THE CAESAR ENCRYPTION

Course / Subject: B. Tech [Computer Science & Engineering]

Submitted By: Adit Verma

Submitted To: Dr. Tanu Singh

Date: November 17, 2025

Abstract

This project implements a classic cryptographic technique, the **Caesar Cipher**. The program accepts a numerical key from the command line, validates it, and applies a shift to each alphabetical character in the plaintext. Non-alphabetic characters remain unchanged. The purpose of this project is to demonstrate command-line arguments, character manipulation, and ASCII operations.

Problem Definition

Supposedly, Caesar (yes, that Caesar) used to “encrypt” (i.e., conceal in a reversible way) confidential messages by shifting each letter therein by some number of places. For instance, he might write A as B, B as C, C as D, …, and, wrapping around alphabetically, Z as A. And so, to say HELLO to someone, Caesar might write IFMMP instead. Upon receiving such messages from Caesar, recipients would have to “decrypt” them by shifting letters in the opposite direction by the same number of places. The secrecy of this “cryptosystem” relied on only Caesar and the recipients knowing a secret, the number of places by which Caesar had shifted his letters (e.g., 1). Not particularly secure by modern standards, but, hey, if you’re perhaps the first in the world to do it, pretty secure! Unencrypted text is generally called *plaintext*. Encrypted text is generally called *ciphertext*. And the secret used is called a *key*.

System Design

Algorithm

Step 1: Start
Step 2: If argc != 2, print usage message and exit
Step 3: For each character in argv[1], check isdigit()
Step 4: Convert key to integer and compute k % 26
Step 5: Read plaintext using fgets()
Step 6: For each character in plaintext:

- If uppercase: apply shift formula
- If lowercase: apply shift formula
- Else: print as is
  Step 7: Print ciphertext
  Step 8: End

Shift Formulas:

- Uppercase:
  (c - 'A' + k) % 26 + 'A'
- Lowercase:
  (c - 'a' + k) % 26 + 'a'

Implementation Details

Key Features Used

- argc, argv[] for command-line arguments
- isdigit() for validation
- atoi() for string-to-integer conversion
- ASCII arithmetic for shifting characters
- fgets() for safe string input
- Loops and conditionals for processing

Important Code Snippets

- **Key Validation**

```
for (int i = 0; i < strlen(argv[1]); i++)
{
    if (!isdigit(argv[1][i]))
    {
        printf("Usage: ./caesar key\n");
        return 1;
    }
}
```

- **Shift Logic**

```
if (isupper(c))
{
    printf("%c", (c - 'A' + k) % 26 + 'A');
}
else if (islower(c))
{
    printf("%c", (c - 'a' + k) % 26 + 'a');
}
```

Testing & Results

Test Case 1

Input:
./caesar 2
Plaintext = Hello, World!

Output:
Ciphertext = Jgnnq, Yqtnf!

Test Case 2

Input:
./caesar Hey.

Output:
Usage: ./caesar key

Conclusion

The Caesar Cipher implementation successfully demonstrates basic encryption using character manipulation and ASCII values. The program is robust in argument validation and correctly handles both uppercase and lowercase characters.

References

- David J. Malan
- CS50 [2025]
- Caesar Cipher Cryptography Principles
