



C++ Programming

MUHAMMAD SHAHZAIB

DEP TASK 3

Table of Contents

User Manual for File Compression and Decompression Program	3
1. Introduction	3
2. Functions.....	3
3. Instructions to Run the Program on a Windows System.....	4
4. How the Code Works	4

User Manual for File Compression and Decompression Program

1. Introduction

The File Compression and Decompression Program is a console-based C++ application designed to compress and decompress text files using the Run-Length Encoding (RLE) algorithm. This application allows users to efficiently reduce the size of text files and restore them to their original form. The application provides a simple interface to choose between compression and decompression, ensuring ease of use.

2. Functions

1. Compress File

Functionality: This function compresses the content of a specified text file using the RLE algorithm and saves the compressed data to a new file.

Steps:

- The system prompts the user to enter the name of the input file containing the original data.
- The user is asked to input the name of the output file where the compressed data will be saved.
- The system checks if the input file is already compressed.
 - If the file is already compressed, an error message is displayed.
 - If the file is not compressed, the content is compressed and saved to the output file.
- A success message is displayed once the compression is completed.

2. Decompress File

Functionality: This function decompresses the content of a specified text file that was compressed using the RLE algorithm and saves the decompressed data to a new file.

Steps:

- The system prompts the user to enter the name of the input file containing the compressed data.
 - The user is asked to input the name of the output file where the decompressed data will be saved.
- The system checks if the input file is already in its original form.
 - If the file is not compressed, an error message is displayed.
 - If the file is compressed, the content is decompressed and saved to the output file.
- A success message is displayed once the decompression is completed.

3. Exit

Functionality: This function allows the user to exit the application.

Steps:

- The system prompts the user to press 'q' to quit or any other key to continue using the application.
- If 'q' is pressed, the system terminates the application.

3. Instructions to Run the Program on a Windows System

Prerequisites

- A C++ compiler (e.g., GCC, MSVC)
- A terminal or command prompt

Steps to Compile and Run the Program

- `g++ -o FileCompression FileCompression.cpp`
- Execute the compiled program:

4. How the Code Works

1. Initialization:

- When the program starts, it enters a loop, allowing the user to choose between compression and decompression.

2. User Interaction:

- The main menu prompts the user to enter 'c' for compression or 'd' for decompression. ◦ The user inputs the names of the input and output files based on the chosen action.
- Depending on the user's choice, the corresponding function is called to compress or decompress the file.

3. File Handling:

- The `readFile` function reads the content of the specified input file.
- The `writeFile` function writes the processed data (compressed or decompressed) to the specified output file.

4. Compression and Decompression:

- The `compressRLE` function compresses the data using the RLE algorithm.
- The `decompressRLE` function decompresses the RLE-compressed data to its original form.
- The `isCompressed` function checks if a file's content is already compressed.

5. Error Handling:

- The system checks for errors such as invalid file names, already compressed or decompressed files, and file handling issues. Appropriate error messages are displayed to guide the user.

6. Exiting the Program:

- The program prompts the user to press 'q' to quit or any other key to continue using the application. If 'q' is pressed, the program terminates gracefully.

CODE:

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include <cctype>

using namespace std;

string readFile(const string &filename) {
    ifstream file(filename, ios::binary);
    if (!file) {
        cerr << "Error: Could not open the file " << filename << endl;
        exit(1);
    }

    string contents((istreambuf_iterator<char>(file)),
istreambuf_iterator<char>());
    file.close();
    return contents;
}

void writeFile(const string &filename, const string &data) {
    ofstream file(filename, ios::binary);
    if (!file) {
        cerr << "Error: Could not write to the file " << filename << endl;
        exit(1);
    }

    file << data;
    file.close();
}

string compressRLE(const string &data) {
    if (data.empty()) return "";

    ostringstream compressed;
    int n = data.length();

    for (int i = 0; i < n; ++i) {
        int count = 1;
        while (i < n - 1 && data[i] == data[i + 1]) {
            ++count;
            ++i;
        }
    }
}
```

```

    }
    compressed << data[i];
    if (count > 9) {
        while (count > 9) {
            compressed << 9;
            count -= 9;
        }
    }
    compressed << count;
}

return compressed.str();
}

string decompressRLE(const string &data) {
    ostringstream decompressed;
    int n = data.length();

    for (int i = 0; i < n; ++i) {
        char ch = data[i];
        ++i;
        int count = data[i] - '0';
        decompressed << string(count, ch);
    }

    return decompressed.str();
}

bool isCompressed(const string &data) {
    for (char ch : data) {
        if (isdigit(ch)) {
            return true;
        }
    }
    return false;
}

int main() {
    while (true) {
        string choice;
        string inputFile;
        string outputFile;

        cout << "Enter 'c' to compress or 'd' to decompress: ";
        cin >> choice;
    }
}

```

```

    if (choice != "c" && choice != "d") {
        cerr << "Invalid choice!" << endl;
        continue;
    }

    cout << "Enter input file name: ";
    cin >> inputFile;
    cout << "Enter output file name: ";
    cin >> outputFile;

    string fileContents = readFile(inputFile);
    string result;

    if (choice == "c") {
        if (isCompressed(fileContents)) {
            cerr << "Compression not possible: file is already compressed."
<< endl;
        } else {
            result = compressRLE(fileContents);
            writeFile(outputFile, result);
            cout << "File compressed successfully." << endl;
        }
    } else if (choice == "d") {
        if (!isCompressed(fileContents)) {
            cerr << "Decompression not possible: file is already in original
form." << endl;
        } else {
            result = decompressRLE(fileContents);
            writeFile(outputFile, result);
            cout << "File decompressed successfully." << endl;
        }
    }

    char cont;
    cout << "Press 'q' to quit or any other key to continue: ";
    cin >> cont;
    system("cls");
    if (cont == 'q') {
        break;
    }
}

return 0;
}

```

