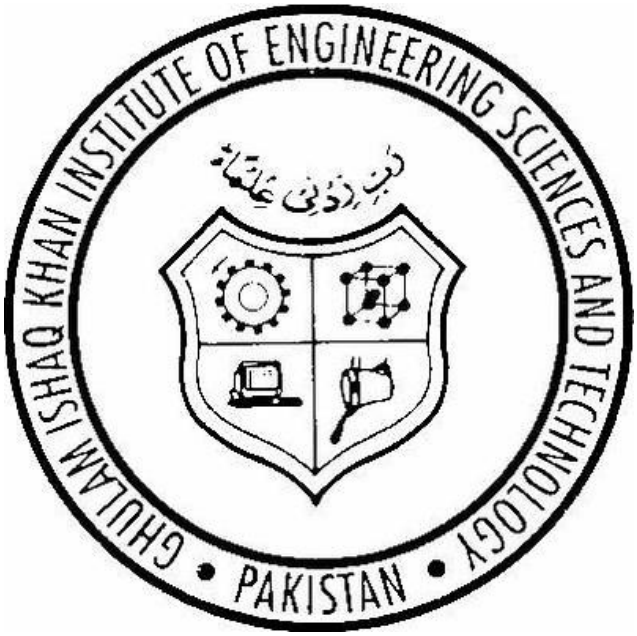
Project Report

File Encryption and Decryption Manager Using Shift Cipher



Team Members:

Aayan Rashid - 2023002

Muaaz Bin Salman – 2023338

Instructor:

Dr. Muhammad Zain Siddiqui

Course Code:

CY-211

Project Overview:

The File Encryption and Decryption Manager is a C++ application designed to handle various file operations, including creating, reading, appending, encrypting, decrypting, and deleting file content. The application uses a simple shift cipher (Caesar cipher) for encryption and decryption, with a randomly generated key for added security.

Objectives:

1. **Develop a File Manager**: Implement a system to create, read, append, encrypt, decrypt, and delete files.
2. **Implement Simple Encryption**: Utilize a shift cipher with a randomly generated key for file encryption.
3. **Provide a User-Friendly Interface**: Implement a menu-driven interface for users to interact with the application.

Key Features and Concepts:

1. **File Creation and Writing**: Create a new file and write user-provided text into it.
2. **File Reading**: Read and display the content of a file.
3. **File Appending**: Append additional text to the existing file content.
4. **File Encryption**: Encrypt the file content using a shift cipher with a randomly generated key.
5. **File Decryption**: Decrypt the file content using the provided key.
6. **File Deletion**: Remove all characters from a file.

Implementation Details:

1. **File Creation and Writing**:
    - **Function**: createAndWriteFile(const string& filename)
    - **Description**: Prompts the user to enter text and writes it to a specified file.

cpp

```
void createAndWriteFile(const string& filename) {
   ofstream outFile(filename);
   if (outFile.is_open()) {
      string text;
```

```
        cout << "\nType text to be entered: ";
        getline(cin, text);
        outFile << text;
        outFile.close();
        cout << "File created and written successfully.\n";
    } else {
        cerr << "Unable to open file for writing.\n";
    }
}
```

2. **File Reading**:
    - **Function**: readFile(const string& filename)
    - **Description**: Reads and displays the content of the specified file.

```
cpp
void readFile(const string& filename) {
    ifstream inFile(filename);
    if (inFile.is_open()) {
        string line;
        while (getline(inFile, line)) {
            cout << line << '\n';
        }
        inFile.close();
    } else {
        cerr << "Unable to open file for reading.\n";
    }
}
```

3. **File Appending**:
    - **Function**: appendToFile(const string& filename, const string& text)

- **Description**: Appends the provided text to the existing content of the file.

cpp

```
void appendToFile(const string& filename, const string& text) {
    ofstream appendFile(filename, ios::app);
    if (appendFile.is_open()) {
        appendFile << text;
        appendFile.close();
        cout << "File appended successfully.\n";
    } else {
        cerr << "Unable to open file for appending.\n";
    }
}
```

4. **File Deletion**:
    - **Function**: deleteAllCharacters(const string& filename)
    - **Description**: Deletes all content from the specified file.

cpp

```
void deleteAllCharacters(const string& filename) {
    ofstream outFile(filename, ios::trunc);
    if (outFile.is_open()) {
        outFile.close();
        cout << "All characters deleted successfully.\n";
    } else {
        cerr << "Unable to open file for deleting characters.\n";
    }
}
```

5. **Generate Random Key**:
    - **Function**: generateRandomKey()

- **Description**: Generates a random key between 1 and 26 for encryption.

cpp

```
int generateRandomKey() {
   srand(time(0));
   return rand() % 26 + 1;  // Generate a random key between 1 and 26
}
```

6. **File Encryption (Shift Cipher)**:
   - **Function**: encryptFile(const string& inputFilename, const string& outputFilename, int key)
   - **Description**: Encrypts the content of a file using a shift cipher and writes the encrypted content to a new file.

cpp

```
void encryptFile(const string& inputFilename, const string& outputFilename, int key) {
   ifstream charFile(inputFilename);
   ofstream encryptedFile(outputFilename);
   if (charFile.is_open() && encryptedFile.is_open()) {
      char ch;
      while (charFile.get(ch)) {
         if (isalpha(ch)) { // Encrypt only alphabetic characters
            if (isupper(ch)) {
               encryptedFile << char((ch - 'A' + key) % 26 + 'A');
            } else if (islower(ch)) {
               encryptedFile << char((ch - 'a' + key) % 26 + 'a');
            }
         } else {
            encryptedFile << ch; // Leave non-alphabetic characters unchanged
         }
      }
```

```
        charFile.close();
        encryptedFile.close();
        cout << "File encrypted and written to " << outputFilename << " successfully.\n";
    } else {
        cerr << "Unable to open files for encryption.\n";
    }
}
```

7. **File Decryption (Shift Cipher)**:
    - **Function**: decryptFile(const string& inputFilename, const string& outputFilename, int key)
    - **Description**: Decrypts the content of a file using a shift cipher and writes the decrypted content to a new file.

cpp

```
void decryptFile(const string& inputFilename, const string& outputFilename, int key) {
    ifstream charFile(inputFilename);
    ofstream decryptedFile(outputFilename);
    if (charFile.is_open() && decryptedFile.is_open()) {
        char ch;
        while (charFile.get(ch)) {
            if (isalpha(ch)) { // Decrypt only alphabetic characters
                if (isupper(ch)) {
                    decryptedFile << char((ch - 'A' - key + 26) % 26 + 'A');
                } else if (islower(ch)) {
                    decryptedFile << char((ch - 'a' - key + 26) % 26 + 'a');
                }
            } else {
                decryptedFile << ch; // Leave non-alphabetic characters unchanged
            }
```

```
        }
        charFile.close();
        decryptedFile.close();
        cout << "File decrypted and written to " << outputFilename << " successfully.\n";
    } else {
        cerr << "Unable to open files for decryption.\n";
    }
}
```

Example Usage:

The main function demonstrates the use of the file manager to create, read, append, encrypt, and delete file content with a random key.

cpp

```
int main() {
    int choice;
    string filename;
    string encryptedFilename;
    string decryptedFilename;
    int key;

    do {
        cout << "\nFile Encryption and Decryption Manager Menu:\n";
        cout << "1. Create and Write File\n";
        cout << "2. Read File\n";
        cout << "3. Append to File\n";
        cout << "4. Encrypt File\n";
        cout << "5. Decrypt File\n";
        cout << "6. Delete All Characters\n";
```

```
cout << "7. Exit\n";
cout << "Enter your choice: ";
cin >> choice;
cin.ignore();  // Clear newline from the buffer

switch (choice) {
    case 1: {
        cout << "Enter the filename to create and write to: ";
        getline(cin, filename);
        createAndWriteFile(filename);
        break;
    }
    case 2: {
        cout << "Enter the filename to read: ";
        getline(cin, filename);
        readFile(filename);
        break;
    }
    case 3: {
        cout << "Enter the filename to append to: ";
        getline(cin, filename);
        cout << "Enter text to append: ";
        string text;
        getline(cin, text);
        appendToFile(filename, text);
        break;
    }
```

```
case 4: {
    cout << "Enter the filename to encrypt: ";
    getline(cin, filename);
    cout << "Enter the filename to save the encrypted content: ";
    getline(cin, encryptedFilename);
    key = generateRandomKey();
    cout << "Generated Key for Encryption: " << key << endl;
    encryptFile(filename, encryptedFilename, key);
    break;
}
case 5: {
    cout << "Enter the filename to decrypt: ";
    getline(cin, filename);
    cout << "Enter the filename to save the decrypted content: ";
    getline(cin, decryptedFilename);
    cout << "Enter the key for decryption: ";
    cin >> key;
    cin.ignore();  // Clear newline from the buffer
    decryptFile(filename, decryptedFilename, key);
```