

PROJECT-PART 2

Title

Vigenere Cipher Decryption

Course

CSC 7

Section

28754

Due Date

12/08/2024

Team Name

SelfDriver

Team Member

Khadiza Akter

Contents

1. Introduction:	3
2. Project Information and details:	3
2.1: Problem Solved in This Project	3
2.2: Solutions Implemented	3
2.3: Explanation of Calculations and Algorithm	3
2.4: Program Objectives and User Interact	4
2.5: Discrete Structures Implemented in the Program	6
2.6: Program Limitations	6
2.7: Recommendations for Improvement	6
3.Pseudocode	7

1. Introduction:

This project, "Vigenère Cipher Decryption," created by Team SelfDriver and member Khadiza Akter, as part of a case study on cryptography. The primary aim is to demonstrate the use of Vigenère cipher, a classical encryption and decryption technique, to secure and decode texts. This cipher, a polyalphabetic substitution method, uses a keyword to determine shifting pattern of text. By implementing this cipher in C++, project shows practical applications of modular arithmetic and discrete structures.

The program provides a menu-driven interface that lets users encrypt or decrypt text while keeping non-alphabetic characters unchanged. It shows how cryptography works in real life and highlights important computer science skills like problem-solving, working with algorithms, and creating user-friendly designs. This documentation explains the program's purpose, how it works, and potential ideas for improvement.

2. Project Information and details:

2.1: Problem Solved in This Project

The Vigenère cipher is a polyalphabetic cipher that encrypts and decrypts text using a secret keyword, providing stronger security than simpler ciphers like the Caesar cipher. This project focuses on creating a C++ program that can encrypt plain text entered by the user into a coded message and decrypt the encrypted message back to its original form using the same keyword. This project highlights the application of discrete structures and modular arithmetic in real-world scenarios.

2.2: Solutions Implemented

This program solves the problem by first preparing the keyword so it matches the length of the text to be encrypted or decrypted. It uses a mathematical method called modular arithmetic to shift letters in the text according to the Vigenère cipher rules. The program also has a simple menu that lets users choose whether to encrypt, decrypt, or exit, making it easy to interact with.

2.3: Explanation of Calculations and Algorithm

Encryption Calculation: To encrypt the text, each character in the plaintext is shifted based on the corresponding character in the keyword. The formula used is:

$$C[i] = (P[i] - \text{offset} + (K[i] - 'A')) \% 26 + \text{offset}$$
 C[i] represents the encrypted character, P[i] is the plaintext character, and K[i] is the keyword character. The offset ensures the correct

handling of uppercase ('A') or lowercase ('a') characters. This method moves the letters based on the Vigenère cipher rules, and encrypt a plaintext into ciphertext.

Decryption Calculation: To decrypt and recover the original plaintext, the program reverses the encryption process using the formula:

$P[i] = (C[i] - \text{offset} - (K[i] - 'A') + 26) \% 26 + \text{offset}$, $P[i]$ is the recovered plaintext character, and $C[i]$ is the ciphertext character. Adding 26 ensures the result is always positive, maintaining accuracy. That's the way undo the encryption, and get the original text.

Key Calculation: Key is prepared by removing any non-alphabetic characters and changing all letters to uppercase to make sure consistency since the Vigenère cipher relies on the alphabetical order of the letters. Then, key is expanded to match the length of the input text by repeating it cyclically. For each alphabetic character in text, a corresponding key character is determined using modular arithmetic ($\text{key}[\text{keyIndex} \% \text{key.size()}]$). Non-alphabetic characters in the text are ignored, so the key stays aligned with only the letters. This makes sure the encryption and decryption use the correct shifts for each valid letter.

2.4: Program Objectives and User Interact

This program provides an encrypted and decrypted text using Vigenère cipher while ensuring a user-friendly experience. It validates user inputs to guarantee proper functionality and accuracy during encryption and decryption processes.

To achieve this, the program interacts with the user by prompting for plaintext or ciphertext along with a keyword, then prints the results of encryption or decryption. A menu system allows users to perform multiple operations.

How to work the Project(Input/Output):

When run the program, it will display a menu like as Figure 1.

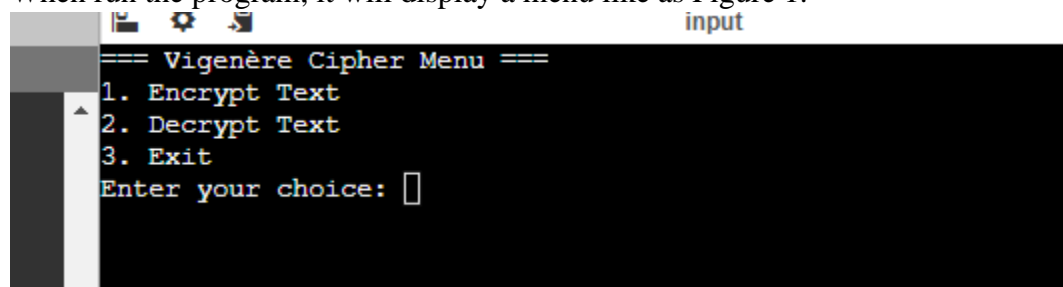
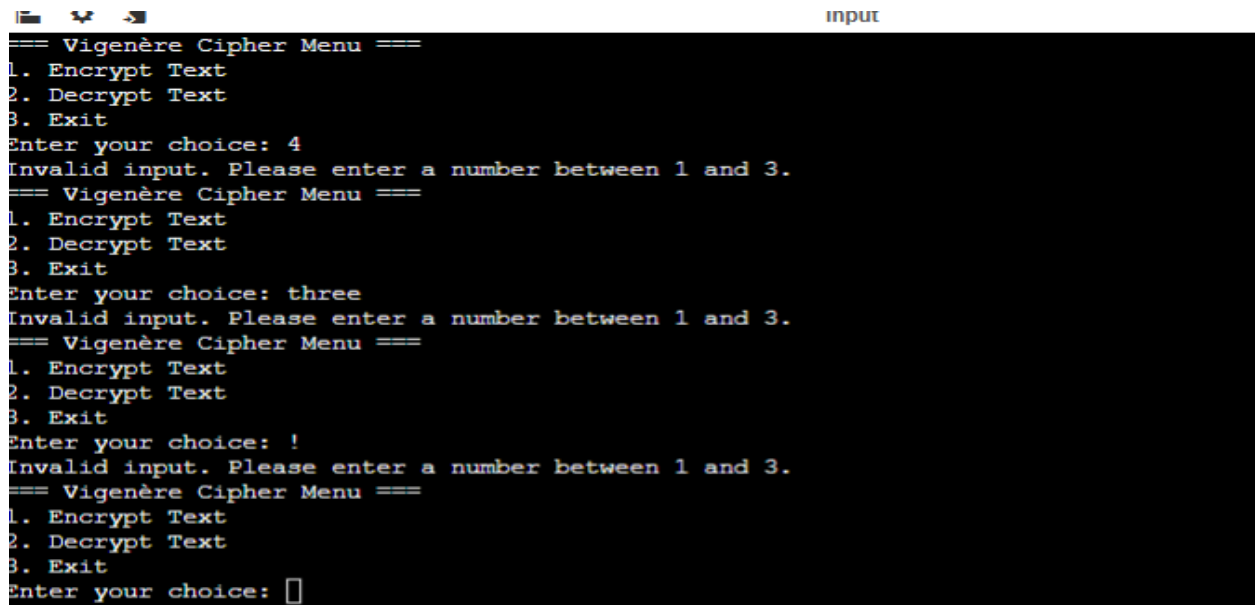


Figure 1. Project Menu

If choose something different than 1,2,3, then will be displayed as Figure 2.



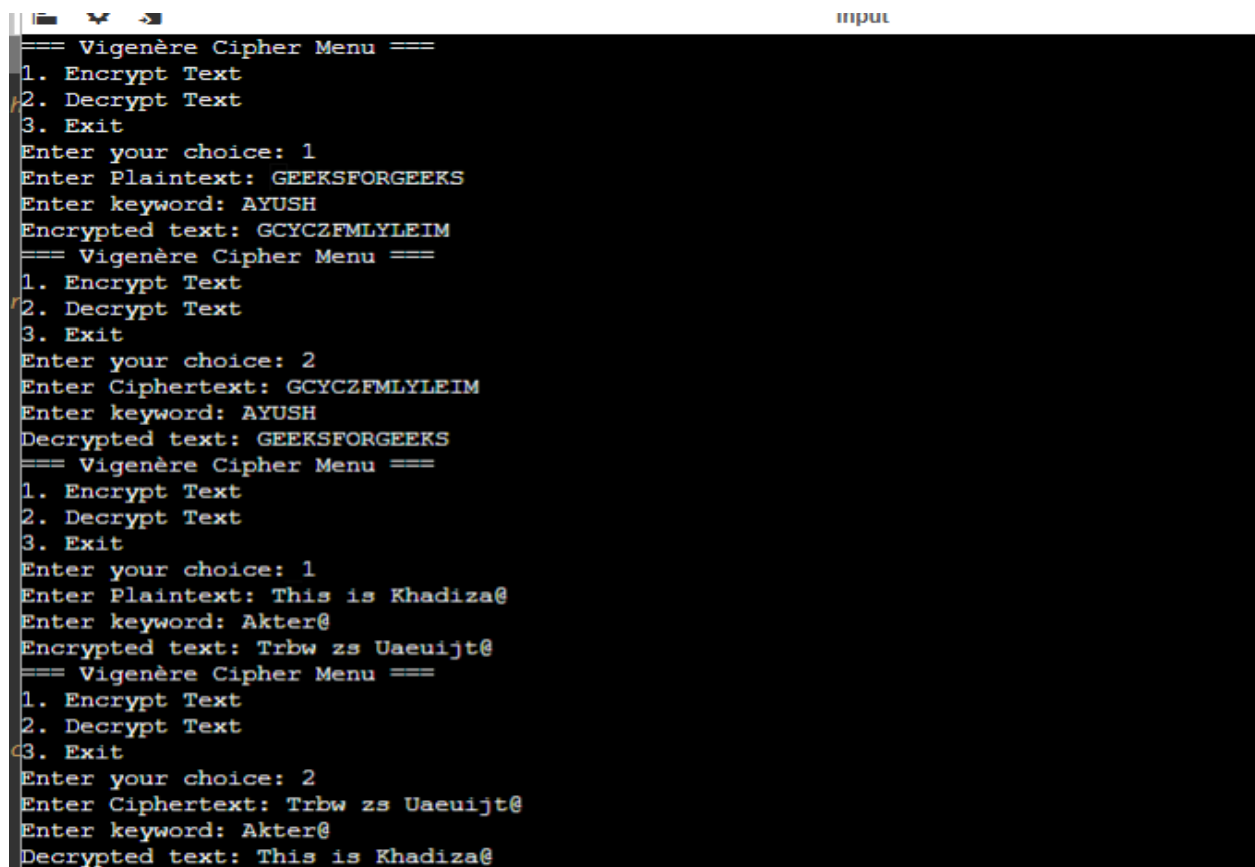
```

input
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: 4
Invalid input. Please enter a number between 1 and 3.
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: three
Invalid input. Please enter a number between 1 and 3.
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: !
Invalid input. Please enter a number between 1 and 3.
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: 

```

Figure 2

When selecting correct menu options, then display or work like figure 3



```

input
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: 1
Enter Plaintext: GEEKSFORGEEKS
Enter keyword: AYUSH
Encrypted text: GCYCZFMLEILEIM
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: 2
Enter Ciphertext: GCYCZFMLEILEIM
Enter keyword: AYUSH
Decrypted text: GEEKSFORGEEKS
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: 1
Enter Plaintext: This is Khadiza@
Enter keyword: Akter@
Encrypted text: Trbw zs Uaeuijt@
=== Vigenère Cipher Menu ===
1. Encrypt Text
2. Decrypt Text
3. Exit
Enter your choice: 2
Enter Ciphertext: Trbw zs Uaeuijt@
Enter keyword: Akter@
Decrypted text: This is Khadiza@

```

Figure 3

When selecting option 3, then get to the exit! Like as Figure 4

```
=== Vigenère Cipher Menu ===  
1. Encrypt Text  
2. Decrypt Text  
3. Exit  
Enter your choice: 3  
Exiting program. Goodbye!  
  
...Program finished with exit code 0  
Press ENTER to exit console.□
```

Figure 4

2.5: Discrete Structures Implemented in the Program

The concept of discrete structures in this Vigenère cipher program is implemented through the use of functions, modular arithmetic, and finite sets. Functions are used to the program's design like preprocessing the key, aligning the key with the text, and performing encryption and decryption, which act as mappings between plaintext, ciphertext, and keys. Also, modular arithmetic is applied to compute shifts in characters during encryption and decryption within a finite alphabet (mod 26). Besides, the program uses sets and relationships to manage alphabetic characters, ensuring each character is correctly mapped to its transformed version. This shows how concepts from discrete mathematics are used to make the cipher logical, efficient, and reversible.

2.6: Program Limitations

The program has some limitations. Non-alphabetic characters are kept in their original positions, which may not meet strict encryption standards. Keywords are converted to uppercase, which could lose case-sensitive meanings. Additionally, if the keyword is guessed or is too short, the encryption becomes easier to break.

2.7: Recommendations for Improvement

To improve the program, it could include logic to encrypt symbols and numbers, ensuring all characters are handled. Adding a feature for generating a strong, randomized keyword would enhance security. Additionally, implementing file input and output functionality would allow users to encrypt or decrypt text from files and save the results for convenience.

3.Pseudocode

START

Initialize variables for plaintext, keyword, ciphertext, and choice

Set exit flag to false

WHILE (exit is false)

 DISPLAY menu options

 PROMPT user to enter choice

 GET user choice

 IF (input is invalid OR choice is not between 1 and 3)

 CLEAR input stream

 IGNORE remaining input

 DISPLAY error message

 CONTINUE to menu

 END IF

 IGNORE leftover newline from input

 SWITCH (choice)

 CASE 1: Encryption

 DISPLAY "Enter Plaintext:"

 GET plaintext (full line input)

 DISPLAY "Enter keyword:"

 GET keyword (full line input)

 CALL encrypt(plaintext, keyword)

 STORE result in ciphertext

 DISPLAY "Encrypted text:" followed by ciphertext

 BREAK

 CASE 2: Decryption

 DISPLAY "Enter Ciphertext:"

 GET ciphertext (full line input)

```

        DISPLAY "Enter keyword:"
        GET keyword (full line input)
        CALL decrypt(ciphertext, keyword)
        STORE result in plaintext
        DISPLAY "Decrypted text:" followed by plaintext
        BREAK
    CASE 3: Exit
        SET exit to true
        DISPLAY "Exiting program. Goodbye!"
        BREAK
END SWITCH
END WHILE
END

```

FUNCTIONS:

```

FUNCTION preprocessKey(keyword)
    Initialize cleanKey to empty string
    FOR each character in keyword
        IF character is alphabetic
            ADD uppercase character to cleanKey
        END IF
    END FOR
    RETURN cleanKey
END FUNCTION

```

```

FUNCTION keyProcessor(text, keyword)
    CALL preprocessKey(keyword)
    Initialize result to empty string
    Initialize keyIndex to 0

```



```

    FOR each character in text
        IF character is alphabetic
            ADD key[keyIndex % key.length] to result
            INCREMENT keyIndex
        END IF
    ELSE
        ADD character to result
    END ELSE
END FOR
RETURN result
END FUNCTION

FUNCTION encrypt(plaintext, keyword)
    CALL keyProcessor(plaintext, keyword)
    Initialize ciphertext to empty string
    FOR each character in plaintext
        IF character is alphabetic
            SET offset to 'A' if uppercase, 'a' if lowercase
            COMPUTE new character using encryption formula
            ADD new character to ciphertext
        END IF
    ELSE
        ADD character to ciphertext
    END ELSE
END FOR
RETURN ciphertext
END FUNCTION

FUNCTION decrypt(ciphertext, keyword)

```

```
CALL keyProcessor(ciphertext, keyword)
Initialize plaintext to empty string

FOR each character in ciphertext
    IF character is alphabetic
        SET offset to 'A' if uppercase, 'a' if lowercase
        COMPUTE original character using decryption formula
        ADD original character to plaintext
    END IF
    ELSE
        ADD character to plaintext
    END ELSE
END FOR
RETURN plaintext
END FUNCTION
```