Course Code: CSE-2104
Course Title: Object Oriented Programming Laboratory
Lab 8: Implementation of Files in Object Oriented Programming.
Date: 27/10/2024
Submission: 03/11/2024

Task:

1. Write a C++ program to create a new text file and write some text into it.
2. Write a C++ program to open an existing text file and display its contents on the console.
3. Write a C++ program to count the number of lines in a text file
4. Write a C++ program to count the number of words in a text file.
5. Write a C++ program to copy the contents of one text file to another.
6. Write a C++ program to find and replace a specific word in a text file.
7. Write a C++ program to append new data to an existing text file.
8. Write a C++ program to merge multiple text files into a single file.
9. Write a C++ program to encrypt the contents of a text file using a simple encryption algorithm.
10. Write a C++ program to decrypt the contents of a text file encrypted using the above algorithm.

Q1 Soln:

```
#include <iostream>  // Include the input/output stream library

#include <fstream>   // Include the file stream library

int main() {

  // Create a new file named "test.txt"

  std::ofstream outputFile("test.txt");   // Open/create a file named "test.txt" for writing

  if (outputFile.is_open()) {  // Check if the file was successfully opened

    // Write some text into the file

    outputFile << "C++ is a high-level language\n";  // Write a line of text to the file

    outputFile << "Modern C++ currently has object-oriented features\n"; // Write a line of text to the file

    // Close the file

    outputFile.close();  // Close the file after writing

    std::cout << "Text has been written to the file." << std::endl;  // Display a success message

  } else {

    std::cout << "Failed to create the file." << std::endl;  // Display an error message if file creation failed

  }

  return 0;  // Return 0 to indicate successful execution

}
```

Q2 Soln:

```
#include <iostream>   // Including the input/output stream library

#include <fstream>    // Including the file stream library

#include <string>     // Including the string handling library

int main() {

  // Open an existing file named "test.txt"

  std::ifstream inputFile("test.txt");    // Opening the file named "test.txt" for reading

  if (inputFile.is_open()) {   // Checking if the file was successfully opened

    std::string line;  // Declaring a string variable to store each line of text

    while (std::getline(inputFile, line)) {  // Loop through each line in the file

      // Display each line on the console

      std::cout << line << std::endl;  // Output the content of 'line' to the console

    }

    inputFile.close();  // Closing the file after reading

  } else {

    std::cout << "Failed to open the file." << std::endl;  // Display an error message if file opening failed

  }

  return 0;  // Return 0 to indicate successful execution

}
```

Q3 Soln:

```
#include <iostream>   // Including the input/output stream library

#include <fstream>    // Including the file stream library

#include <string>     // Including the string handling library

int main() {

  // Open the text file

  std::ifstream inputFile("test.txt");    // Opening the file named "test.txt" for reading

  if (inputFile.is_open()) {   // Checking if the file was successfully opened

    std::string line;  // Declaring a string variable to store each line of text

    int lineCount = 0;  // Initializing a variable to count lines

    while (std::getline(inputFile, line)) {  // Loop through each line in the file

      lineCount++;  // Incrementing line count for each line read

    }

    inputFile.close();  // Closing the file after counting lines

    std::cout << "Number of lines in the file: " << lineCount << std::endl; // Outputting the total line count

  } else {

    std::cout << "Failed to open the file." << std::endl;  // Display an error message if file opening failed

  }

  return 0;  // Return 0 to indicate successful execution

}
```

Q4 Soln:

```
#include <iostream>     // Including the input/output stream library
#include <fstream>      // Including the file stream library
#include <string>       // Including the string handling library
#include <sstream>      // Including the stringstream library


int main() {
  std::ifstream inputFile("test.txt"); // Open the text file named "test.txt" for reading
  if (inputFile.is_open()) {  // Checking if the file was successfully opened
    std::string line;         // Declaring a string variable to store each line of text
    int wordCount = 0;        // Initializing a variable to count words
    while (std::getline(inputFile, line)) {  // Loop through each line in the file
      std::stringstream ss(line);  // Create a stringstream object with the current line content
      std::string word;  // Declare a string variable to store each word
      while (ss >> word) {  // Extract words from the stringstream
        wordCount++;  // Increment word count for each word extracted
      }
    }
    inputFile.close();  // Closing the file after counting words
    std::cout << "Number of words in the said file: " << wordCount << std::endl;  // Outputting the total word count
  } else {
    std::cout << "Failed to open the file." << std::endl;  // Display an error message if file opening failed
  }
  return 0;  // Return 0 to indicate successful execution
}
```

Q5 Soln:

```
#include <iostream>   // Including the input/output stream library

#include <fstream>    // Including the file stream library

#include <string>     // Including the string handling library

int main() {

  // Open the input file

  std::ifstream inputFile("test.txt");       // Opening the file named "test.txt" for reading

  // Create or overwrite the output file

  std::ofstream outputFile("test_copy.txt");  // Creating/overwriting the file named "test_copy.txt" for writing

  if (inputFile.is_open() && outputFile.is_open()) {  // Checking if both input and output files were successfully opened

    std::string line;  // Declaring a string variable to store each line of text

    while (std::getline(inputFile, line)) {  // Loop through each line in the input file

      // Write each line to the output file

      outputFile << line << "\n";  // Writing each line to the output file with a newline character

    }

    inputFile.close();   // Closing the input file after copying

    outputFile.close();  // Closing the output file after copying

    std::cout << "File copied successfully." << std::endl;  // Displaying success message

  } else {

    std::cout << "Failed to open the files." << std::endl;  // Display an error message if file opening failed

  }

  return 0;  // Return 0 to indicate successful execution

}
```

Q6 Soln:

```
#include <iostream>   // Including the input/output stream library
#include <fstream>    // Including the file stream library
#include <string>     // Including the string handling library

// Function to display the content of a file
void displayFileContent(const std::string & filename) {
  std::ifstream file(filename); // Open file with given filename
  std::string line; // Declare a string to store each line of text

  if (file.is_open()) { // Check if the file was successfully opened
    std::cout << "File content:" << std::endl; // Displaying a message indicating file content
    while (std::getline(file, line)) { // Read each line from the file
      std::cout << line << std::endl; // Display each line of the file
    }
    file.close(); // Close the file
  } else {
    std::cout << "Failed to open the file." << std::endl; // Display an error message if file opening failed
  }
}
int main() {
  std::ifstream inputFile("test.txt"); // Open the input file named "test.txt" for reading
  std::ofstream outputFile("new_test.txt"); // Create or overwrite the output file named "new_test.txt" for writing
  if (inputFile.is_open() && outputFile.is_open()) { // Check if both input and output files were successfully opened
    std::string line; // Declare a string variable to store each line of text
    std::string searchWord = "C++"; // Define the word to search for
    std::string replaceWord = "CPP"; // Define the word to replace with

    std::cout << "Search word:" << searchWord << std::endl; // Display the word to search for
```

```
    std::cout << "Replace word:" << replaceWord << std::endl; // Display the word to replace with

    std::cout << "\nBefore find and replace:" << std::endl; // Display a message before find and replace

    displayFileContent("test.txt"); // Display the content of the input file before find and replace

    while (std::getline(inputFile, line)) { // Loop through each line in the input file

      size_t pos = line.find(searchWord); // Find the position of the search word in the line


      while (pos != std::string::npos) { // Repeat until all occurrences are replaced

        line.replace(pos, searchWord.length(), replaceWord); // Replace the search word with the replace word

        pos = line.find(searchWord, pos + replaceWord.length()); // Find the next occurrence of the search word

      }

      outputFile << line << "\n"; // Write the modified line to the output file

    }

    inputFile.close(); // Close the input file

    outputFile.close(); // Close the output file


    std::cout << "After find and replace:" << std::endl; // Display a message after find and replace

    displayFileContent("new_test.txt"); // Display the content of the output file after find and replace


    std::cout << "\nWord replaced successfully." << std::endl; // Display a success message

  } else {

    std::cout << "\nFailed to open the files." << std::endl; // Display an error message if file opening failed

  }

  return 0; // Return 0 to indicate successful execution

}
```

Q7 Soln:

```
#include <iostream>   // Including the input/output stream library
#include <fstream>    // Including the file stream library
#include <string>     // Including the string handling library
// Function to display the content of a file
void displayFileContent(const std::string & filename) {
  std::ifstream file(filename); // Open file with given filename for reading
  std::string line; // Declare a string to store each line of text
  if (file.is_open()) { // Check if the file was successfully opened
    std::cout << "File content:" << std::endl; // Displaying a message indicating file content
    while (std::getline(file, line)) { // Read each line from the file
      std::cout << line << std::endl; // Display each line of the file
    }
    file.close(); // Close the file
  } else {
    std::cout << "Failed to open the file." << std::endl; // Display an error message if file opening failed
  }
}
int main() {
  displayFileContent("new_test.txt");    //    Display    content    of "new_test.txt" before any modification
  std::cout << std::endl;
  std::ofstream outputFile; // Declare an output file stream object
  // Open the file in append mode
  outputFile.open("new_test.txt", std::ios::app); // Open "new_test.txt" in append mode
  displayFileContent("new_test.txt");    //    Display    content    of "new_test.txt" after opening in append mode
  std::cout << std::endl;
  if (outputFile.is_open()) { // Check if the file was successfully opened
    std::string newData; // Declare a string to store new data entered by the user
    std::cout << "Enter the data to append: "; // Prompt the user to enter data
```

```
    // Read the new data from the user
    std::getline(std::cin, newData); // Get user input for new data
    // Append the new data to the file
    outputFile << newData << std::endl; // Write the new data to the file
    outputFile.close(); // Close the file
    std::cout << "Data appended successfully." << std::endl; // Display a success message
    displayFileContent("new_test.txt");    //    Display    content    of "new_test.txt" after appending data
    std::cout << std::endl;
  } else {
    std::cout << "Failed to open the file." << std::endl; // Display an error message if file opening failed
  }
  return 0; // Return 0 to indicate successful execution
}
```

Q8 Soln:

```
#include <iostream>   // Including the input/output stream library
#include <fstream>    // Including the file stream library
#include <string>     // Including the string handling library
#include <vector>     // Including the vector container
// Function to display the content of a file
void displayFileContent(const std::string & filename) {
  std::ifstream file(filename); // Open file with given filename for reading
  std::string line; // Declare a string to store each line of text

  if (file.is_open()) { // Check if the file was successfully opened
    std::cout << "File content:" << std::endl; // Displaying a message indicating file content
    while (std::getline(file, line)) { // Read each line from the file
      std::cout << line << std::endl; // Display each line of the file
    }
    file.close(); // Close the file
  } else {
    std::cout << "Failed to open the file." << std::endl; // Display an error message if file opening failed
  }
}
int main() {
  std::vector<std::string> inputFiles = { // List of input files
    "test1.txt",
    "test2.txt",
    "test3.txt",
    "test4.txt"
  };
  std::cout << "Content of test1.txt, test2.txt, test3.txt, text4.txt: " << std::endl;

  displayFileContent("test1.txt"); // Display content of "test1.txt"
```

```
  displayFileContent("test2.txt"); // Display content of "test2.txt"
  displayFileContent("test3.txt"); // Display content of "test3.txt"
  displayFileContent("test4.txt"); // Display content of "test4.txt"
  std::string outputFile = "merged_test_file.txt"; // Output file
  std::ofstream mergedFile(outputFile); // Create or overwrite the output file named "merged_test_file.txt" for writing
  if (mergedFile.is_open()) { // Check if the output file was successfully opened
    for (const auto & inputFile: inputFiles) { // Iterate through each input file
      std::ifstream inputFileStream(inputFile); // Open each input file for reading
      if (inputFileStream.is_open()) { // Check if the input file was successfully opened
        std::string line; // Declare a string to store each line of text
        while (std::getline(inputFileStream, line)) { // Read each line from the input file
          mergedFile << line << "\n"; // Write each line to the merged file
        }
        inputFileStream.close(); // Close the input file
      } else {
        std::cout << "Failed to open input file: " << inputFile << std::endl; // Display an error message if file opening failed
      }
    }
    mergedFile.close(); // Close the merged file
    std::cout << "\nFiles merged successfully." << std::endl; // Display a success message
    std::cout << "\nContent of the merged file:" << std::endl;
    displayFileContent("merged_test_file.txt"); // Display content of "merged_test_file.txt"
  } else {
    std::cout << "Failed to open the output file." << std::endl; // Display an error message if output file opening failed
  }
  return 0; // Return 0 to indicate successful execution
}
```

Q9 Soln:

```
#include <iostream>   // Including the input/output stream library
#include <fstream>    // Including the file stream library
#include <string>     // Including the string handling library

// Function to display the content of a file
void displayFileContent(const std::string & filename) {
  std::ifstream file(filename); // Open file with given filename for reading
  std::string line; // Declare a string to store each line of text

  if (file.is_open()) { // Check if the file was successfully opened
    std::cout << "File content:" << std::endl; // Displaying a message indicating file content
    while (std::getline(file, line)) { // Read each line from the file
      std::cout << line << std::endl; // Display each line of the file
    }
    file.close(); // Close the file
  } else {
    std::cout << "Failed to open the file." << std::endl; // Display an error message if file opening failed
  }
}

// Function to encrypt a file using a simple algorithm (incrementing ASCII values)
void encryptFile(const std::string & inputFile, const std::string & outputFile) {
  std::ifstream input(inputFile); // Open input file for reading
  std::ofstream output(outputFile); // Open or create output file for writing

  if (input.is_open() && output.is_open()) { // Check if both files were successfully opened
    char ch; // Declare a character variable to read characters from the input file
```

```
    while (input.get(ch)) { // Loop through each character in the input file
      ch++; // Simple encryption algorithm: Increment ASCII value by 1
      output.put(ch); // Write the encrypted character to the output file
    }
    input.close(); // Close the input file
    output.close(); // Close the output file
    std::cout << "File encrypted successfully.\n" << std::endl; // Display a success message
  } else {
    std::cout << "Failed to open the files.\n" << std::endl; // Display an error message if file opening failed
  }
}
int main() {
  std::string inputFile = "test.txt"; // Input file
  displayFileContent("test.txt"); // Display content of "test.txt"
  std::cout << std::endl; // Output a newline for formatting

  std::string outputFile = "encrypted_test.txt"; // Output file for encrypted content
  encryptFile(inputFile, outputFile); // Encrypt "test.txt" and write to "encrypted_test.txt"
  displayFileContent("encrypted_test.txt"); // Display content of "encrypted_test.txt"
  std::cout << std::endl; // Output a newline for formatting

  return 0; // Return 0 to indicate successful execution
}
```

Q10 Soln:

```
#include <iostream>   // Including the input/output stream library
#include <fstream>    // Including the file stream library
#include <string>     // Including the string handling library

// Function to display the content of a file
void displayFileContent(const std::string & filename) {
  std::ifstream file(filename); // Open file with given filename for reading
  std::string line; // Declare a string to store each line of text

  if (file.is_open()) { // Check if the file was successfully opened
    std::cout << "File content:" << std::endl; // Displaying a message indicating file content
    while (std::getline(file, line)) { // Read each line from the file
      std::cout << line << std::endl; // Display each line of the file
    }
    file.close(); // Close the file
  } else {
    std::cout << "Failed to open the file." << std::endl; // Display an error message if file opening failed
  }
}

// Function to decrypt a file using a simple algorithm (decrementing ASCII values)
void decryptFile(const std::string & inputFile, const std::string & outputFile) {
  std::ifstream input(inputFile); // Open input file for reading
  std::ofstream output(outputFile); // Open or create output file for writing

  if (input.is_open() && output.is_open()) { // Check if both files were successfully opened
    char ch; // Declare a character variable to read characters from the input file
```

```
    while (input.get(ch)) { // Loop through each character in the input
file
      ch--; // Simple decryption algorithm: Decrement ASCII value by 1
      output.put(ch); // Write the decrypted character to the output file
    }

    input.close(); // Close the input file
    output.close(); // Close the output file

    std::cout << "File decrypted successfully.\n" << std::endl; // Display
a success message
  } else {
    std::cout << "Failed to open the files.\n" << std::endl; // Display
an error message if file opening failed
  }
}

int main() {
  std::string inputFile = "encrypted_test.txt"; // Input file (encrypted)
  displayFileContent("encrypted_test.txt");   //   Display   content   of
"encrypted_test.txt"
  std::cout << std::endl; // Output a newline for formatting

  std::string   outputFile   =   "decrypted_test.txt";   //   Output   file
(decrypted)
  decryptFile(inputFile, outputFile); // Decrypt "encrypted_test.txt" and
write to "decrypted_test.txt"
  displayFileContent("decrypted_test.txt");   //   Display   content   of
"decrypted_test.txt"
  std::cout << std::endl; // Output a newline for formatting

  return 0; // Return 0 to indicate successful execution
}
```