**C++ Programming: Pointers Assignment Instructions**

**Overview**

The objective of this assignment is to demonstrate the use of pointers in a program utilizing c-strings and tokenization. The use of pointers is foundational in C++. This program provides an exercise in using pointers, passing them into functions, using them as return data types, and leveraging them in traversing arrays.

**Instructions**

Since many of you are studying the field of Cyber Security, you may find this assignment to be especially interesting as it uses a cipher that has been around for many years.

Your job is to write a program to encrypt blocks of text that can be sent over unsecured channels and decrypt blocks of text that have been received. The encryption and decryption will be accomplished using a Vigenere Cypher.

In this assignment, you will parse a string of words from a file into tokens using the **strtok\_s** command with pointers and then use the Vigenere Cypher algorithm to encrypt and decrypt the parsed words.

If you are unfamiliar with the “Vigenere Cypher,” an excellent description of it is provided in this short video: <https://www.youtube.com/watch?v=SkJcmCaHqS0>

Note: Don’t panic as you watch the video. I included it only to provide you with a visual depiction of how this cipher works. In this assignment, you will be given the exact formulas to use to encrypt and decrypt the text.

Before performing any encryption/decryption activities, the first step is to read blocks of text from a file and parse the text into individual words.

**The Program: Setting up classes and understanding tokenization**

The first step in encrypting/decrypting blocks of text is to parse the text into words before applying the encryption/decryption algorithms to them. This parsing process is called “tokenization.”

**Tokenization:**

The process of parsing sentences into words is called tokenization. To tokenize a sentence into words, use the C++ function ***strtok\_s***. Note: Do **not** try to use the C++ ***strtok*** function because it has been deemed unsafe and has therefore been deprecated.]

In your “client” code (i.e., the file that contains your main function), you will need to declare a character array that will hold **1000** characters.

Read the contents of a file into this array using the getline command. To view a discussion on how to use getline with a file object, see this link: <https://stackoverflow.com/questions/13035674/how-to-read-line-by-line-or-a-whole-text-file-at-once>

*Note: I realize that it would be easier to read the file’s contents into a string variable; that way, you wouldn’t have to tokenize the block of text. HOWEVER… one of the objectives of this assignment is to illustrate the use of tokenization of c-strings using the strtok\_s function and pointers. Therefore, you MUST use a character array to read the data from the file and tokenize it as described below to receive credit for this assignment.*

Using the character array, your client should call the function *strtok\_s*() to tokenize the character array into separate words. Here is an excellent discussion of tokenizing and the *strtok\_s* function. *Note that if you scroll down the web page to the “Example” section, you will see some code you can tweak your program.*

<https://msdn.microsoft.com/en-us/library/ftsafwz3.aspx>

**Class Construction: General Overview**

There should be two classes in your program: **Vigenere** and **MyText**.

**Vigenere** contains the encryption key and the logic for encrypting and decrypting individual words.

**MyText** contains a vector of words that have been encrypted or decrypted and the logic for calling the functions in the **Vigenere** class to encrypt or decrypt a word. The **MyText** class serves as a middle-man between your client code and the **Vigenere** class and holds the encrypted/decrypted results in a vector.

**Class Construction: Details**

**Vigenere Class**

Data Member:

* **key** - string

Functions:

* **Vigenere(string k)** – one-argument constructor
* **setKey(string k)** – data member setter
* **getKey()** – data member getter
* **toUpperCase(string k)** – function that converts the string received to upper case and then returns that string
* **encrypt(string w)** – function that encrypts the string received and then returns that string
* **decrypt(string w)** – function that decrypts the string received and then returns that string

The **Vigenere** class should store an encryption key in a data member called “**key**.”

The class should have a one-argument constructor that receives a string that represents the encryption key. The encryption key must be in all capital letters for the encryption and decryption algorithms to work. Therefore, before setting the encryption key’s value, it should first be converted entirely to upper case. Do this in your ***toUpperCase*** function.

*Note: Due to the way the Vigenere class is connected to the MyText class, you may find you have to go directly to the Vigenere class setKey() function to get the key into the class.*

To convert a string to all upper case, loop over all the characters in the string and use the ***toupper*** function (<http://www.cplusplus.com/reference/cctype/toupper/> ) to set each character in the string equal to the uppercase version of the character.

The purpose of this assignment is **not** to measure your ability to create encryption and decryption algorithms but rather to demonstrate your ability to use the strtok\_s function, along with pointers, to tokenize character arrays and to leverage object-oriented programming in the design of your program. Therefore, you **may** use the following code for the ***encrypt*** and ***decrypt*** functions in your **Vigenere** class. Cite your source.

string Vigenere::encrypt(string word)

{

string output;

for (int i = 0, j = 0; i < word.length(); ++i) {

char c = word[i];

if (c >= 'a' && c <= 'z')

c += 'A' - 'a';

else if (c < 'A' || c > 'Z')

continue;

output += (c + key[j] - 2 \* 'A') % 26 + 'A'; //added 'A' to bring it in range of ASCII alphabet [ 65-90 | A-Z ]

j = (j + 1) % key.length();

}

return output;

}

string Vigenere::decrypt(string word)

{

string output;

for (int i = 0, j = 0; i < word.length(); ++i) {

char c = word[i];

if (c >= 'a' && c <= 'z')

c += 'A' - 'a';

else if (c < 'A' || c > 'Z')

continue;

output += (c - key[j] + 26) % 26 + 'A';//added 'A' to bring it in range of ASCII alphabet [ 65-90 | A-Z ]

j = (j + 1) % key.length();

}

return output;

}

*[source:* [*https://www.tutorialspoint.com/cplusplus-program-to-implement-the-vigenere-cypher*](https://www.tutorialspoint.com/cplusplus-program-to-implement-the-vigenere-cypher) *]*

**MyText Class**

Data members:

* **words** - vector<string>
* **v** - Vigenere

Functions:

* **MyText(string *k*)** – one argument constructor  
  *k* is the key passed into the constructor, which is then passed into the Vigenere class’s setKey function.
* **encryptWord(char\* *token*)** – function (provided) that encrypts one word
* **decryptWord(char\* *token*)** – function (provided) that decrypts one word
* **FileCreate(string *fileName*)** – function that creates a file and outputs the vector of words to it.
* **DisplayText()** – function that displays to the screen the list of words in the vector.
* **WordCount()** – function that returns the number of words stored in the vector.

The **MyText** class contains a vector of strings that holds the words that were either encrypted or decrypted. The class should contain functions to encrypt a word, decrypt a word, create a file containing the vector of encrypted or decrypted words, print the vector of encrypted or decrypted words, getters and setters for the data elements wordsEncrypted and wordsDecrypted, and functions to increment these two data elements.

**Logic of the program**

**Main()**

In main(), call a function to create a menu that displays options to Encrypt a file, Decrypt a file, and Quit. The function should be called ***displayMenu*** and return an integer for the selected option. After each selection, the menu should be re-displayed until the user enters “3” for Quit. The menu should look like the one in the screenshot below:

Text

Description automatically generated

In ***main***(), you will need to create an **object** of the **MyText** class. You will use this object to call the functions in the **MyText** class that will encrypt, decrypt, save, and print the words encrypted or decrypted.

*Note: If you do not clear the vector out between encryption and decryption, you will get strange results. You can either clear the vector or create the* ***MyText*** *object before starting each action of encryption and decryption.*

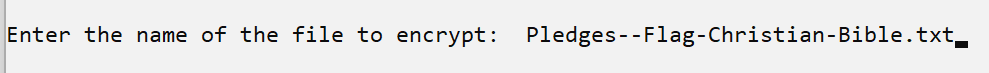
Before creating this **MyText** object for encryption, you should prompt the user to enter an encryption/decryption key. The same key must be used for decryption as was used for encryption. Otherwise, the algorithms will not produce the correct results. Once the user enters the key, pass this string into the constructor of the **MyText** class when creating your **MyText** object. This key is ultimately converted to upper case and stored in the “*key*” data member in the **Vigenere** class.



**Menu Options:**

***Option 1 – Encrypt File***

When the user selects this option, they should immediately be prompted to enter the file’s name to be encrypted: *Note that the file name can contain spaces.*



This file should already exist and contain a single paragraph of information (not to exceed 1000 characters.) Ensure appropriate error checking to ensure that the file exists and can be opened.

You may test with a file that contains something like this as its contents: Note: Capitalization and punctuation in the file will not matter as all punctuation symbols will be stripped out in the final result, and all letters will be converted to upper case.

I pledge allegiance to the Flag of the United States of America, and to the Republic for which it stands, one Nation under God, indivisible, with liberty and justice for all. I pledge allegiance to the Christian flag, and to the Savior for whose kingdom it stands. One Savior, crucified, risen and coming again, with life and liberty for all who believe. I pledge allegiance to the Bible, God’s Holy Word. I will make it a lamp unto my feet and a light unto my path. I will hide its words in my heart that I might not sin against God.

Your program should open the file and read its contents into a character array using the ***getline*** function. As soon as the file’s contents have been read into the character array, ***main***() should print the array’s contents to the screen.

Text

Description automatically generated

Next, using the ***strtok\_s*** function as described above, tokenize the entire block of text into separate words. As each word is tokenized, call the ***encryptWord*** function in the **MyText** class (using the **MyText** object to invoke the function and passing in each token as an argument ONE AT A TIME).

The ***encryptWord*** function in the **MyText** class should receive the token as a char\* data type and immediately cast it into a *string* variable. It should then pass this *string* into the ***encrypt*** function in the **Vigenere** class using the **Vigenere** object that is a data member in the **MyText** class. Once the word is encrypted, the ***encryptWord*** function in the **MyText** class should push the encrypted word onto the vector of words in the **MyText** class.

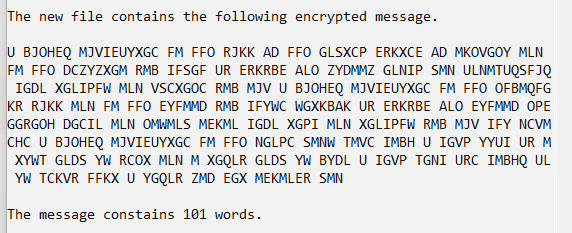
This process is repeated until the entire file’s contents have been tokenized and encrypted.

Once the entire file has been tokenized and its encrypted words have been stored in the vector of words in the **MyText** class, ***main***() should call the function in the **MyText** class to create a new file that contains the encrypted words. Therefore, the user must be prompted to enter the name of the file in which the encrypted data will be stored:

A picture containing graphical user interface

Description automatically generated

The program will then create a new file based on the name that the user entered, and that file will contain the contents of the ***words*** vector in the **MyText** class. Recall that this vector contains the words that are now encrypted. Therefore, ***main***() should call the ***DisplayText*** function in the **MyText** class to print the contents of the words vector to the screen and then call the ***FileCreate*** function in the **MyText** class to create the file and print the contents of the vector to the file, followed by the number of words in the text.



After the file is created and its contents are displayed on the screen, the Main Menu should be redisplayed.

***Option 2 – Decrypt File***

When the user selects option 2, they should immediately be prompted to enter the file’s name to be decrypted: [Note that the file name can contain spaces.]



Your program should open the file and read its contents into a character array using the ***getline*** function. As soon as the file’s contents have been read into the character array, ***main***() should print the array’s contents to the screen.

A picture containing scatter chart

Description automatically generated

Next, using the ***strtok\_s*** function as described above, tokenize the entire block of text into separate words. As each word is tokenized, call the ***decryptWord*** function in the **MyText** class (using the **MyText** object to invoke the function and passing in each token as an argument ONE AT A TIME).

The ***decryptWord*** function in the **MyText** class should receive the token as a char\* data type and immediately cast it into a *string* variable. It should then pass this *string* into the ***decrypt*** function in the **Vigenere** class using the **Vigenere** object that is a data member in the **MyText** class. Once the word is decrypted, the ***decryptWord*** function in the **MyText** class should push the decrypted word onto the vector of words in the **MyText** class.

This process is repeated until the entire file’s contents have been tokenized and decrypted.

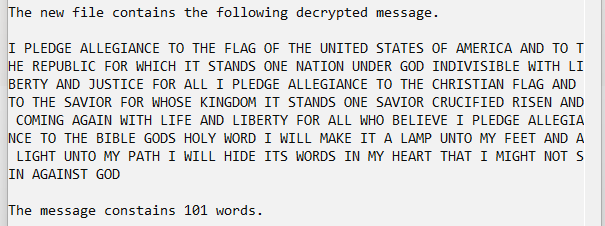
Once the entire file has been tokenized and its decrypted words have been stored in the vector of words in the **MyText** class, ***main***() should call the function in the **MyText** class to create a new file that contains the decrypted words. Therefore, the user must next be prompted to enter the file name in which the decrypted data will be stored.

A picture containing text

Description automatically generated

The program will then create a new file based on the name that the user entered, and that file will contain the contents of the ***words*** vector in the **MyText** class. Recall that this vector contains the words that are now decrypted. Therefore, ***main***() should call the ***DisplayText*** function in the **MyText** class to print the contents of the words vector to the screen and then call the ***FileCreate*** function in the **MyText** class to create the file and print the contents of the vector to the file.

After the file is created, its contents and word count are displayed on the screen. Below is an example of what it could look like.



Then the Main Menu should be displayed.

**Notice that there is a LOT of similar functionality between menu options 1 and 2. The only differences are in the functions called in the MyText class to either encrypt or decrypt the words. This overlap of functionality represents an excellent opportunity to simplify your code into functions--thus making it more modular and generic.**

Here is a link that provides some additional information on the Vigenere Cypher if you are interested in learning more about it:

<https://brilliant.org/wiki/vigenere-cipher/>

# Video: Cryptanalysis of Vigenere cipher: not just how, but why it works…

<https://www.youtube.com/watch?v=QgHnr8-h0xI>

Here is a video on how to break the Vigenere Cypher if you don’t know the key:

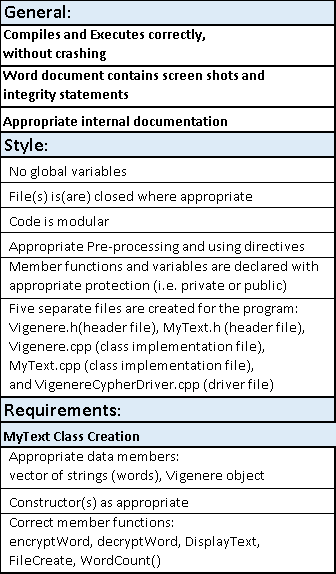
<https://www.youtube.com/watch?v=LaWp_Kq0cKs&t=0s>

**Deliverables**:

* Complete the programming assignment described above and submit your completed assignment in accordance with the lab submission policies.

Integrity statements document w/ screenshot(s) of the application’s execution.

Below is a checklist you will want to go through before submitting the lab.

Table

Description automatically generated