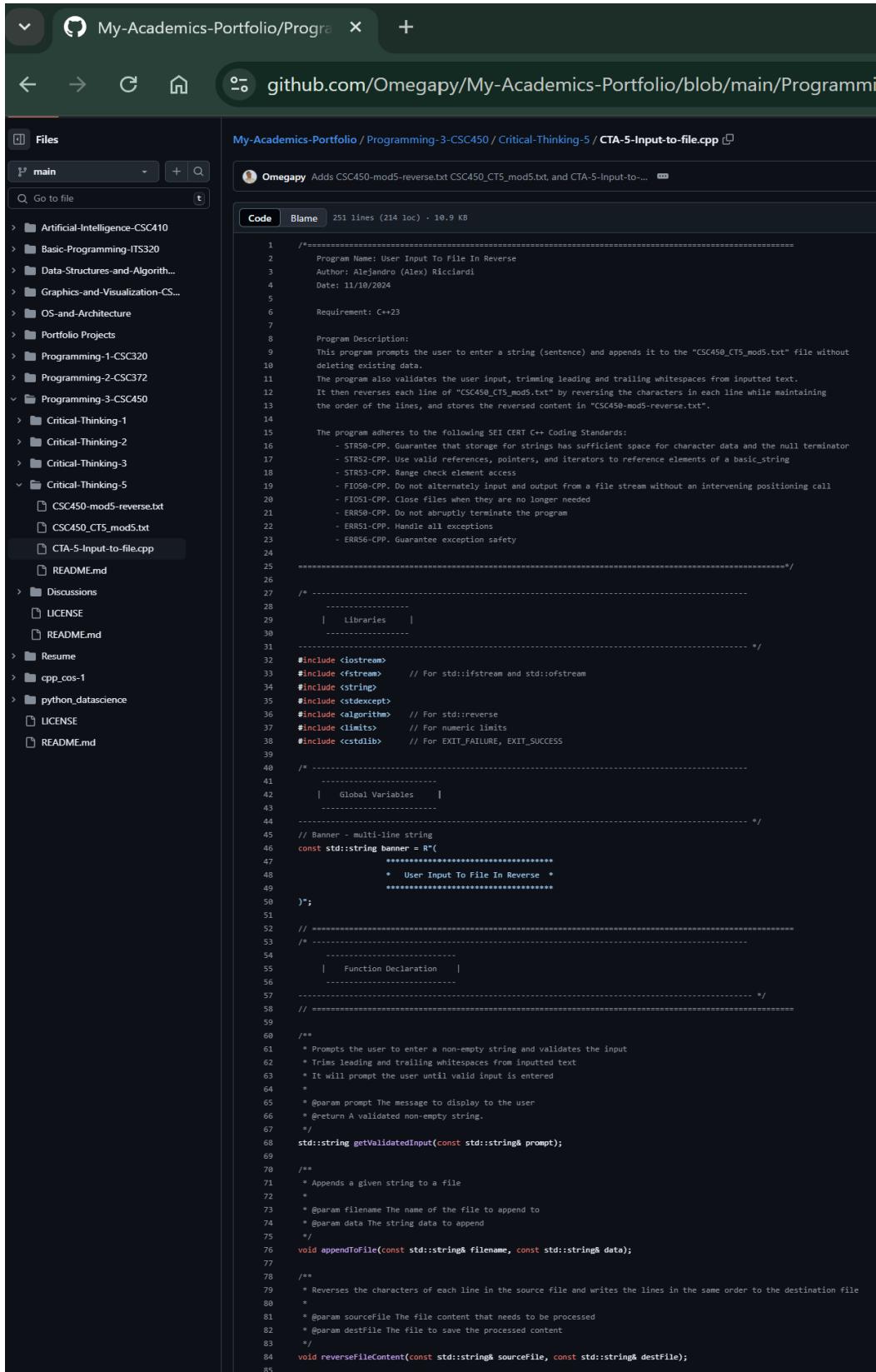


Image of the source code in the GitHub: see next page

Figure 1
Source Code in GitHub



The screenshot shows a GitHub repository interface. The repository name is "My-Academics-Portfolio/Programm". The current file being viewed is "CTA-5-Input-to-file.cpp". The code is a C++ program that reads a string from the user, appends it to a file, and then reverses the file's content. It includes comments explaining its functionality and adherence to SEI CERT C++ Coding Standards.

```

1  //=====================================================================
2  // Program Name: User Input To File In Reverse
3  // Author: Alejandro (Alex) Ricciardi
4  // Date: 11/10/2024
5
6  Requirement: C++23
7
8  Program Description:
9  This program prompts the user to enter a string (sentence) and appends it to the "CSC450_CTS5_mod5.txt" file without
10 deleting existing data.
11 The program also validates the user input, trimming leading and trailing whitespaces from inputted text.
12 It then reverses each line of "CSC450_CTS5_mod5.txt" by reversing the characters in each line while maintaining
13 the order of the lines, and stores the reversed content in "CSC450-mod5-reverse.txt".
14
15 The program adheres to the following SEI CERT C++ Coding Standards:
16 - STRS0-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
17 - STRS2-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
18 - STRS3-CPP. Range check element access
19 - FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
20 - FIO51-CPP. Close files when they are no longer needed
21 - ERRS0-CPP. Do not abruptly terminate the program
22 - ERRS1-CPP. Handle all exceptions
23 - ERRS6-CPP. Guarantee exception safety
24
25 =====
26
27 /**
28  -----
29  |   Libraries   |
30  -----
31 -----
32 #include <iostream>
33 #include <fstream>    // For std::ifstream and std::ofstream
34 #include <string>
35 #include <stdexcept>
36 #include <algorithm> // For std::reverse
37 #include <limits>    // For numeric limits
38 #include <cstdlib>   // For EXIT_FAILURE, EXIT_SUCCESS
39
40 /**
41  -----
42  |   Global Variables   |
43  -----
44 -----
45 // Banner - multi-line string
46 const std::string banner = R"(
47     ****
48     * User Input To File In Reverse *
49     ****
50 )";
51
52 /**
53  -----
54  |   Function Declaration   |
55  -----
56 -----
57 // ****
58 // ****
59 /**
60 /**
61 * Prompts the user to enter a non-empty string and validates the input
62 * Trims leading and trailing whitespaces from inputted text
63 * It will prompt the user until valid input is entered
64 *
65 * @param prompt The message to display to the user
66 * @return A validated non-empty string.
67 */
68 std::string getValidatedInput(const std::string& prompt);
69
70 /**
71 * Appends a given string to a file
72 *
73 * @param filename The name of the file to append to
74 * @param data The string data to append
75 */
76 void appendToFile(const std::string& filename, const std::string& data);
77
78 /**
79 * Reverses the characters of each line in the source file and writes the lines in the same order to the destination file
80 *
81 * @param sourceFile The file content that needs to be processed
82 * @param destFile The file to save the processed content
83 */
84 void reverseFileContent(const std::string& sourceFile, const std::string& destFile);
85

```

Files

main

Go to file

- Artificial-Intelligence-CSC410
- Basic-Programming-ITS320
- Data-Structures-and-Algorith...
- Graphics-and-Visualization-CS...
- OS-and-Architecture
- Portfolio Projects
- Programming-1-CSC320
- Programming-2-CSC372
- Programming-3-CSC450
 - Critical-Thinking-1
 - Critical-Thinking-2
 - Critical-Thinking-3
 - Critical-Thinking-5
 - CSC450-mod5-reverse.txt
 - CSC450_CTS5_mod5.txt
 - CTA-5-Input-to-file.cpp
 - README.md
 - Discussions
 - LICENSE
 - README.md
- Resume
- cpp_cos-1
- python_datascience
- LICENSE
- README.md

Code Blame 251 lines (214 loc) · 10.9 KB

```

86 // =====
87 /* -----
88 | Main Function |
89 -----
90
91     The main function runs the program.
92     Asks the user to enter data to be added to file.
93     Catch any possible exceptions.
94
95     Handles Rules:
96     - ERR50-CPP. Do not abruptly terminate the program
97     - ERR51-CPP. Handle all exceptions
98
99 -----
100 // -----
101 int main() {
102     std::cout << banner << std::endl;
103
104     std::cout << "Welcome to the User Input to File Program!\n\n";
105
106     // Try-catch block to handle all exceptions (ERR51-CPP)
107     try {
108         std::string userInput = getValidatedInput("Enter data to append to CSC450_CTS5_mod5.txt: ");
109
110         appendToFile("CSC450_CTS5_mod5.txt", userInput + "\n"); // Append with newline
111
112         reverseFileContent("CSC450_CTS5_mod5.txt", "CSC450-mod5-reverse.txt");
113
114         std::cout << "\nData has been successfully appended and reversed.\n";
115     }
116     catch (const std::exception& e) {
117         // Catch any standard exceptions thrown in the try block (ERR51-CPP)
118         std::cerr << "\nERROR: Exception caught: " << e.what() << std::endl;
119         return EXIT_FAILURE; // Exit the program with an error code (ERR50-CPP)
120     }
121     catch (...) {
122         // Catch any non-standard exceptions (ERR51-CPP)
123         std::cerr << "\nERROR: Unknown exception caught. Exiting program.\n";
124         return EXIT_FAILURE; // Exit the program with an error code (ERR50-CPP)
125     }
126
127     return EXIT_SUCCESS; // Successful program termination
128 }
129
130 // -----
131 // -----
132 | Function Definitions |
133 -----
134 -----
135 -----
136 // -----
137 // -----
138 /**
139 * Prompts the user to enter a non-empty string and validates the input
140 * Trims leading and trailing whitespaces from inputted text
141 * It will prompt the user until valid input is entered
142 *
143 * @param prompt The message to display to the user
144 * @return A validated non-empty string.
145 */
146 std::string getValidatedInput(const std::string& prompt) {
147     while (true) {
148         std::string input;
149
150         std::cout << prompt;
151
152         // Get the entire line of input and store it in a string object
153         std::getline(std::cin, input); // Using std::string ensures sufficient storage (STR50-CPP)
154
155         // Trim leading and trailing whitespaces
156         size_t start = input.find_first_not_of("\t\n\r"); // Valid index operation (STR53-CPP)
157         size_t end = input.find_last_not_of("\t\n\r"); // Valid index operation (STR53-CPP)
158
159         if (start == std::string::npos) {
160             std::cerr << "Invalid input: Input cannot be empty or just whitespace. Please try again.\n";
161             continue; // Prompt the user again
162         }
163
164         // Trim the input
165         input = input.substr(start, end - start + 1); // Range-checked substring (STR53-CPP)
166
167         return input; // Validated input
168     }
169 }
170

```

```

    > Graphics-and-Visualization-CSCI372
    > OS-and-Architecture
    > Portfolio Projects
    > Programming-1-CSC320
    > Programming-2-CSC372
    > Programming-3-CSC450
    > Critical-Thinking-1
    > Critical-Thinking-2
    > Critical-Thinking-3
    > Critical-Thinking-5
        CSC450-mod5-reverse.txt
        CSC450_CTS_mod5.txt
        CTA-5-Input-to-file.cpp
        README.md
    > Discussions
        LICENSE
        README.md
    > Resume
    > cpp_cos-1
    > python_datascience
        LICENSE
        README.md

172 // -----
173 /**
174 * Appends a given string to a file
175 *
176 * Handles Rules:
177 * - FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
178 * - FIO51-CPP. Close files when they are no longer needed
179 * - ERR51-CPP. Handle all exceptions
180 * - ERR56-CPP. Guarantee exception safety
181 *
182 *
183 * @param filename The name of the file to append to
184 * @param data The string data to append
185 */
186 void appendToFile(const std::string& filename, const std::string& data) {
187     std::ofstream outFile; // write data streams to files
188
189     try {
190         // RAI: Automatically opens the file - (ERR56-CPP)
191         outFile.open(filename, std::ios::app); // Open file in append mode
192         // No alternating input/output without repositioning (FIO50-CPP)
193         outFile << data; // Write data to file
194         outFile.close(); // Close file when done (FIO51-CPP)
195     }
196     catch (const std::ofstream::failure& e) {
197         // Handle file operation exceptions (ERR51-CPP)
198         throw std::runtime_error("Failed to open or write to file '" + filename + ".\"");
199     }
200 }
201
202 // -----
203 /**
204 * Reverses the characters of each line in the source file and writes the lines in the same order to the destination file
205 *
206 * Handles Rules:
207 * - STR52-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
208 * - FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
209 * - FIO51-CPP. Close files when they are no longer needed
210 * - ERR51-CPP. Handle all exceptions
211 * - ERR56-CPP. Guarantee exception safety
212 *
213 *
214 * @param sourceFile The file content that needs to be processed
215 * @param destFile The file to save the processed content
216 */
217 void reverseFileContent(const std::string& sourcefile, const std::string& destFile) {
218     std::ifstream inFile; // read data streams from files
219     std::ofstream outFile; // write data streams to files
220     std::string line;
221
222     try {
223         // RAI: Automatically opens the file - (ERR56-CPP)
224         inFile.open(sourcefile, std::ios::in); // Open source file for reading
225         outFile.open(destfile, std::ios::out | std::ios::trunc); // Open destination file for writing
226
227         // Process each line individually
228         while (std::getline(inFile, line)) {
229             // Reverse the characters in the current line using valid iterators (STR52-CPP)
230             std::reverse(line.begin(), line.end());
231             outFile << line << '\n'; // Write the reversed line to the destination file
232         }
233
234         inFile.close(); // Close source file (FIO51-CPP)
235         outFile.close(); // Close destination file (FIO51-CPP)
236     }
237     catch (const std::ios_base::failure& e) { // Unified exception type
238         // Handle file operation exceptions (ERR51-CPP)
239         if (!infile.is_open()) {
240             throw std::runtime_error("Failed to open or read from file '" + sourcefile + ". " + e.what());
241         }
242         else if (!outfile.is_open()) {
243             throw std::runtime_error("Failed to open or write to file '" + destfile + ". " + e.what());
244         }
245         else {
246             throw std::runtime_error("File operation failed: " + std::string(e.what()));
247         }
248     }
249 }
250 // -----

```

Figure 2
Source Code in IDE

CTA-5-Input-to-file.cpp:1 X

Module-5 CTA

```

1 //=====
2 // Program Name: User Input To File In Reverse
3 // Author: Alejandro (Alex) Ricciardi
4 // Date: 11/10/2024
5 //
6 // Requirement: C++23
7 //
8 // Program Description:
9 // This program prompts the user to enter a string (sentence) and appends it to the "CSC450_CT5_mod5.txt" file without
10 // deleting existing data.
11 // The program also validates the user input, trimming leading and trailing whitespaces from inputted text.
12 // It then reverses each line of "CSC450_CT5_mod5.txt" by reversing the characters in each line while maintaining
13 // the order of the lines, and stores the reversed content in "CSC450-mod5-reverse.txt".
14 //
15 // The program adheres to the following SEI CERT C++ Coding Standards:
16 // - STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator
17 // - STR52-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
18 // - STR53-CPP. Range check element access
19 // - FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
20 // - FIO51-CPP. Close files when they are no longer needed
21 // - ERR50-CPP. Do not abruptly terminate the program
22 // - ERR51-CPP. Handle all exceptions
23 // - ERR56-CPP. Guarantee exception safety
24 //
25 //=====
26
27 /*
28 | Libraries   |
29 |
30 */
31 #include <iostream>
32 #include <fstream>      // For std::ifstream and std::ofstream
33 #include <string>
34 #include <stdexcept>
35 #include <algorithm>    // For std::reverse
36 #include <limits>       // For numeric limits
37 #include <cstdlib>      // For EXIT_FAILURE, EXIT_SUCCESS
38
39 /*
40 | Global Variables   |
41 |
42 */
43 // Banner - multi-line string
44 const std::string banner = R"(
45 ***** * User Input To File In Reverse * *****
46 )";
47
48 // =====
49 /*
50 | Function Declaration   |
51 |
52 */
53
54 /**
55 * Prompts the user to enter a non-empty string and validates the input
56 * Trims leading and trailing whitespaces from inputted text
57 * It will prompt the user until valid input is entered
58 *
59 * @param prompt The message to display to the user
60 * @return A validated non-empty string.
61 */
62 std::string getValidatedInput(const std::string& prompt);
63
64 /**
65 * Appends a given string to a file
66 *
67 * @param filename The name of the file to append to
68 * @param data The string data to append
69 */
70 void appendToFile(const std::string& filename, const std::string& data);
71
72 /**
73 * Reverses the characters of each line in the source file and writes the lines in the same order to the destination file
74 *
75 * @param sourceFile The file content that needs to be processed
76 * @param destFile The file to save the processed content
77 */
78 void reverseFileContent(const std::string& sourceFile, const std::string& destFile);
79
80
81
82
83
84
85

```

```

86 // =====
87 /* -----
88 | Main Function |
89 -----
90
91     The main function runs the program.
92     Asks the user to enter data to be added to file.
93     Catch any possible exceptions.
94
95     Handles Rules:
96         - ERR50-CPP. Do not abruptly terminate the program
97         - ERR51-CPP. Handle all exceptions
98
99
100 // -----
101 int main() {
102     std::cout << banner << std::endl;
103
104     std::cout << "Welcome to the User Input to File Program!\n\n";
105
106     // Try-catch block to handle all exceptions (ERR51-CPP)
107     try {
108         std::string userInput = getValidatedInput("Enter data to append to CSC450_CT5_mod5.txt: ");
109
110         appendToFile("CSC450_CT5_mod5.txt", userInput + "\n"); // Append with newline
111
112         reverseFileContent("CSC450_CT5_mod5.txt", "CSC450-mod5-reverse.txt");
113
114         std::cout << "\nData has been successfully appended and reversed.\n";
115     }
116
117     catch (const std::exception& e) {
118         // Catch any standard exceptions thrown in the try block (ERR51-CPP)
119         std::cerr << "\nERROR: Exception caught: " << e.what() << std::endl;
120         return EXIT_FAILURE; // Exit the program with an error code (ERR50-CPP)
121     }
122
123     catch (...) {
124         // Catch any non-standard exceptions (ERR51-CPP)
125         std::cerr << "\nERROR: Unknown exception caught. Exiting program.\n";
126         return EXIT_FAILURE; // Exit the program with an error code (ERR50-CPP)
127     }
128
129     return EXIT_SUCCESS; // Successful program termination
130 }
131
132 /* -----
133 | Function Definitions |
134 -----
135 */
136 // -----
137
138 /**
139 * Prompts the user to enter a non-empty string and validates the input
140 * Trims leading and trailing whitespaces from inputted text
141 * It will prompt the user until valid input is entered
142 *
143 * @param prompt The message to display to the user
144 * @return A validated non-empty string.
145 */
146 std::string getValidatedInput(const std::string& prompt) {
147     while (true) {
148         std::string input;
149
150         std::cout << prompt;
151
152         // Get the entire line of input and store it in a string object
153         std::getline(std::cin, input); // Using std::string ensures sufficient storage (STR50-CPP)
154
155         // Trim leading and trailing whitespaces
156         size_t start = input.find_first_not_of(" \t\n\r"); // Valid index operation (STR53-CPP)
157         size_t end = input.find_last_not_of(" \t\n\r"); // Valid index operation (STR53-CPP)
158
159         if (start == std::string::npos) {
160             std::cerr << "Invalid input: Input cannot be empty or just whitespace. Please try again.\n";
161             continue; // Prompt the user again
162         }
163
164         // Trim the input
165         input = input.substr(start, end - start + 1); // Range-checked substring (STR53-CPP)
166
167     }
168
169     return input; // Validated input
170 }

```

```

172 // -----
173
174 /**
175 * Appends a given string to a file
176 */
177 /* Handles Rules:
178 * - FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
179 * - FIO51-CPP. Close files when they are no longer needed
180 * - ERR51-CPP. Handle all exceptions
181 * - ERR56-CPP. Guarantee exception safety
182 */
183 /* @param filename The name of the file to append to
184 * @param data The string data to append
185 */
186 void appendToFile(const std::string& filename, const std::string& data) {
187     std::ofstream outFile; // write data streams to files
188
189     try {
190         // RAI: Automatically opens the file - (ERR56-CPP)
191         outFile.open(filename, std::ios::app); // Open file in append mode
192         // No alternating input/output without repositioning (FIO50-CPP)
193         outFile << data; // Write data to file
194         outFile.close(); // Close file when done (FIO51-CPP)
195     }
196     catch (const std::ofstream::failure& e) {
197         // Handle file operation exceptions (ERR51-CPP)
198         throw std::runtime_error("Failed to open or write to file '" + filename + ".\"");
199     }
200 }
201
202 // -----
203
204 /**
205 * Reverses the characters of each line in the source file and writes the lines in the same order to the destination file
206 */
207 /* Handles Rules:
208 * - STR52-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
209 * - FIO50-CPP. Do not alternately input and output from a file stream without an intervening positioning call
210 * - FIO51-CPP. Close files when they are no longer needed
211 * - ERR51-CPP. Handle all exceptions
212 * - ERR56-CPP. Guarantee exception safety
213 */
214 /* @param sourceFile The file content that needs to be processed
215 * @param destFile The file to save the processed content
216 */
217 void reverseFileContent(const std::string& sourceFile, const std::string& destFile) {
218     std::ifstream inFile; // read data streams from files
219     std::ofstream outFile; // write data streams to files
220     std::string line;
221
222     try {
223         // RAI: Automatically opens the file - (ERR56-CPP)
224         inFile.open(sourceFile, std::ios::in); // Open source file for reading
225         outFile.open(destFile, std::ios::out | std::ios::trunc); // Open destination file for writing
226
227         // Process each line individually
228         while (std::getline(inFile, line)) {
229             // Reverse the characters in the current line using valid iterators (STR52-CPP)
230             std::reverse(line.begin(), line.end());
231             outFile << line << '\n'; // Write the reversed line to the destination file
232         }
233
234         inFile.close(); // Close source file (FIO51-CPP)
235         outFile.close(); // Close destination file (FIO51-CPP)
236     }
237     catch (const std::ios_base::failure& e) { // Unified exception type
238         // Handle file operation exceptions (ERR51-CPP)
239         if (!inFile.is_open()) {
240             throw std::runtime_error("Failed to open or read from file '" + sourceFile + ". " + e.what());
241         }
242         else if (!outFile.is_open()) {
243             throw std::runtime_error("Failed to open or write to file '" + destFile + ". " + e.what());
244         }
245         else {
246             throw std::runtime_error("File operation failed: " + std::string(e.what()));
247         }
248     }
249 }
250
251 // -----

```

Figure 3*Console Welcome Screen*

```
*****  
* User Input To File In Reverse *  
*****  
  
Welcome to the User Input to File Program!  
  
Enter data to append to CSC450_CT5_mod5.txt:
```

Figure 4*Outputs Console After Entering Data*

```
*****  
* User Input To File In Reverse *  
*****  
  
Welcome to the User Input to File Program!  
  
Enter data to append to CSC450_CT5_mod5.txt: STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator.  
  
Data has been successfully appended and reversed.  
  
P:\CSU projects\Programming-3-CSC450\Programming-3-cpp\Module-5 CTA\x64\Debug\Module-5 CTA.exe (process 1952) exited with code 0.  
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.  
Press any key to close this window . . .
```

Figure 5
Given Text File

```
CSC450_CT5_mod5.txt C:\...\Downloads
C: > Users > User > Downloads > CSC450_CT5_mod5.txt
1 Please be sure to append your data to this text file.
2
3 If these first three lines are deleted, then your program is not functioning as expected.
4
5
6

Ln 6, Col 1 Spaces: 4 UTF-8 CRLF Plain Text
```

Note: The given file, "CSC450_CT5_mod5.txt", has 6 lines, the user input will be appended on the 6th line. The file shown here is from the download folder.

Figure 5
Given Text File After User Input

```
CSC450_CT5_mod5.txt C:\...\Downloads
P: > CSU projects > Programming-3-CSC450 > Programming-3-cpp > Module-5 CTA > CSC450_CT5_mod5.txt
1 Please be sure to append your data to this text file.
2
3 If these first three lines are deleted, then your program is not functioning as expected.
4
5
6 STR50-CPP. Guarantee that storage for strings has sufficient space for character data and the null terminator.
7

Ln 7, Col 1 Spaces: 4 UTF-8 CRLF Plain Text
```

Note: The user input was appended on the 6th line in the "CSC450_CT5_mod5.txt" file.

Figure 6
Reverse Text in New File

```
CSC450_CT5_mod5.txt C:\...\Downloads
CSC450_CT5_mod5.txt P:\...\Module-5 CTA
CSC450-mod5-reverse.txt
1 .elif txt siht ot atad ruoy dneppa ot erus eb esaelP
2
3 .detcepexe sa gninoitcnuf ton si margorp ruoy neht ,detaled era senil eerht tsrif eseht fI
4
5
6 .rotanimret llun eht dna atad retcarahc rof ecaps tneiciffus sah sgnirts rof egarots taht eetnarauG .PPC-05RTS
7

Ln 1, Col 1 Spaces: 4 UTF-8 CRLF Plain Text
```

Note: A new file named "CSC450-mod5-reverse.txt" was created and each line from the file "CSC450_CT5_mod5.txt" was treated as a separate string, reversing only the characters in each line while maintaining the order of the lines

As shown in Figures 3 through 6 the program runs without any issues displaying the correct outputs as expected.