DAY- 14 Assignment

Topic : File Handling

DATE : 14 / 06/ 2024

// (* Important ***) Implementation of code on File Handling**

```
/* #include <iostream>
#include <fstream> // File input / output stream
#include <string>

using namespace std;

int main() {
    string fileName;
    char choice;

    // Get file name from user
    cout << "Enter the file name: ";
    cin >> fileName;

    // Get user's choice for operation (read or write)
    cout << "Enter '1' to read from the file or '2' to write to the file: ";
    cin >> choice;

    if (choice == '1') {
```

```
    // Open the file in read mode
    ifstream inputFile(fileName);

    if (inputFile.is_open()) {
        string line;

        // Read data from the file and print line by line
        while (getline(inputFile, line)) {
            cout << line << endl;
        }

        inputFile.close();
    } else {
        cout << "Error opening file for reading." << endl;
    }
} else if (choice == '2') {
    // Open the file in write mode (truncates existing content)
    ofstream outputFile(fileName);

    if (outputFile.is_open()) {
        string content;

        // Get content from user to write to the file
        cout << "Enter the content to write to the file: ";
        getline(cin, content, '\n');     // Include newline character
```

```
            outputFile << content << endl;


            outputFile.close();
            cout << "Content written to the file successfully." << endl;
        } else {
            cout << "Error opening file for writing." << endl;
        }
    } else {
        cout << "Invalid choice. Please enter '1' or '2'." << endl;
    }


    return 0;
} */
```

/* Problem 1: Read from a File

Task: 1

1. **Write a C++ program that reads a text file named input.txt and prints its content to the console.** */

```
/* #include <iostream>
#include <fstream>
#include <string>
using namespace std;
```

```
int main() {
    // Define the file name
    const string fileName = "input.txt";

    // Create an ifstream object for file input
    ifstream inputFile(fileName);

    // Check if the file is open
    if (!inputFile.is_open()) {
        cout << "Could not open the file " << fileName <<endl;
        return 1; // Return with error code
    }

    // Variable to hold each line of the file
    string line;

    // Read the file line by line and print each line to the console
    while (getline(inputFile, line)) {
        cout << line <<endl;
    }

    // Close the file
    inputFile.close();

    return 0; // Return with success code
```

} *

/* Questions: 1

1. How do you open a file for reading in C++?

// In C++, if we want to open a file for reading we use the ' ifstream' class from the '<fstream>' library.

Eg : cout << "Enter the file name: ";

cin >> fileName;

```
// Get user's choice for operation (read or write)
cout << "Enter '1' to read from the file or '2' to write to the file: ";
cin >> choice;

if (choice == '1') {
    // Open the file in read mode ( Required code )
    ifstream inputFile(fileName);

    if (inputFile.is_open()) {
        string line;

        // Read data from the file and print line by line
        while (getline(inputFile, line)) {
            cout << line << endl;
        }

        inputFile.close();
```

```
    } else {
        cout << "Error opening file for reading." << endl;
    }
}
```

2. What is the purpose of the ifstream class in C++?

// The purpose of ifstream class in C++ is used to open the file for reading .

3. How can you check if a file was successfully opened?

// We can check if a file was successfully opened in C++, by using the is_open() member function of the ifstream class. The is_open() function returns true if the file is successfully opened, and false otherwise.

Eg :

```
if (inputFile.is_open()) {
    string line;

    // Read data from the file and print line by line
    while (getline(inputFile, line)) {
        cout << line << endl;
    }

    inputFile.close();
} else {
    cout << "Error opening file for reading." << endl;
}
```

4. What function do you use to read a line from a file?

// We can read a line from a file in C++, by using the getline() function. The getline function reads characters from an input stream and stores them into a string until it a newline character ('\n') or the end of the file ('endl').

Eg : // Read data from the file and print line by line

```
while (getline(inputFile, line)) {
    cout << line << endl;
}
```

5. How do you properly close a file after reading?

// We can close a file in C++ during file handling by using the 'close()' method of the 'ifstream class' .

Eg : inputFile.close();

*/

/* Problem 2: Write to a File

Task:

Write a C++ program that writes the following lines to a file named output.txt:

bash

Copy code

Hello, world!

This is a test file. */

/* #include <iostream>

#include <fstream>

#include <string>

using namespace std;

```
int main() {
    // Define the file name
    const string fileName = "output.txt";

    // Create an ofstream object for file output
    ofstream outputFile(fileName);

    // Check if the file is open
    if (!outputFile.is_open()) {
        cout << "Could not open the file " << fileName << endl;
        return 1; // Return with error code
    }

    // Define the lines to be written to the file
    string lines[] = {"Hello, world!", "This is a test file."};

    // Write each line to the file
    for (const auto& line : lines) {
        outputFile << line << endl;
    }

    // Close the file
    outputFile.close();

    cout << "Lines written to " << fileName << endl;
```

```
    return 0; // Return with success code
} */
```

/* Questions: 2

1. How do you open a file for writing in C++?

```
  // First create a file ' Output.txt'. Then,
  // Check if the file is open
   if (!outputFile.is_open()) {
       cout << "Could not open the file " << fileName << endl;
       return 1; // Return with error code
   }

   // Define the lines to be written to the file
   string lines[] = {"Hello, world!", "This is a test file."};
```

2. **What is the purpose of the ofstream class in C++?**

// The purpose of ofstream class in C++ is used to open the file for writing .

3. How do you write a string to a file in C++?

// In C++, you can write a string to a file using the file output stream class ofstream from the <fstream> library.

Eg : // Define the lines to be written to the file

```
   string lines[] = {"Hello, world!", "This is a test file."};
```

```
    // Write each line to the file
    for (const auto& line : lines) {
        outputFile << line << endl;
    }
```

4. What is the importance of closing a file after writing to it?

// The importance of closing a file after writing is that we can prevent Data Corruptions, Mange Error Handling, help to release resources.

Eg : cout << "File written and closed successfully" << endl;

Problem 3: Append to a File

Task:

1. Write a C++ program that appends the following line to a file named log.txt:

bash

Copy code

New log entry. */

```
/* #include <iostream>
#include <fstream>
using namespace std;

int main() {
    // Create an ofstream object in append mode
    ofstream logfile;
    logfile.open("log.txt", ios::app);
```

```
    // Check if the file was opened successfully
    if (!logfile) {
        cerr << "Error opening file for appending" << endl;
        return 1;
    }

    // Append the line to the file
    logfile << "New log entry." <<endl;

    // Close the file
    logfile.close();

    // Check for errors during closing
    if (!logfile) {
        cerr << "Error occurred while closing the file" << endl;
        return 1;
    }
    cout << "Log entry appended successfully" << endl;
    return 0;
} */
```

/* Questions: 3

1. How do you open a file for appending in C++?

// In C++, we can open a file for appending using the ofstream class from the <fstream> library. The ios::app flag is used to open a file in append mode, which ensures that data is added to the end of the file without overwriting any existing content.

Eg : // Create an ofstream object in append mode

```
ofstream logfile;

logfile.open("log.txt", ios::app);
```

2. What is the difference between opening a file in write mode and append mode?

// The difference between opening a file in write mode and append mode is that in write mode we editing / writing stuff to the opened file whereas during append we are adding lines / text to already written file.

3. How do you use the ofstream class to append data to a file?

// In C++, we can use the ofstream class from the <fstream> header to write to files. To append data to a file, we need to open the file in append mode.

Eg : ofstream outfile("example.txt", ios::app); // ios::app opens the file in append mode

4. What happens if the file does not exist when you try to open it in append mode?

// When we try to open a file in append mode (ios::app) using ofstream, if the file does not exist, a new file will be created. The file creation happens automatically when we attempt to write to it.

Eg : ofstream outfile("nonexistent.txt", ios::app);

```
if (outfile.is_open()) {

    outfile << "Appending data to a new file." << endl;

    outfile.close();

} else {

    cerr << "Error opening file for appending!" << endl;

}
```

5. How can you ensure data integrity when appending to a file?

// We can ensure data integrity when appendinga file by following approaches - File Opening and Closing Check, Flush Buffer and Error Handling.

```
Eg:  if (outfile.is_open()) {
        outfile << "Data appended with integrity.\n";
        // Optional: Flush the buffer to ensure immediate write
        // outfile.flush();
        outfile.close();
        cout << "Data appended successfully!" << endl;
    } else {
        cerr << "Error opening file for appending!" << endl;
    }
```

Problem 4: Copy a File

Task:

Write a C++ program that copies the content of a file named source.txt to another file named destination.txt. */

```
/* #include <fstream>
#include <iostream>
#include <string>
using namespace std;

void copyFile(const string& sourceFileName, const string& destinationFileName) {
    ifstream sourceFile(sourceFileName);
    ofstream destinationFile(destinationFileName);

    if (sourceFile && destinationFile) {
```

```
        char ch;
        while (sourceFile.get(ch)) {
            destinationFile.put(ch);
        }
        cout << "File copied successfully!" << endl;
    } else {
        cerr << "Error: Could not open files for copying." << endl;
    }

    sourceFile.close();
    destinationFile.close();
}

int main() {
    string sourceFileName = "source.txt";
    string destinationFileName = "destination.txt";

    copyFile(sourceFileName, destinationFileName);

    return 0;
} */
```

/* Questions: 4

1. How do you read from one file and write to another file in C++?

// To read from one file and write to another file in C++, we can use ifstream for reading and ofstream for writing.

2. How can you efficiently copy the contents of a file in C++?

// Efficiently copying the contents of a file in C++ involves reading and writing in chunks rather than character by character.

This can be achieved by using buffer-based operations.

3. What are the potential errors you should handle when copying a file?

// The potential errors we should handle when copying a file is to check whether the other file exist or not to which the contents are to be copied or

- File not found: The input file may not exist, or the output file may not be able to be created.

- Permission denied: The program may not have permission to read from the input file or write to the output file.

- File is too large: The file may be too large to fit in memory, or the output file may not be able to be written to.

- I/O error: There may be an error reading from the input file or writing to the output file.

4. How do you check the end-of-file (EOF) condition when reading a file?

// 1. To check the end-of-file (EOF) condition when reading a file, you can use the following code:

```
while (inputFile.peek() != EOF) {
    // read from the file
}
```

This code checks if the next character in the file is the EOF character. If it is, the loop will exit.

5. How do you ensure both files are properly closed after the copy operation?

// 1. To ensure both files are properly closed after the copy operation, you can use the following code:

```
inputFile.close();

outputFile.close();
```

This code closes the input file and the output file using the close() method. This is important to ensure that the files are properly closed and that any system resources are released

Problem 5: Count Words in a File

Task:

Write a C++ program that reads a file named data.txt and counts the number of words in the file. */

```
/* #include <iostream>

#include <fstream>

#include <string>

#include <sstream>


using namespace std;


int main() {

    ifstream file("data.txt"); // Open the file

    string word;

    int wordCount = 0;


    if (file.is_open()) {

        while (file >> word) { // Read each word from the file

            wordCount++;
```

```
        }
        file.close(); // Close the file
        cout << "Number of words in the file: " << wordCount << endl;
    } else {
        cout << "Unable to open file." << endl;
    }

    return 0;
} */
```

/* Questions: 5

1. **How do you define a word in the context of reading from a file?**

// In the context of reading from a file, a word is typically defined as a sequence of characters delimited by whitespace (spaces, tabs, newlines, etc.). Punctuation marks and special characters are usually not considered part of a word unless explicitly defined otherwise.

2. **What functions can you use to read words from a file in C++?**

// In C++, common functions to read words from a file include:

a. operator>> with a std::ifstream object for formatted input.

b. std::getline() for reading a line at a time and then parsing words from that line using std::istringstream.

3. **How do you handle different word delimiters (spaces, newlines, etc.)?**

// By default, C++ functions like operator>> for std::ifstream and std::getline() split input based on whitespace. Custom delimiters can be specified using std::getline() with a delimiter parameter or by using other parsing techniques with std::string methods.

4. **How can you keep track of the word count while reading the file?**

// Use a simple integer variable initialized to zero ('int wordCount = 0;"). Increment this variable each time a word is successfully read from the file.

5. How do you handle large files to avoid memory issues while counting words?

// We handle large files to avoid memory issues while counting word to avoid memory issues with large files, read and count words sequentially without storing all words in memory simultaneously.

Use a loop to read and process portions of the file at a time, ensuring efficient memory usage and performance.