**Objective:**

* To understand the concept of polymorphism in C++.

**Challenge - 1:** Encryptor App  **(**15**)**

Imagine you are working as a cybersecurity engineer at a leading technology company, tasked with developing a secure messaging feature for a new communication platform. The platform aims to provide end-to-end encryption for messages exchanged between users to ensure confidentiality and privacy.

Your task is to design and implement a program that can encrypt and decrypt messages using various encryption algorithms. The program should include the following classes:

**Encryptor:** This is an abstract base class representing a generic encryption algorithm.

* virtual String encrypt(const String& message) const = 0;
* virtual String decrypt(const String& cipher) const = 0;

**CaesarCipher:** This class inherits from the Encryptor class. The Caesar cipher shifts each letter in the message by a fixed number of positions in the alphabet.

**encrypt:**

Each letter in the message will be replaced by the letter located a fixed number of positions forward in the alphabet.

* If the shift is 3, 'A' will be replaced by 'D', 'B' will be replaced by 'E', if ‘Z’ it will round off to ‘C’ and so on.
* Non-alphabetic characters (such as punctuation and spaces) will remain unchanged.

**decrypt:**

* To decrypt the message, we'll reverse the process by replacing each letter in the ciphertext with the letter located a fixed number of positions backward in the alphabet.
* Non-alphabetic characters will remain unchanged

**VigenereCipher:** This class inherits from the Encryptor class. It is the simplified version of The Caesar cipher. It shifts each letter in the message by a fixed number of positions. It is not restricted to alphabets only

**encrypt:**

Each letter in the message will be replaced by the letter located a fixed number of positions forward in the alphabet.

* If the key is 3, '{' will be replaced by corresponding character, '.' will be replaced by corresponding character and so on.
* Alphabetic characters will remain unchanged.

**decrypt:**

* To decrypt the message, we'll reverse the process by replacing each letter in the text with the letter located a fixed number of positions backward .
* Alphabetic characters will remain unchanged.

|  |
| --- |
| **Encryptor** |
| class Encryptor  {  public:  virtual String encrypt(const String& message) const = 0;  virtual String decrypt(const String& cipher) const = 0;  }; |

|  |  |
| --- | --- |
| **CaesarCipher** | **VigenereCipher** |
| class CaesarCipher : public Encryptor  {  int shift;  char shiftChar(char c, int shift) const;  bool isAlphabet(char c) const;  public:  CaesarCipher(int shift);  String encrypt(const String& message) const override;  String decrypt(const String& cipher) const override;    }; | class VigenereCipher : public Encryptor  {  private:  int shift;  char shiftChar(char c, int shift) const;  bool isAlphabet(char c) const;  public:  VigenereCipher(int shift);  virtual String encrypt(const String& message) const override;  virtual String decrypt(const String& cipher) const override;  }; |

**Solution:**

**Encryptor.h**

#ifndef ENCRYPTOR\_H

#define ENCRYPTOR\_H

#include "String.h"

class Encryptor

{

public:

virtual String encrypt(const String& message) const = 0;

virtual String decrypt(const String& cipher) const = 0;

};

#endif // ENCRYPTOR\_H

**CaesarCipher.h**

#ifndef CAESAR\_CIPHER\_H

#define CAESAR\_CIPHER\_H

#include "String.h"

#include "Encryptor.h"

class CaesarCipher : public Encryptor

{

int shift;

char shiftChar(char c, int shift) const;

bool isAlphabet(char c) const;

public:

CaesarCipher(int shift);

String encrypt(const String& message) const override;

String decrypt(const String& cipher) const override;

};

#endif // CAESAR\_CIPHER\_H

**CaesarCipher.cpp ----- (10 Marks)**

**#include "CaesarCipher.h"**

//Private Functions:

char CaesarCipher::shiftChar(char c, int shift) const

{

char base = (c>= 'a' && c <= 'z') ? 'a' : 'A';

return ((c - base + shift + 26) % 26) + base;

}

bool CaesarCipher::isAlphabet(char c) const

{

return (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');

}

CaesarCipher::CaesarCipher(int shift) : shift(shift) {}

String CaesarCipher::encrypt(const String& message) const ----- 5

{

String result;

for (int i = 0; i < message.getLength(); i++) {

char c = message[i];

if (isAlphabet(c))

{

result += shiftChar(c, shift);

}

else

{

result += c;

}

}

return result;

}

Sample Runs:

Hello World!

Shift: 3

Result: Khoor Zruog! ---- 2 Marks

This is a secret message.

Shift: 5

Result: Ymnx nx f xjhwjy rjxxflj. ------ 3 Marks

String CaesarCipher::decrypt(const String& cipher) const ----- 5

{

String result;

for (int i = 0; i < cipher.getLength(); i++)

{

char c = cipher[i];

if (isAlphabet(c))

{

result += shiftChar(c, -shift);

}

else

{

result += c;

}

}

return result;

}

Sample Run:

Khoor Zruog!

Shift: 3

Output: Hello World! ---- 2 Marks

Ymnx nx f xjhwjy rjxxflj.

Shift: 5

Output: This is a secret message. ------ 3 Marks

**VigenereCipher.h**

#ifndef VIGENERECIPHER\_H

#define VIGENERECIPHER\_H

#include "Encryptor.h"

class VigenereCipher : public Encryptor

{

private:

int shift;

char shiftChar(char c, int shift) const;

bool isAlphabet(char c) const;

public:

VigenereCipher(int shift);

virtual String encrypt(const String& message) const override;

virtual String decrypt(const String& cipher) const override;

};

#endif // VIGENERECIPHER\_H

**VigenereCipher.cpp ----- (5 Marks)**

#include "VigenereCipher.h"

//Private Functions:

char VigenereCipher::shiftChar(char c, int shift) const

{

return c+shift;

}

bool VigenereCipher::isAlphabet(char c) const

{

return (c >= 'A' && c <= 'Z') || (c >= 'a' && c <= 'z');

}

VigenereCipher::VigenereCipher(int shift) : shift(shift) {}

String VigenereCipher::encrypt(const String& message) const ---- 2.5 Marks

{

String result;

for (int i = 0; i < message.getLength(); i++) {

char c = message[i];

if (isAlphabet(c))

{

result += c;

}

else

{

result += shiftChar(c, shift);

}

}

return result;

}

Sample Run:

....

Shift: 3

Output: 1111 ---- 1 Mark

45%

Shift: 2

Output: 67' ------ 1.5 Marks

String VigenereCipher::decrypt(const String& cipher) const ---- 2.5 Marks

{

String result;

for (int i = 0; i < cipher.getLength(); i++)

{

char c = cipher[i];

if (isAlphabet(c))

{

result += c;

}

else

{

result += shiftChar(c, -shift);

}

}

return result;

}

Sample Run:

1111

Shift: 3

Output: .... ---- 1 Mark

67'

Shift: 2

Output: 45% ------ 1.5 Marks