Team Name: Team[JAK]

Team Members: Alexis Luke, Keldin Maldonado, Julissa Mota

Course: CIS-7 (29175)

Instructor: Professor Nguyen

Date: 20 November, 2020

Vigenere Cipher

**Project Information and Details:**

1. What problems are you solving in this project?  
   For this project, we have created a Vigenere Cipher Decryption using the programming language C++. The problem we are solving is to create a program that encrypts and decrypts a user input message. Our program proposes a more advanced Vigenere Cipher, which includes not only the alphabet, but symbols and numbers as well.
2. What solutions are you implementing in the project?  
   Solutions that we have implemented into the program would be implementing the formulas for encryption and decryption to ensure the accuracy of the Vigenere Cipher chart. We have also taken upper/lower case letters, special characters, and numbers into consideration to be included in the encryption and decryption process.
3. Provide explanation of calculations and algorithm implementation.  
   The original Vigenere Cipher uses only upper case characters and no special symbols. We took the original idea of the Vigenere Cipher and expanded it to not only include, upper, lower case characters but also special characters found in the ASCII table. Expanding the Vigenere Cipher left us with 95 characters that are able to encrypt and decrypt. The original formula for Vigenere Cipher is Ei = (Pi + Ki) mod 25. Our expanded encryption and decryption formula is as follows:  
   -Encryption: Ei = (Pi + Ki) mod 95   
   - Decryption: Di = (Ei - Ki + 95) mod 95
4. What are the program objectives? Explain how your program is interacting with the user and its purpose.  
   The program’s objective is to be able to encrypt and decrypt messages using an expanded Vigenere Cipher that uses upper/lower case letters, special symbols, and numbers. The program will ask the user to enter a name for the file that will be created to input the encrypted and/or decrypted result. After, it will ask the user to input a message and a key to perform the encrypted and/or decrypted message.
5. How are discrete structures implemented in the C++ program?  
   Discrete structures are implemented into the program by using Cryptography and Modular Arithmetic. In this program, we used the ASCII table decimal values and an algebraic formula to encrypt and decrypt messages with a Vigenere Cipher. We utilized encoding, the process of putting a sequence of characters such as letters, numbers, punctuation, and certain symbols into a specialized format for storage, and the opposite process, decoding, the conversion of an encoded format back into the original sequence of characters. To achieve this we used an expanded Vigenere Cipher and modified version of the ASCII chart which reorganized the characters to start from uppercase A-Z to lowercase a-z, and other special characters.
6. What are the limitations of the program?  
   The program is limited in that the use of arrows in the menu option is allowed when you want to return to the main menu. The program should only allow you to press “enter” and “escape” simultaneously to go back. The program does not accept characters found in other languages, a good example of this limitation is in the Spanish language with the ‘ñ.’
7. Provide recommendation on improving the limitations of the program.  
   Ways to improve the limitations of the program would be to incorporate the use of arrows, from the keyboard, as invalid character inputs. Another would be that the program should detect when the keys “enter” and “escape” are specifically and simultaneously pressed by invalidating all other keys to be able to perform the same function of returning to the main menu. The language barrier between special characters of different languages and the program can be broken by allowing the program to detect and translate different languages.

**Pseudocode:**

*main function*

1. declare a vector of chars to hold the chart

2. declared 5 strings, choice, message, key, shifted message, and file name

to hold corresponding parts of encrypted message

3. declare a boolean to restart the program

4. call a function, pass the vector previously declared to it, and fill in chart

5. print out user menu and take in input from user

6. if the user does not make correct choice keep asking for correct choice

7. if the user selected option to write a file call the function to write a file

and pass the chart and the corresponding string save the output of the function

call into the boolean to restart the program previously declared

8. if the user decides to read a file call the read a file function and pass the

chart and the corresponding strings to take in and save output and save the

output of the function to the boolean restart previously declared

9. if the user selected to print out the expanded vigenere cipher call the

print vigenere function and save the output of the function into the boolean

restart program previously declared

10. if any the boolean is on from any of the previous saved function calls

keep repeating the menu

11. if the boolean is off from any of the previous saved function calls stop

repeating the menu and exit

12. display output message to user

**Pseudocode (cont):**

*write\_file function*

Set encrypt = true

Declare file stream

Declare string choice

Print Header Menu

Print Enter the name of the file (ESC + Enter to go back):

Call getline (cin, file\_name)

if (file\_name[0] == 27)

Return 1

Endif

Call file.open(file\_name)

While file is not open

Print You did not make a valid choice.

Print Please make a valid choice.

Print Choice Y/N:

Call getline(cin, choice)

While choice[0] is NOT equal to y or Y or n or N

Print You did not make a valid choice.

Print Please make a valid choice.

Print Choice Y/N:

Call getline(cin, choice)

Endwhile

if choice[0] is equal to y or Y

Print Enter the name of the file (ESC + Enter to go back):

Call getline(cin, file\_name);

If (file\_name[0] == 27)

Return 1

endif

Call file.open(file\_name)

else

Return -1

Endwhile

Print Enter Message:

Call getline(cin, message)

while (no\_illegal(message))

Print You entered an illegal key.

Print Enter message:

Call getline(cin, message)

end while

Print Enter key:

Call getline(cin, key)

while (no\_illegal(key))

Print Error: you entered an illegal key.

Print Enter key:

Call getline(cin, key);

Endwhile

Call encrypt\_decrypt(chart, message, key, shift\_message, encrypt)

Write literal string Encrypted message and variable shift\_message to a file

Close file

Print Success! The file was saved under the name: and variable file\_name

Print Return to main menu? No to exit program.

Print Choice Y/N:

Call getline(cin, choice);

While choice[0] is NOT equal to y or Y or n or N

Print You did not make a valid choice.

Print Please make a valid choice.

Print Choice Y/N:

Call getline(cin, choice)

Endwhile

If choice[0] is equal to y or Y

Return 1

Return 0

Endif

End of write\_file

**Pseudocode (cont):**

*read\_file function*

Create an integer function called "read\_file" and call on (vector<char> and reference: chart,

string message, string key, string shift\_message, string file\_name).

Declare ifstream file and string choice.

Output menu header.

Cout asking user to enter name of file or have option to press ESC + Enter to return to menu.

Getline calls on cin, file\_name.

If statement to check (file\_name[0]==7) is true, return 1.

Call on file.open(file\_name).

While loop conditioned to !file,

couts an error message and options for Y/N (Yes to retry reading file and no to return to menu).

Getline(cin, choice).

Nested while loop for choices !((choice[0] == 'y') || (choice[0] == 'Y') || (choice[0] == 'n') || (choice[0] == 'N')),

couts an error message and options Y/N.

Getline(cin, choice).

Nested if statement for (choice[0] == 'y') || (choice[0] == 'Y'),

cout to enter name of file (or ESC to go back to previous menu.

Getline(cin, file\_name).

Nested if statement for (file\_name[0] == 27),

then return to 1.

Otherwise, file.open(file\_name).

Else, return -1.

File.ignore(19); and getline(file, message);.

Cout message to enter key.

Getline(cin, key).

While loop for (no\_illegal(key)),

cout error message for illegal key, ask to enter key again.

Getline(cin, key).

encrypt\_decrypt(chart, message, key, shift\_message, encrypt)

Cout encrypted and decrypted message.

Cout to ask user to write a decrypted message to the file with choice (Y/N).

Getline(cin, choice).

While loop for (!((choice[0] == 'y') || (choice[0] == 'Y') || (choice[0] == 'n') || (choice[0] == 'N'))),

cout error message, ask user to make valid choice, options (Y/N).

Getline(cin, choice).

If statement for (choice[0] == 'y') || (choice[0] == 'Y'),

ofstream file; and file.open(file\_name);

Nested if statement for (!file),

cout error because unable to write file, make sure directory/file is not read only.

Nested else statement,

file << "Encrypted message: " << message << endl;

file << "Decrypted message: " << shift\_message;,

cout success message for file update and saved with decrypted message.

Cout to ask user to return to menu or exit program (Y/N).

Getline(cin, choice).

While loop for !((choice[0] == 'y') || (choice[0] == 'Y') || (choice[0] == 'n') || (choice[0] == 'N')),

cout error message and ask user to make valid choice. (Y/N).

Getline(cin, choice).

If statement (choice[0] == 'y') || (choice[0] == 'Y'),

return 1.

Otherwise, return 0.

**Pseudocode (cont):**

*encrypt\_decrypt function*

1. declare 3 vectors of integers to hold message, key, and shifted message  
2. check to see if the key size is greater than the message size  
3. if the key size is greater resize the key to match the message size  
4. otherwise if the the key size is smaller then subtract the size of the key from the message size and store that difference  
5. loop the number of times that was the difference from the previous calculation  
6. add the element in the key that matches the current loop cycle  
7. loop the same amount of times that is equal to the size of the message  
8. another loop inside previous loop that goes through the entire size of the chart  
9. if the ascii character stored in the message currently being look at equals to the same character currently being looked at in the chart store the number of the loop in the second loop in the vector to hold the casted message  
10. loop the same amount of times that is equal to the size of the key  
11. another loop inside previous loop that goes through the entire size of the chart  
12. if the ascii character stored in the key currently being look at equal to the same character currently being looked at in the chart store the number of the loop in the second loop in the vector to hold the casted key  
13. decide whether we are doing encryption or not  
14. if we are doing encryption then loop through the amount of times that is equal to the message size  
15. add the current element being looked at in the casted message plus the current element being looked at in the casted key then take the modulus of the size of the chart (95) and store it in the cast message vector  
16. if we are not doing decryption then loop through same amount of times that is equal to the message size  
17. subtract the current element being looked at in the casted message from the current element being looked at in the casted key then add the size of the chart size (95) and finally take the modules of the size of the chart (95) and store it in the cast message vector  
18. loop the same amount of times equal to the size of the casted shifted message  
19. another loop inside previous loop that goes through the entire size of the chart  
20. if the ascii character stored in the casted shifted message currently being look at equals to the number of times we have looped through in the second loop store the ascii character store in the chart at the position equal to the loop cycle into a string that outputs the encrypted or decrypted message

**Pseudocode (cont):**

*fill\_chart function*

Initialize for loop  
For i is equal A, i is equal to Z and iterated I by one  
 chart.push\_back(i)  
 Initialize for loop  
For i is equal a, i is equal to z and iterated I by one  
 chart.push\_back(i)  
Initialize for loop  
For i is equal space, i is equal to @ and iterated I by one  
 chart.push\_back(i)  
Initialize for loop  
For i is equal square bracket, and i is equal to backtick and iterated I by one  
 chart.push\_back(i)  
 Initialize for loop  
For i is equal curly bracket, and i is equal tilde and iterated I by one  
 chart.push\_back(i)  
End of fill\_chart

**Pseudocode (cont):**

*no\_illegal function*

Create a bool function called "no\_illegal" and call on and reference the string\_check variable.  
Use for loop to initialize i = 0, condition i < string\_check.size(), and update on i++.  
Inside the for loop, use if statement to check if the variable string\_check[i] > '~' or stromg\_checl[i] < ' ') to return true.  
Otherwise, return false.

*print\_vigenere function*

Create an integer function called"print\_vigenere"and call on (vector<char> chart).  
String choice;.  
Cout "\033[2J\033[1;1H" (to clear output window).  
Cout the top of the cipher bar.  
For statement (int i = 0; i < 189; i++),  
Cout "-". for divider.  
Cout to print out left hand bar of the cipher and first cipher line,  
for loop (int i = 0; i < chart.size(); i++),  
cout chart[i].  
For (int i = 0; i < 94; i++),  
rotate(chart.begin(), chart.begin() + 1, chart.end()),  
cout chart[0].  
For (int i = 0; i < chart.size(); i++),  
cout chart[i].  
Cout to return to main menu or exit program (Y/N).  
Getline(cin, choice).  
While loop (!((choice[0] == 'y') || (choice[0] == 'Y') || (choice[0] == 'n') || (choice[0] == 'N'))),  
cout error message and to ask to make valid choice. (Y/N).  
Getline(cin, choice).  
If statement ((choice[0] == 'y') || (choice[0] == 'Y')),  
return 1.  
Otherwise, return 0.

**Flowchart:**

Found in a separate PDF file titled: FinalProject\_Flowchart.pdf