Name: Noor Akhtar

C++ Programming Internship

Task 3

File Compression Algorithm

Table of Contents

File Compression Algorithm:

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

void compress(const string& inputFile, const string& outputFile) {

    ifstream inFile(inputFile, ios::binary);

    ofstream outFile(outputFile, ios::binary);
    if (!inFile.is_open() || !outFile.is_open()) {
        cerr << "Error opening files!" << endl;
        return;
    }

    char currentChar;
    char previousChar = '\0';
    int count = 0;

    while (inFile.get(currentChar)) {
        if (currentChar == previousChar && count < 255) {
            ++count;
        }
        else {
            if (count > 0) {
                outFile.put(previousChar);
                outFile.put(static_cast<char>(count));
            }
            previousChar = currentChar;
            count = 1;
        }
    }

    if (count > 0) {
        outFile.put(previousChar);
        outFile.put(static_cast<char>(count));
    }

    inFile.close();
    outFile.close();
}

void decompress(const string& inputFile, const string& outputFile) {
    ifstream inFile(inputFile, ios::binary);
    ofstream outFile(outputFile, ios::binary);
    if (!inFile.is_open() || !outFile.is_open()) {
       cerr << "Error opening files!" << endl;
        return;
    }

    char currentChar;
    char countChar;

    while (inFile.get(currentChar)) {
        inFile.get(countChar);
        int count = static_cast<unsigned char>(countChar);
        for (int i = 0; i < count; ++i) {
            outFile.put(currentChar);
        }
```

```
    }

    inFile.close();
    outFile.close();
}

int main() {
    string inputFilePath, outputFilePath;
    char choice;

    cout << "Enter input file path: ";
    getline(cin, inputFilePath);

    cout << "Enter output file path: ";
    getline(cin, outputFilePath);

    cout << "'c' to compress and 'd' for decompress the file? ";
    cin >> choice;

    try {
        if (choice == 'c' || choice == 'C') {
            compress(inputFilePath, outputFilePath);
            cout << "File compressed successfully!" << endl;
        }
        else if (choice == 'd' || choice == 'D') {
            decompress(inputFilePath, outputFilePath);
            cout << "File decompressed successfully!" << endl;
        }
        else {
            cerr << "Invalid choice!" << endl;
        }
    }
    catch (const std::exception& e) {
        cerr << "An error occurred: " << e.what() << endl;
    }

    return 0;
}
```

Documentation

1. Introduction

The Simple File Compression Algorithm is designed to compress and decompress files using the Run-Length Encoding (RLE) algorithm. This tool reads a file, compresses its contents, and writes the compressed data to a new file. It can also decompress files compressed using the same algorithm.

2. Setup Instructions

2.1 File Setup

1. **Input File**:
    - Ensure you have an input file that you want to compress or decompress.
    - The input file should be a text file (.txt) or any other type of file that you want to test.
2. **Output File**:
    - Specify the output file path where the compressed or decompressed data will be stored.

2.2 Environment Setup

Ensure your development environment is set up for C++ development. The following steps provide a general guide:

1. **Install a C++ Compiler**:
    - **Linux**: Use the package manager (e.g., sudo apt-get install g++).
    - **macOS**: Use Homebrew (e.g., brew install gcc).
    - **Windows**: Download and install MinGW or use an IDE like Visual Studio.
2. **Set Up an Integrated Development Environment (IDE)**:
    - **Visual Studio Code**: A lightweight IDE with extensions for C++ development.
    - **CLion**: A powerful IDE for C++ development by JetBrains.
    - **Visual Studio**: A full-featured IDE for Windows.

3. Running the Application

3.1 Compilation and Execution

Follow these steps to compile and run the application:

1. **Compile the Application**:
    - Use a C++ compiler to compile the source code file main.cpp.
2. **Run the Application**:
    - Execute the compiled executable (file_compressor or equivalent based on compilation).

4. Functionality

4.1 Compression

The compress function compresses the input file using the Run-Length Encoding (RLE) algorithm and writes the compressed data to the output file.

4.2 Decompression

The decompress function decompresses the RLE-compressed input file and writes the original data to the output file.

5. Error Handling

The application includes mechanisms to handle errors that may occur during file operations and user input:

5.1 File Operation Errors

- **File Opening Errors**:
  - The application checks if the input and output files can be opened for reading or writing. If a file cannot be opened, an error message is displayed.

5.2 User Input Errors

- **Invalid Choices**:
  - The application verifies the user's choice and prompts them to enter a valid option if the input is incorrect.

6. How to Run the Program:

1. Create a text file named input.txt in the directory D:\DEP C++\Task-3 Compression\ with some sample content.
2. Compile and run the program.
3. When prompted, enter the full path to the input file:

   Enter input file path: D:\DEP C++\Task-3 Compression\input.txt

4. Enter the full path to the output file:

   Enter output file path: D:\DEP C++\Task-3 Compression\compressed.rle

5. Choose whether to compress or decompress by entering c or d.

By specifying the full paths including filenames, the program should be able to open the files correctly and perform the compression or decompression as needed.

7. Conclusion

The Simple File Compression Algorithm is a basic tool for compressing and decompressing files using the Run-Length Encoding (RLE) algorithm. It demonstrates key concepts in file handling and user interaction in C++. Further customization and extension of the application functionality can be done based on specific requirements.