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

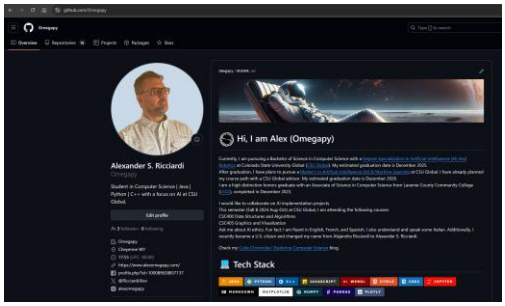
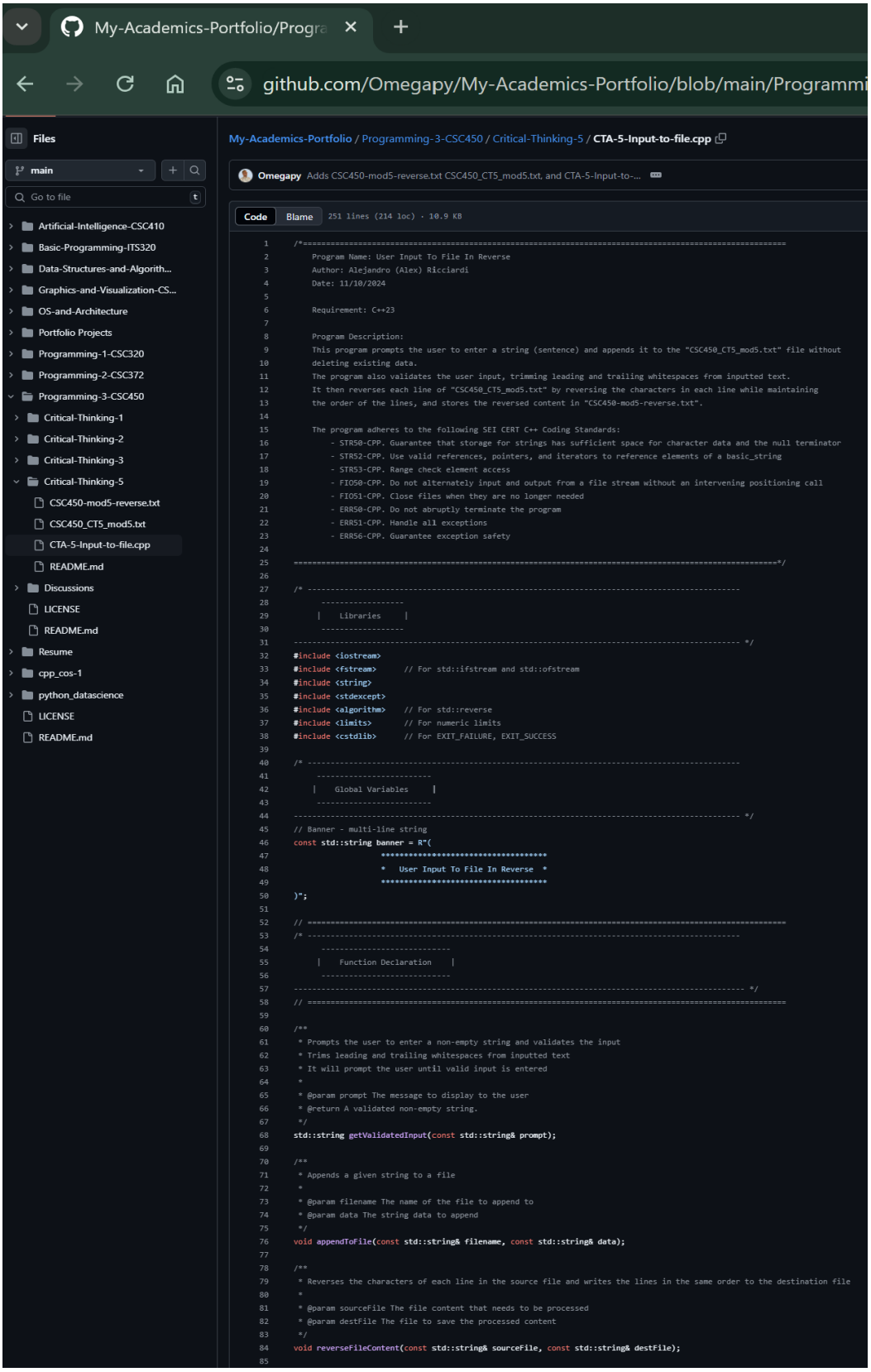
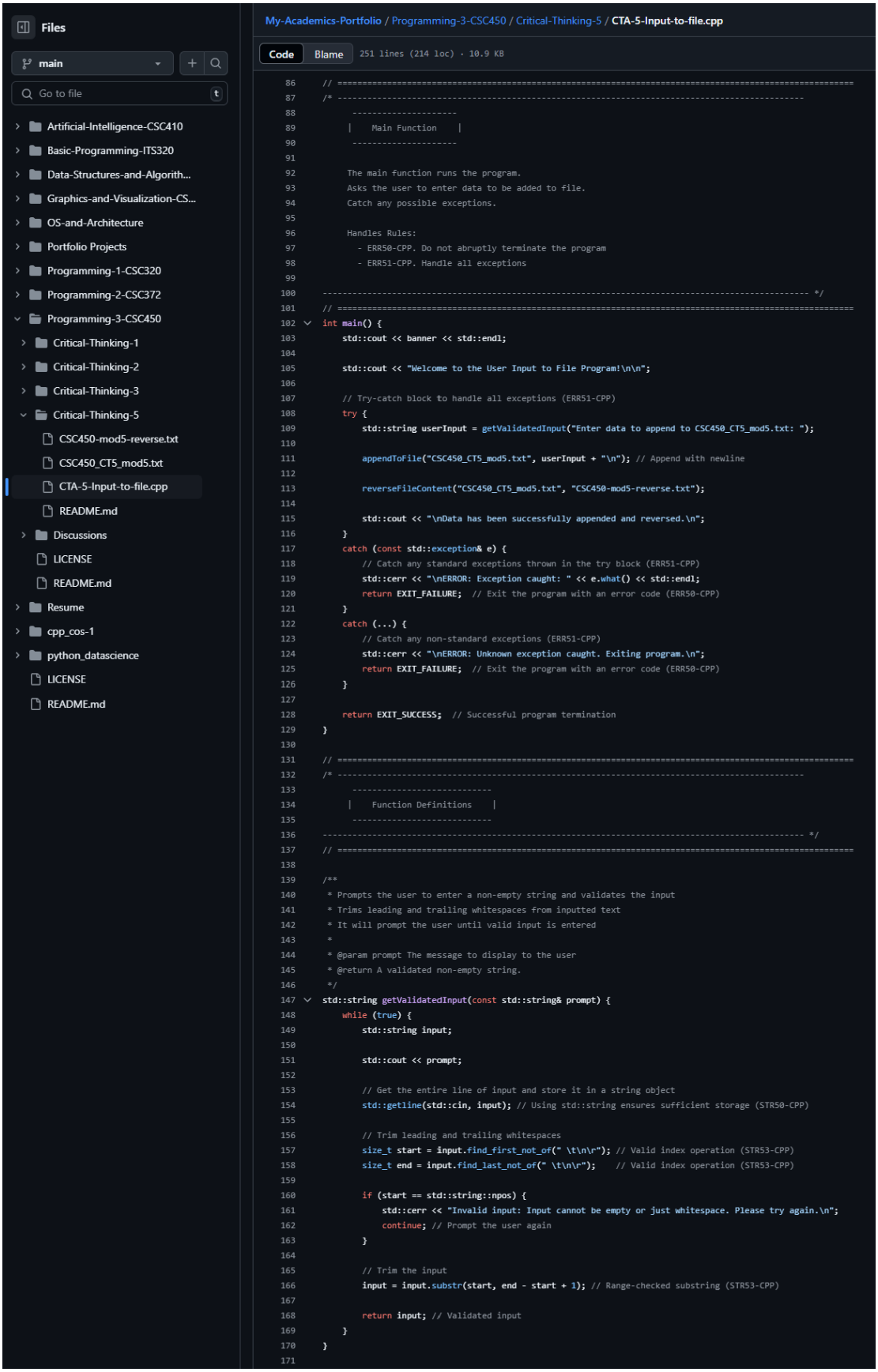


Image of the source code in the GitHub: see next page

Figure 1

Source Code in GitHub






```cpp
// ============================================================================================================
/* ------------------------------------------------------------------------------------------------------------
     --------------------
    |    Main Function    |
     --------------------

    The main function runs the program.
    Asks the user to enter data to be added to file.
    Catch any possible exceptions.

    Handles Rules:
      - ERR50-CPP. Do not abruptly terminate the program
      - ERR51-CPP. Handle all exceptions

------------------------------------------------------------------------------------------------------------ */
// ============================================================================================================
int main() {
    std::cout << banner << std::endl;

    std::cout << "Welcome to the User Input to File Program!\n\n";

    // Try-catch block to handle all exceptions (ERR51-CPP)
    try {
        std::string userInput = getValidatedInput("Enter data to append to CSC450_CT5_mod5.txt: ");

        appendToFile("CSC450_CT5_mod5.txt", userInput + "\n"); // Append with newline

        reverseFileContent("CSC450_CT5_mod5.txt", "CSC450-mod5-reverse.txt");

        std::cout << "\nData has been successfully appended and reversed.\n";
    }
    catch (const std::exception& e) {
        // Catch any standard exceptions thrown in the try block (ERR51-CPP)
        std::cerr << "\nERROR: Exception caught: " << e.what() << std::endl;
        return EXIT_FAILURE;  // Exit the program with an error code (ERR50-CPP)
    }
    catch (...) {
        // Catch any non-standard exceptions (ERR51-CPP)
        std::cerr << "\nERROR: Unknown exception caught. Exiting program.\n";
        return EXIT_FAILURE;  // Exit the program with an error code (ERR50-CPP)
    }

    return EXIT_SUCCESS;  // Successful program termination
}

// ============================================================================================================
/* ------------------------------------------------------------------------------------------------------------
     ---------------------------
    |    Function Definitions    |
     ---------------------------
------------------------------------------------------------------------------------------------------------ */
// ============================================================================================================

/**
 * Prompts the user to enter a non-empty string and validates the input
 * Trims leading and trailing whitespaces from inputted text
 * It will prompt the user until valid input is entered
 *
 * @param prompt The message to display to the user
 * @return A validated non-empty string.
 */
std::string getValidatedInput(const std::string& prompt) {
    while (true) {
        std::string input;

        std::cout << prompt;

        // Get the entire line of input and store it in a string object
        std::getline(std::cin, input); // Using std::string ensures sufficient storage (STR50-CPP)

        // Trim leading and trailing whitespaces
        size_t start = input.find_first_not_of(" \t\n\r"); // Valid index operation (STR53-CPP)
        size_t end = input.find_last_not_of(" \t\n\r");    // Valid index operation (STR53-CPP)

        if (start == std::string::npos) {
            std::cerr << "Invalid input: Input cannot be empty or just whitespace. Please try again.\n";
            continue; // Prompt the user again
        }

        // Trim the input
        input = input.substr(start, end - start + 1); // Range-checked substring (STR53-CPP)

        return input; // Validated input
    }
}
```

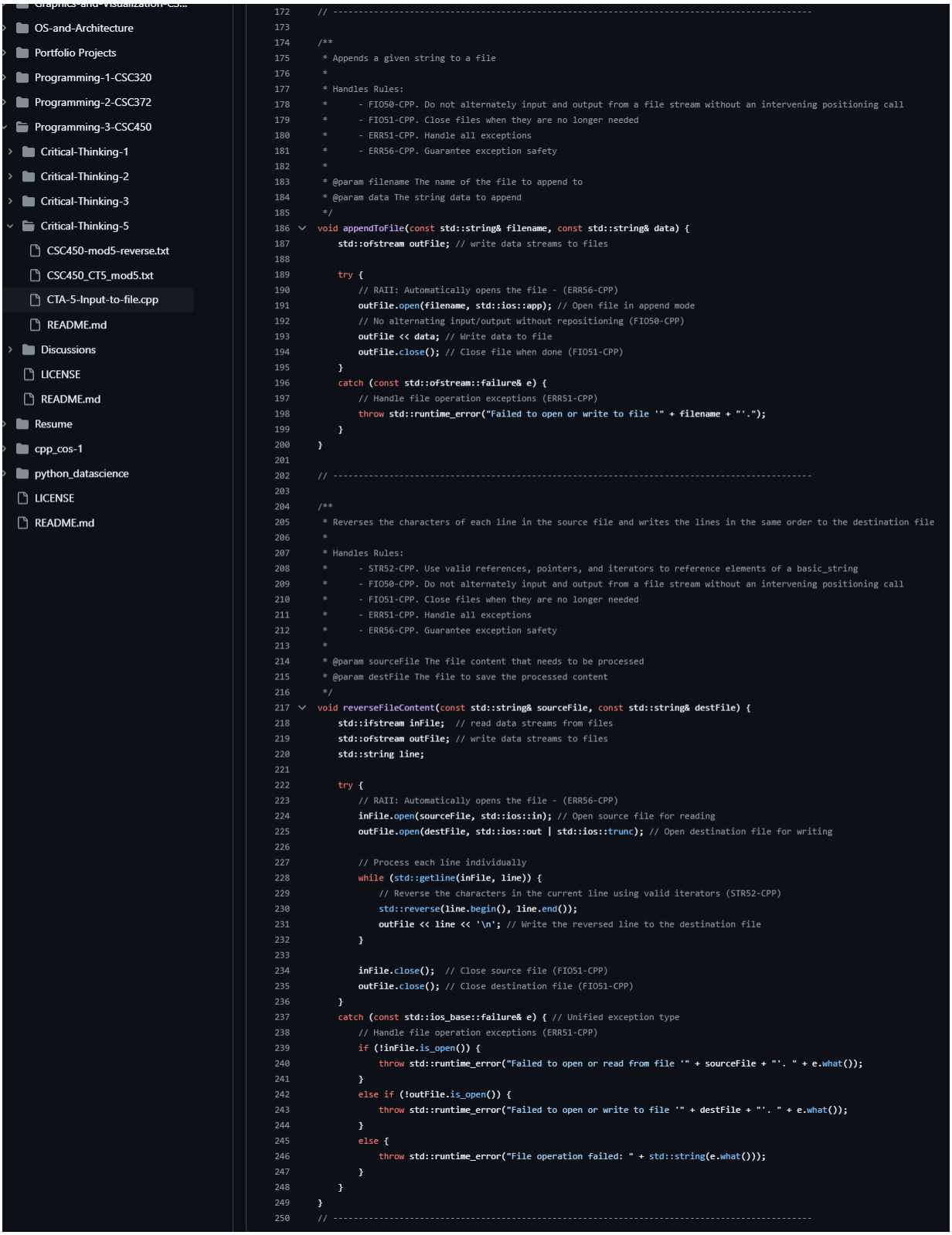

OS-and-Architecture
Portfolio Projects
Programming-1-CSC320
Programming-2-CSC372
Programming-3-CSC450
Critical-Thinking-1
Critical-Thinking-2
Critical-Thinking-3
Critical-Thinking-5
CSC450-mod5-reverse.txt
CSC450_CT5_mod5.txt
CTA-5-Input-to-file.cpp
README.md
Discussions
LICENSE
README.md
Resume
cpp_cos-1
python_datascience
LICENSE
README.md
// ----------
/**
* Appends a given string to a file
*
* Handles Rules:
* - 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
* - ERR51-CPP. Handle all exceptions
* - ERR56-CPP. Guarantee exception safety
*
* @param filename The name of the file to append to
* @param data The string data to append
*/
void appendToFile(const std::string& filename, const std::string& data) {
std::ofstream outFile; // write data streams to files
try {
// RAII: Automatically opens the file - (ERR56-CPP)
outFile.open(filename, std::ios::app); // Open file in append mode
// No alternating input/output without repositioning (FIO50-CPP)
outFile << data; // Write data to file
outFile.close(); // Close file when done (FIO51-CPP)
}
catch (const std::ofstream::failure& e) {
// Handle file operation exceptions (ERR51-CPP)
throw std::runtime_error("Failed to open or write to file '" + filename + "'.");
}
}
// ----------
/**
* Reverses the characters of each line in the source file and writes the lines in the same order to the destination file
*
* Handles Rules:
* - STR52-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
* - 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
* - ERR51-CPP. Handle all exceptions
* - ERR56-CPP. Guarantee exception safety
*
* @param sourceFile The file content that needs to be processed
* @param destFile The file to save the processed content
*/
void reverseFileContent(const std::string& sourceFile, const std::string& destFile) {
std::ifstream inFile; // read data streams from files
std::ofstream outFile; // write data streams to files
std::string line;
try {
// RAII: Automatically opens the file - (ERR56-CPP)
inFile.open(sourceFile, std::ios::in); // Open source file for reading
outFile.open(destFile, std::ios::out | std::ios::trunc); // Open destination file for writing
// Process each line individually
while (std::getline(inFile, line)) {
// Reverse the characters in the current line using valid iterators (STR52-CPP)
std::reverse(line.begin(), line.end());
outFile << line << '\n'; // Write the reversed line to the destination file
}
inFile.close(); // Close source file (FIO51-CPP)
outFile.close(); // Close destination file (FIO51-CPP)
}
catch (const std::ios_base::failure& e) { // Unified exception type
// Handle file operation exceptions (ERR51-CPP)
if (!inFile.is_open()) {
throw std::runtime_error("Failed to open or read from file '" + sourceFile + "'. " + e.what());
}
else if (!outFile.is_open()) {
throw std::runtime_error("Failed to open or write to file '" + destFile + "'. " + e.what());
}
else {
throw std::runtime_error("File operation failed: " + std::string(e.what()));
}
}
}
// ----------

Figure 2

Source Code in IDE

```cpp
// ================================================================================================
/* ------------------------------------------------------------------------------------------------
     ---------------------
    |    Main Function    |
     ---------------------

    The main function runs the program.
    Asks the user to enter data to be added to file.
    Catch any possible exceptions.

    Handles Rules:
      - ERR50-CPP. Do not abruptly terminate the program
      - ERR51-CPP. Handle all exceptions

------------------------------------------------------------------------------------------------ */
// ================================================================================================
int main() {
    std::cout << banner << std::endl;

    std::cout << "Welcome to the User Input to File Program!\n\n";

    // Try-catch block to handle all exceptions (ERR51-CPP)
    try {
        std::string userInput = getValidatedInput("Enter data to append to CSC450_CT5_mod5.txt: ");

        appendToFile("CSC450_CT5_mod5.txt", userInput + "\n"); // Append with newline

        reverseFileContent("CSC450_CT5_mod5.txt", "CSC450-mod5-reverse.txt");

        std::cout << "\nData has been successfully appended and reversed.\n";
    }
    catch (const std::exception& e) {
        // Catch any standard exceptions thrown in the try block (ERR51-CPP)
        std::cerr << "\nERROR: Exception caught: " << e.what() << std::endl;
        return EXIT_FAILURE;  // Exit the program with an error code (ERR50-CPP)
    }
    catch (...) {
        // Catch any non-standard exceptions (ERR51-CPP)
        std::cerr << "\nERROR: Unknown exception caught. Exiting program.\n";
        return EXIT_FAILURE;  // Exit the program with an error code (ERR50-CPP)
    }

    return EXIT_SUCCESS;  // Successful program termination
}

// ================================================================================================
/* ------------------------------------------------------------------------------------------------
     ----------------------------
    |    Function Definitions    |
     ----------------------------
------------------------------------------------------------------------------------------------ */
// ================================================================================================

/**
* Prompts the user to enter a non-empty string and validates the input
* Trims leading and trailing whitespaces from inputted text
* It will prompt the user until valid input is entered
*
* @param prompt The message to display to the user
* @return A validated non-empty string.
*/
std::string getValidatedInput(const std::string& prompt) {
    while (true) {
        std::string input;

        std::cout << prompt;

        // Get the entire line of input and store it in a string object
        std::getline(std::cin, input); // Using std::string ensures sufficient storage (STR50-CPP)

        // Trim leading and trailing whitespaces
        size_t start = input.find_first_not_of(" \t\n\r"); // Valid index operation (STR53-CPP)
        size_t end = input.find_last_not_of(" \t\n\r");    // Valid index operation (STR53-CPP)

        if (start == std::string::npos) {
            std::cerr << "Invalid input: Input cannot be empty or just whitespace. Please try again.\n";
            continue; // Prompt the user again
        }

        // Trim the input
        input = input.substr(start, end - start + 1); // Range-checked substring (STR53-CPP)

        return input; // Validated input
    }
}

//
```

```cpp
// ------------------------------------------------------------------------------------------

/**
 * Appends a given string to a file
 *
 * Handles Rules:
 *      - 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
 *      - ERR51-CPP. Handle all exceptions
 *      - ERR56-CPP. Guarantee exception safety
 *
 * @param filename The name of the file to append to
 * @param data The string data to append
 */
void appendToFile(const std::string& filename, const std::string& data) {
    std::ofstream outFile; // write data streams to files

    try {
        // RAII: Automatically opens the file - (ERR56-CPP)
        outFile.open(filename, std::ios::app); // Open file in append mode
        // No alternating input/output without repositioning (FIO50-CPP)
        outFile << data; // Write data to file
        outFile.close(); // Close file when done (FIO51-CPP)
    }
    catch (const std::ofstream::failure& e) {
        // Handle file operation exceptions (ERR51-CPP)
        throw std::runtime_error("Failed to open or write to file '" + filename + "'.");
    }
}

// ------------------------------------------------------------------------------------------

/**
 * Reverses the characters of each line in the source file and writes the lines in the same order to the destination file
 *
 * Handles Rules:
 *      - STR52-CPP. Use valid references, pointers, and iterators to reference elements of a basic_string
 *      - 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
 *      - ERR51-CPP. Handle all exceptions
 *      - ERR56-CPP. Guarantee exception safety
 *
 * @param sourceFile The file content that needs to be processed
 * @param destFile The file to save the processed content
 */
void reverseFileContent(const std::string& sourceFile, const std::string& destFile) {
    std::ifstream inFile;  // read data streams from files
    std::ofstream outFile; // write data streams to files
    std::string line;

    try {
        // RAII: Automatically opens the file - (ERR56-CPP)
        inFile.open(sourceFile, std::ios::in); // Open source file for reading
        outFile.open(destFile, std::ios::out | std::ios::trunc); // Open destination file for writing

        // Process each line individually
        while (std::getline(inFile, line)) {
            // Reverse the characters in the current line using valid iterators (STR52-CPP)
            std::reverse(line.begin(), line.end());
            outFile << line << '\n'; // Write the reversed line to the destination file
        }

        inFile.close();  // Close source file (FIO51-CPP)
        outFile.close(); // Close destination file (FIO51-CPP)
    }
    catch (const std::ios_base::failure& e) { // Unified exception type
        // Handle file operation exceptions (ERR51-CPP)
        if (!inFile.is_open()) {
            throw std::runtime_error("Failed to open or read from file '" + sourceFile + "'. " + e.what());
        }
        else if (!outFile.is_open()) {
            throw std::runtime_error("Failed to open or write to file '" + destFile + "'. " + e.what());
        }
        else {
            throw std::runtime_error("File operation failed: " + std::string(e.what()));
        }
    }
}
// ------------------------------------------------------------------------------------------

```

Figure 3
Console Welcome Screen

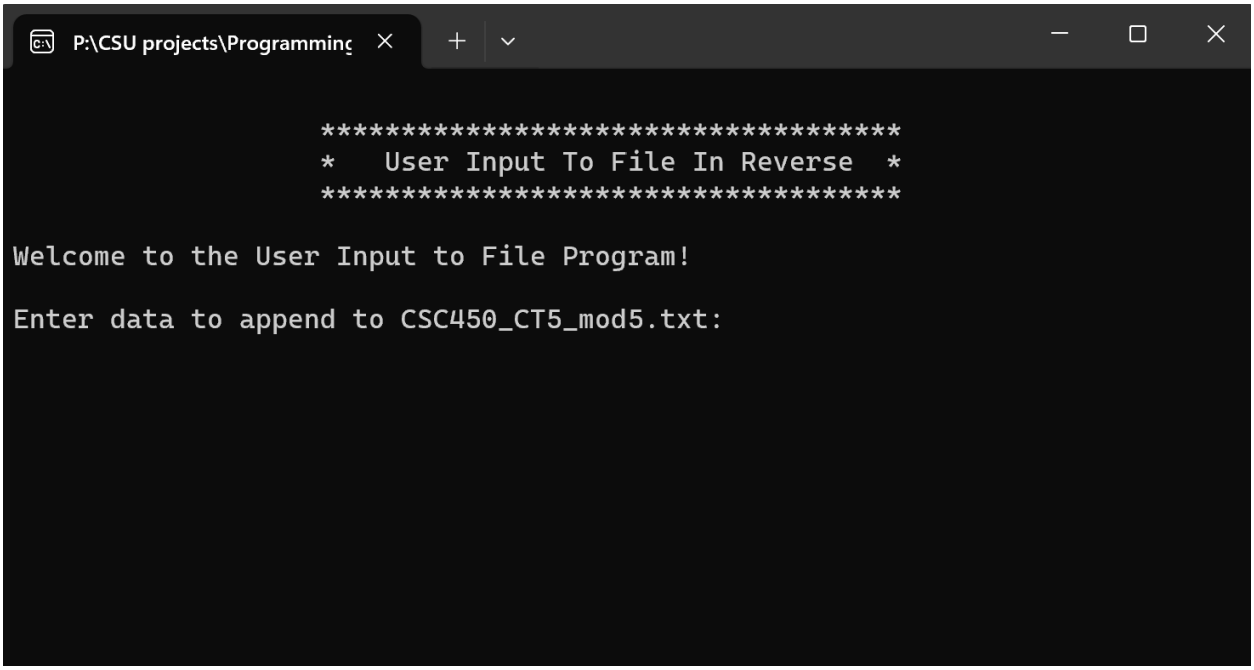


Figure 4
Outputs Console After Entering Data

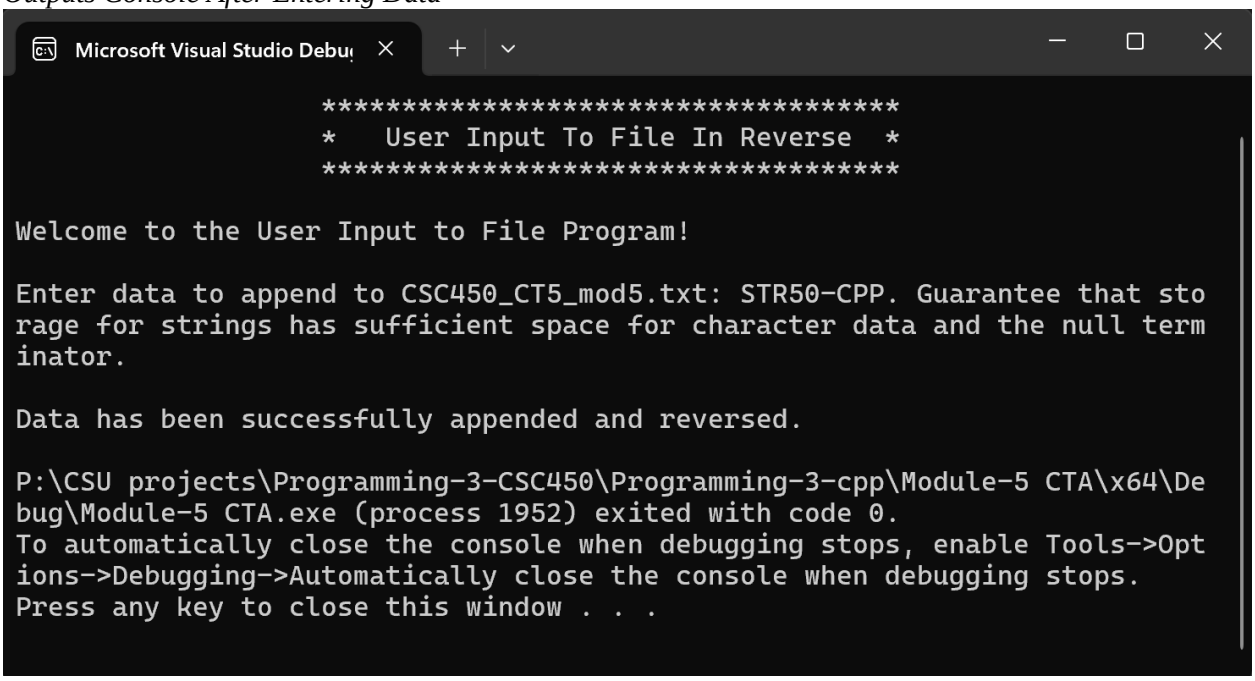
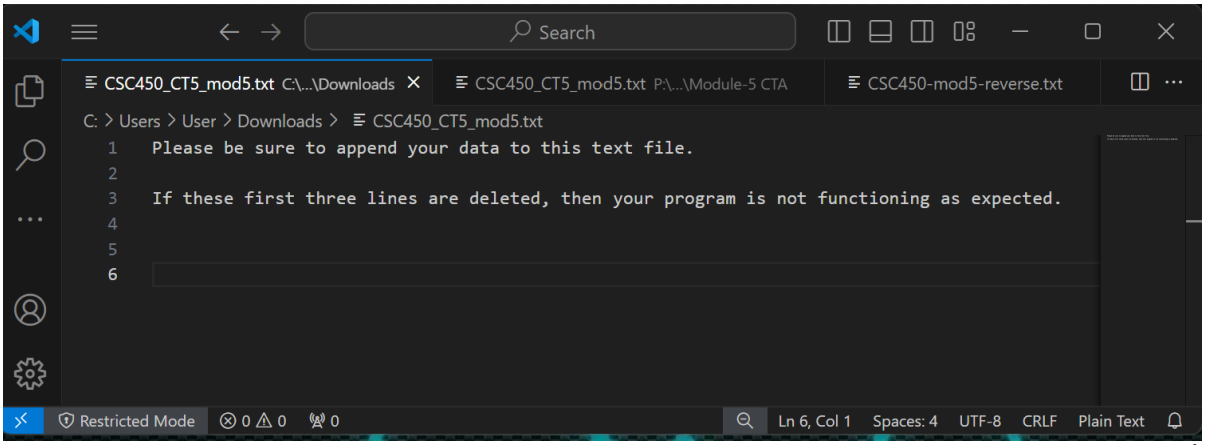


Figure 5

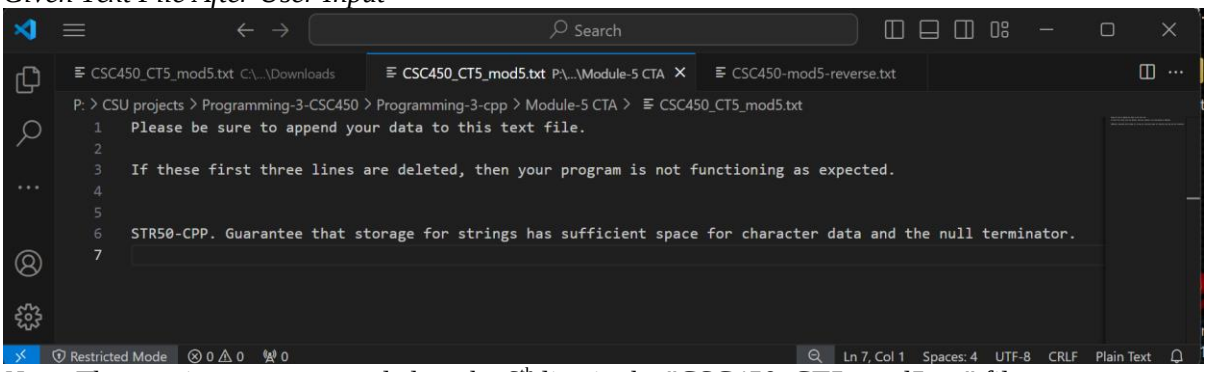
Given Text File



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

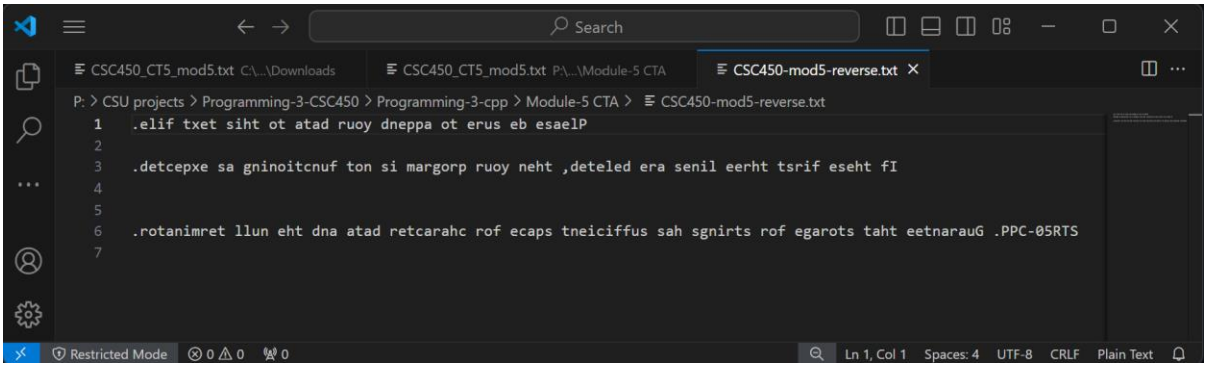
Given Text File After User Input



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

Figure 6

Reverse Text in New File



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.