TEXT ENCRYPTION AND DECRYPTION USING CAESAR CIPHER IN C

1_INTRODUCTION :

Cryptography is the practice of securing information by transforming it into an unreadable format .
One of the simplest and most well-known encryption techniques is the caesar Cipher.
This technique works by shifting each letter in a message by a fixed number of positions in the alphabet
Although Caesar Cipher is not considered secure for modern communication, it is very useful for understanding the basic principles of encryption and decryption. This project presents an implementation of the Caesar Cipher using the C programming language. The program allows the user to encrypt or decrypt a message by choosing a shift key between 1 and 25. Both uppercase and lowercase letters are handled, while non-alphabetic characters remain unchanged.

2. PROJECT OBJECTIVES

The main objective of this project is to design and implement a simple text encryption
and decryption system using the Caesar Cipher technique in the C programming language.

The specific goals of this project are:
- To understand the basic concept of cryptography.
- To implement encryption and decryption using character shifting.
- To handle both uppercase and lowercase letters correctly.
- To validate user inputs such as the shift key.
- To provide a clear and readable output for the user.

3. METHODOLOGY

The program is based on the Caesar Cipher algorithm, which encrypts text by shifting
each letter forward or backward in the alphabet by a fixed number of positions.

First, the user chooses whether to encode or decode a message.
Then, the user enters a text message with a maximum length of 100 characters.
After that, a shift key between 1 and 25 is provided.

The program processes each character individually.
If the character is a letter, it is shifted according to the selected operation.
Non-alphabetic characters are not modified.

4. IMPLEMENTATION DETAILS

4.1 Input Handling

The program starts by asking the user to choose between encryption and decryption.
Then, the user enters a text message with a maximum length of 100 characters.

The message is stored in a character array, and a copy of the original message
is saved before any modification.
This allows the program to display both the original and processed messages.

```c
#include<stdio.h>
int main(){
    char message [100] , original[100];
    int shift , choice , i ;
    printf("choose an option :\n1.encode\n2.decode\n");
    scanf("%d",&choice);
    printf("enter the message(maximum 100 characters):\n ");
    getchar(); //to read the whole line
    scanf("%99[^\n]",message);
```

4.2 Shift Key Validation

To ensure correct encryption and decryption, the shift key must be between 1 and 25.
The program uses a loop to repeatedly ask the user for a valid shift value until
a correct input is provided.

This validation prevents invalid shifts and guarantees correct character rotation
within the alphabet.

```c
    do{
        printf("error:shift key (between 1 and 25):\n");
        scanf("%d",&shift);
        if(shift < 1 || shift > 25){
            printf("error: shift must be between 1 and 25: \n");
        }
    }
    while (shift < 1 || shift > 25 );
```

4.3 Saving the Original Message

Before modifying the message, the program creates a copy of the original text. This allows the user to view both the original and the encrypted or decrypted message after processing.

```c
    //save the original massage before changing it
    for (i = 0 ; message[i] != '\0';i++)
    {
        original[i] = message[i];
    }
    original[i]= '\0';
```

4.4 Encryption and Decryption Logic

The program processes the message character by character.
If the character is an uppercase letter, it is encrypted or decrypted using the ASCII value of 'A' as a reference.
If the character is a lowercase letter, 'a' is used as the reference.

For encryption, the shift value is added.
For decryption, the shift value is subtracted.
The modulo operation ensures that the result stays within the alphabet range.

uppercase:

```
//for uppercase
if (message[i] >= 'A' && message[i] <= 'Z')
{
    if (choice == 1)
    //forward (sumbmission encryption)
    message[i] = (message[i]-'A'+ shift) % 26 + 'A';
    else
    //decryptio(rewind)
    message[i] = (message[i] - 'A' - shift + 26) % 26 +'A' ;
}
```

lowercase:

```
//letters are small
else if(message[i] >= 'a' && message[i] <= 'z'){
    if (choice == 1)
    //encryption
    message[i] = (message[i]-'a'+shift + 26)% 26 + 'a';
    else
    //decryptin
    message[i] = (message[i]-'a'-shift + 26) % 26 + 'a';
}
```

4.5 Output Display:

After processing the message, the programme displays the original text, the selected shift key, and the final encrypted or decrypted result. This output format allows the user to clearly compare the input and output.

```
47      printf("\n original message: %s \n ",original);
48      printf("shift key :%d\n",shift);
49      if(choice == 1 )
50      printf("encoded message: %s\n",message);
51      else
52      printf("decoded message: %s\n",message);
53      return 0;
54  }
```

5. ALGORITHM EXPLANATION

1. Read the user's choice (encode or decode).
2. Read the input message.
3. Validate the shift key.
4. Store the original message.
5. Loop through each character of the message:
   - Check if it is uppercase or lowercase.
   - Apply encryption or decryption.
6. Display the original message, shift key, and final result.

6. RESULTS AND OUTPUT

After executing the program, the user can see:
- The original message.
- The selected shift key.
- The encrypted or decrypted message.

The output is displayed in a clear and readable format, allowing the user to easily
compare the original and processed text.

Exemple:

```
choose an option :
1.encode
2.decode
1
enter the message(maximum 100 charac
ters):
 Hello Meet me at 5 PM !!
error:shift key (between 1 and 25):
4

 original message: Hello Meet me at
5 PM !!
 shift key :4
encoded message: Lipps Qiix qi ex 5
TQ !!
```

```
choose an option :
1.encode
2.decode
2
enter the message(maximum 100 charac
ters):
 Lipps Qiix qi ex 5 TQ !!
error:shift key (between 1 and 25):
4

 original message: Lipps Qiix qi ex
5 TQ !!
 shift key :4
decoded message: Hello Meet me at 5
PM !!
```

7. CONCLUSION

This project demonstrates a simple and effective implementation of the Caesar Cipher
using the C programming language.
It provides a clear understanding of basic encryption and decryption techniques.

Although Caesar Cipher is not suitable for modern security systems, it is an excellent
educational tool for learning cryptography fundamentals and character manipulation in C.

https://github.com/Dima-dji/MyCipherProject.git

by:
AMARI WISSEM
DJILALI DJILALI DOUAA