*1.INTRODUCTION

Introduction to the Cryptography Application Program

This C++ program implements a basic cryptographic technique known as the Caesar Cipher, a type of substitution cipher where each letter in the plaintext is 'shifted' a certain number of places down or up the alphabet. Named after Julius Caesar, who reportedly used it for military communication, this cipher is simple yet effective for learning about encryption and decryption principles.

The program features two main functionalities: encryption and decryption. Users can enter plaintext, specify a shift key (between 1 and 25), and save the encrypted text to a file. Additionally, they can decrypt previously encrypted messages using the same key.

The code is structured into a `CaesarCipher` class that encapsulates the encryption and decryption logic, ensuring clarity and modularity. Through a user-friendly command-line interface, users can select options to either encrypt or decrypt messages, or exit the application.

This implementation serves as a practical introduction to basic file handling and string manipulation in C++, making it an excellent educational tool for those interested in cryptography and programming.

*2.KEY FEATURES :-

Key Features

1. **Encryption and Decryption**:
   - The program allows users to encrypt plaintext into ciphertext using a specified key and to decrypt ciphertext back into plaintext.
2. **Key Management**:
   - Users can choose a key between 1 and 25, ensuring a manageable range for the shift, which is essential for the Caesar Cipher.
3. **File Handling**:
   - Encrypted text and the corresponding key can be saved to a file, allowing for easy storage and retrieval of encrypted messages.
4. **User-Friendly Interface**:
   - The command-line interface presents clear options, guiding users through the process of encryption and decryption.
5. **Robust Input Validation**:
   - The program checks for valid key input and provides feedback if the user enters an invalid option, enhancing user experience and preventing errors.
6. **Support for Case Sensitivity**:
   - The encryption and decryption functions handle both uppercase and lowercase letters, ensuring that the original case is preserved.

7. **Non-Alphabetic Character Handling**:
    - The program leaves non-alphabetic characters unchanged, making it versatile for various types of input.
8. **Modular Design**:
    - The use of a class (`CaesarCipher`) encapsulates the encryption logic, promoting code organization and reusability.
9. **Comprehensive Output**:
    - After encryption, the program displays the ciphertext and key in the terminal, providing immediate feedback to the user.

These features make the program a practical tool for understanding basic encryption concepts while demonstrating fundamental programming techniques in C++.

*3.HARDWARE REQUIREMENTS :-

- **Workstations/PCs**:
  - **Processor**: Intel Core i5 or i7, 2.0 GHz or higher.
  - **RAM**: 8 GB minimum (16 GB for smoother performance).
  - **Storage**: 250 GB SSD.
  - **Display**: 19-inch or larger LCD/LED screen.
  - **Operating System**: Windows 10 or 11, Linux (Ubuntu or Fedora), or macOS.
  - **Network Connectivity**: Wired (Ethernet) or Wireless (Wi-Fi) capable.
- **Laptops/Tablets** (for mobile staff):
  - **Processor**: Intel Core i5 or i7, 1.6 GHz or higher.
  - **RAM**: 8 GB minimum.
  - **Storage**: 128 GB SSD.
  - **Display**: 13-inch or larger.
  - **Operating System**: Windows, Linux, or Android/iOS (if using tablets).

*4.SOFTWARE REQUIREMENTS :-

- **Operating System**:
    - The program is compatible with any operating system that supports C++ compilers, such as:
        - Windows
        - macOS
        - Linux
- **C++ Compiler**:
    - A C++ compiler that supports C++11 or later is required. Some popular options include:
        - GCC (GNU Compiler Collection)
        - Clang
        - MSVC (Microsoft Visual C++)
- **Development Environment** (Optional):
    - While not strictly necessary, using an Integrated Development Environment (IDE) can simplify the development process. Popular IDEs include:
        - Visual Studio
        - Code::Blocks
        - CLion
        - Eclipse CDT
        - Xcode (for macOS)
- **Text Editor** (Optional):
    - If not using an IDE, any text editor can be used to write the code, such as:
        - Visual Studio Code
        - Sublime Text
        - Notepad++

    - Atom

- **Standard C++ Libraries**:
    - The program utilizes standard libraries such as `<iostream>`, `<string>`, and `<fstream>`, which are typically included with any C++ compiler.
- **File System Access**:
    - The program requires permissions to read from and write to files in the directory where it is executed.

*5.ALGORITHM :-

Algorithm

Main Function:

1. Initialize: Create a `CaesarCipher` object, and define `text`, `key`, and `choice`.

2. Display Menu: Loop until the user chooses to exit:

   - Show options: Encrypt, Decrypt, Exit.

   - Read user choice.

3. If Encrypt:

   - Get plaintext and key.

   - Validate key (1-25).

   - Call `encrypt(text, key)` and display the result.

4.If Decrypt:

   - Get ciphertext and key.

   - Validate key (1-25).

   - Call `decrypt(text, key)` and display the result.

5. If Exit: Show exit message.

6. If Invalid Choice: Show error message.

Encryption Function:

1. Initialize: `ciphertext` as an empty string.

2. For each character `c` in `plaintext`:

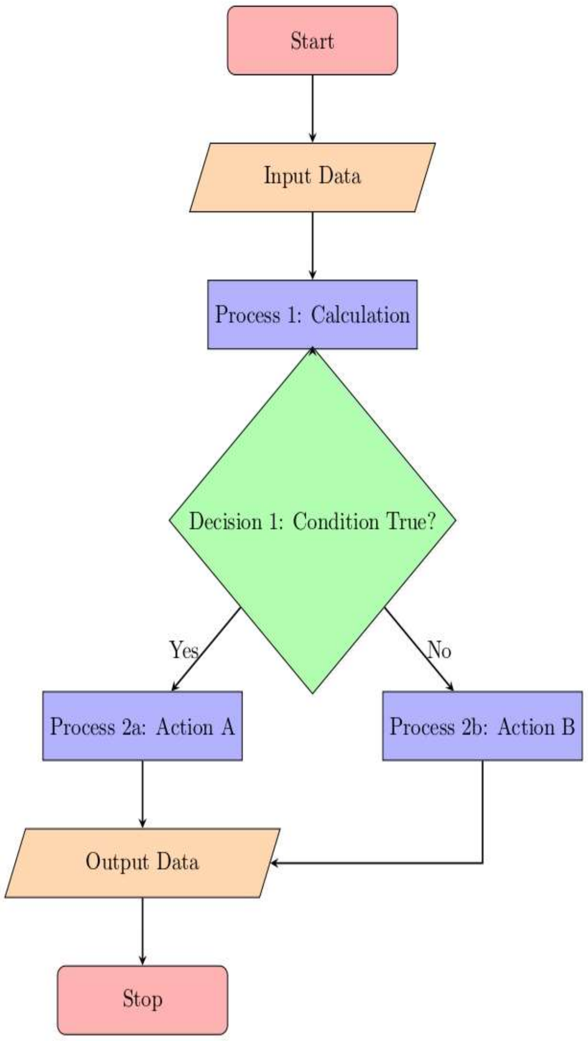
   - If `c` is uppercase: Encrypt using the formula `((c - 'A' + key) % 26) + 'A'`.

```
  - If `c` is lowercase: Encrypt using `((c - 'a' + key) % 26) + 'a'`.
  - If not alphabetic: Append `c` unchanged.
3. Return `ciphertext`.
```

Decryption Function:

```
1. Initialize: `plaintext` as an empty string.
2. For each character `c` in `ciphertext`:
  - If `c` is uppercase: Decrypt using `((c - 'A' - key + 26) % 26) + 'A'`.
  - If `c` is lowercase: Decrypt using `((c - 'a' - key + 26) % 26) + 'a'`.
  - If not alphabetic: Append `c` unchanged.
3. Return `plaintext`
```

*6.FLOWCHART :-


Start
Input Data
Process 1: Calculation
Decision 1: Condition True?
Yes
No
Process 2a: Action A
Process 2b: Action B
Output Data
Stop

*7.MAIN CODE /PROGRAM :-

```
#include <iostream>

#include <string>

#include <fstream> // For file handling


class CaesarCipher {

public:

  // Function to encrypt the text

  std::string encrypt(const std::string& plaintext, int key) {

    std::string ciphertext = "";

    for (char c : plaintext) {

      // Encrypt uppercase characters

      if (isupper(c)) {

        ciphertext += char(int(c + key - 65) % 26 + 65);

      }

      // Encrypt lowercase characters

      else if (islower(c)) {

        ciphertext += char(int(c + key - 97) % 26 + 97);

      }
```

```
        // Leave non-alphabetic characters unchanged
        else {
            ciphertext += c;
        }
    }
    return ciphertext;
}

// Function to decrypt the text
std::string decrypt(const std::string& ciphertext, int key) {
    std::string plaintext = "";
    for (char c : ciphertext) {
        // Decrypt uppercase characters
        if (isupper(c)) {
            plaintext += char(int(c - key - 65 + 26) % 26 + 65);
        }
        // Decrypt lowercase characters
        else if (islower(c)) {
            plaintext += char(int(c - key - 97 + 26) % 26 + 97);
        }
```

```
            // Leave non-alphabetic characters unchanged
            else {
                plaintext += c;
            }
        }
        return plaintext;
    }
};

int main() {
    CaesarCipher cipher;
    std::string text, fileName;
    int key, choice;

    do {
        std::cout << "\n--- Caesar Cipher Cryptography Application ---\n";
        std::cout << "1. Encrypt and Save to File\n";
        std::cout << "2. Decrypt\n";
        std::cout << "3. Exit\n";
        std::cout << "Choose an option: ";
        std::cin >> choice;

        switch (choice) {
        case 1: {
            std::cout << "Enter plaintext: ";
            std::cin.ignore();
            std::getline(std::cin, text);
            std::cout << "Enter key (1-25): ";
            std::cin >> key;

            if (key < 1 || key > 25) {
                std::cout << "Invalid key. Please enter a key between 1 and
25.\n";
            } else {
                std::string encryptedText = cipher.encrypt(text, key);

                // Ask the user for a file name to save the ciphertext and key
                std::cout << "Enter file name to save (e.g., 'output.txt'): ";
                std::cin.ignore();  // Clear the newline left by the previous
input
```

```
            std::getline(std::cin, fileName);


            // Save the ciphertext and key to the specified file
            std::ofstream outFile(fileName);
            if (outFile.is_open()) {
                outFile << "Ciphertext: " << encryptedText << "\n";
                outFile << "Key: " << key << "\n";
                outFile.close();
                std::cout << "Encrypted text and key saved to '" << fileName
<< "'.\n";


                // Display the ciphertext and key in the terminal
                std::cout << "Ciphertext: " << encryptedText << "\n";
                std::cout << "Key: " << key << "\n";
            } else {
                std::cout << "Error opening file!\n";
            }
        }
        break;
    }
```

```
        case 2: {
            std::cout << "Enter ciphertext: ";
            std::cin.ignore();
            std::getline(std::cin, text);
            std::cout << "Enter key (1-25): ";
            std::cin >> key;

            if (key < 1 || key > 25) {
                std::cout << "Invalid key. Please enter a key between 1 and
25.\n";
            } else {
                std::string decryptedText = cipher.decrypt(text, key);
                std::cout << "Decrypted text: " << decryptedText << "\n";
            }
            break;
        }

        case 3:
            std::cout << "Exiting the application.\n";
            break;
        default:
            std::cout << "Invalid choice. Please try again.\n";
        }
    } while (choice != 3);

    return 0;
}
```

*8.OUTPUT :-

*** Start the system**

```
--- Caesar Cipher Cryptography Application ---
1. Encrypt and Save to File
2. Decrypt
3. Exit
Choose an option: 1

Enter plaintext: Hello, World!
Enter key (1-25): 3
Enter file name to save (e.g., 'output.txt'): encrypted.txt
Encrypted text and key saved to 'encrypted.txt'.
Ciphertext: Khoor, Zruog!
Key: 3

--- Caesar Cipher Cryptography Application ---
1. Encrypt and Save to File
2. Decrypt
3. Exit
Choose an option: 2
```

```
Enter ciphertext: Khoor, Zruog!
Enter key (1-25): 3
Decrypted text: Hello, World!

--- Caesar Cipher Cryptography Application ---
1. Encrypt and Save to File
2. Decrypt
3. Exit
Choose an option: 3
Exiting the application.
```

*9.CONCLUSION :-

The Cryptography Application provides a straightforward yet effective implementation of a classical encryption technique, allowing users to encrypt and decrypt messages with a specified key. Through its user-friendly command-line interface, the program effectively demonstrates key concepts in cryptography, such as substitution ciphers and key management.

By incorporating file handling capabilities, users can easily save their encrypted messages and keys for future reference, enhancing the practicality of the application. The modular design, encapsulated within a dedicated class, promotes code organization and reusability, making it a valuable learning resource for both novice and experienced programmers.

Overall, this program not only serves as an educational tool for understanding basic encryption principles but also lays the groundwork for exploring more advanced cryptographic methods and techniques in the future.