Unraveling the Secret Message

Due: Sep 16, 2024 11:59 PM CST

Documentation

Before you start your assignment, you will need to add documentation similar to what we demonstrated in class.

```
// Programmer: San Yeung
// Date: 9/4/1002
// File: fahr2celc.cpp
// Assignment: HW2
// Purpose: this file contains the main function of the
//   program which will input Fahrenheit temps from the user,
//   then convert and output Celsius.
```

Background

Today's the big day! As a talented young wizard, you can't wait to find out if you have been accepted into the prestigious Oscar Academy of Magic after going through a LONG application process. The acceptance letter has been delivered through the chimney of your house, but unfortunately it is encrypted using the Caesar Cipher. Since the letter holds important information about your acceptance and you don't currently own a Caesar decipher device, you decided to create a C++ program to help you break the news!

Implement a program that can encode and decode messages using the Caesar cipher. The Caesar cipher is a simple encryption technique that replaces each letter in the original message with a letter shifted a certain number of positions down the alphabet.

Requirements:

To decrypt the letter, you need to write a C++ program to implement the Caesar Cipher encryption and decryption algorithm. The program should be able to encrypt and decrypt messages using a shift value specified by the user. For this assignment, assume that the shift value supplied by the user is always greater or equal to zero. Proper use of decision branching (if-else), and loops (for, while) is expected.

1. First, the program should decrypt the acceptance decision letter and output the message. Use a shift value of 20 for this. To help you get started, recall that the ASCII values for 'a' and 'z' are 97 and 122 respectively, and the ASCII values for 'A' and 'Z' are 65 and 90 respectively. Here's the blob of encryption to be included in your program:

```
Bottub! Alyyncham zlig nby nqchefcha niqylm iz nby Imwul Uwuxygs iz
Guacw! Qy'ly vovvfcha ipyl qcnb yrwcnygyhn ni fyn sio ehiq nbun sio, sym,
sio bupy yhwbuhnyx om uff uhx mywolyx u mjin un iol yhwbuhncha yhwfupy!
Vyzily sio juwe siol vuam uhx mjyff viiem, lygygvyl ni vlcha hin domn nby
womniguls ycabn quhxm zil iol mjyffvchxcha ijyhcha qyye wylygihcym von
ufmi u nlomns wugyf! Nbymy guahczcwyhn wlyunolym ulyh'n domn zil mbiq,
nbys'ly nby jylzywn jufm zil qctulxm qbcttcha uvion ih nblcffcha
uxpyhnolym. Xoy ni u jozzs jozz iz chzfuncih nbun'm vyqcnwbyx iol
ywihigs, nby nocncih zyy cm jyaayx un u guacwuf $50,000 jyl mygymnyl. Qy
wuh'n qucn ni ombyl sio nblioab iol uhwcyhn, cps-wipylyx ulwbqusm uhx
chni nby byuln iz guacw cnmyfz. Ayn lyuxs ni mncl jincihm, wbulg gsnbcwuf
vyumnm, uhx jylbujm, wbuhay nby wiolmy iz nby wimgim!
```

Remember to figure out a way to ensure that the shift value stays within the range of the alphabet. For example, if your shift value is 2, and you are shifting a character 'Z' (ASCII value = 90), then the result should be 'B' (ASCII value = 66) and not '\' (ASCII value = 92). You can verify the accuracy of your encoding/decoding algorithm at this location.

2. The program should then prompt the user for the message to be encoded or decoded (stored as a string variable) and the number of positions to shift.
    - The shift value has to be between 1 and 40 (inclusive) for the Caesar cipher.
    - If the shift value entered by the user is not within the range, the program should print an error message and prompt the user to enter a valid shift value again.
    - When using cin and then immediately using getline(), don't forget to use cin.ignore() to ignore the remaining characters in the input buffer, to avoid any unexpected behavior caused by this conflicting pair of usage.

3. The program should continuously ask the user if they want to continue the program. If the user chooses to continue, the program should repeat the encryption and decryption process, allowing the user to input new messages and shift values. If the user chooses to quit, the program should terminate.

Implementation Details:

Inclusion of Number Encryption

For this assignment, your Caesar cipher also needs to be compatible with number encryption. The range for the numbers is (0-9). Similarly, each number also has a corresponding ASCII value in the ASCII table. For example, the ASCII value for 1 is 49 and for 3 is 51. *Use the same shift value from user input for both alphabet letters and numbers.*

Special Rules

When encoding the user's message, your program will also include a special rule for vowels ('a', 'e', 'i', 'o', 'u'/ 'A', 'E', 'I', 'O', 'U'). When a vowel is encountered in the message, instead of shifting by the user-specified value, the program will shift the vowel by the user-specified value plus a random integer (uses 27 for this assignment). You must also consider how to wrap around the alphabet correctly when applying this rule.

In addition to the above rule, for letters within the range of 'e'/ 'E' and 'n'/ 'N' (exclusive), apply the user-specified shift value if the letter's ASCII value is even. If the ASCII value is odd, the letter remains unchanged. This rule **overrides** the vowel enhancement rule.

Also, please note that the rules described above do not apply to the decryption of the acceptance letter.

For decrypting the user's message, **ignore** the additional rules applied during encryption. Simply shift back using the shift value provided by the user to decrypt the message.

Special Note on Decryption Ambiguity: In this assignment, the additional encryption rules can make the decryption process ambiguous. This means that the output might not precisely match the original input. This ambiguity is expected and acceptable for the purpose of this assignment. Not all encryption methods are perfectly reversible, especially the ones with added complexity that result in *lossy encryption*, where some information might not be perfectly recoverable.

Bonus:

1. *Shift Values Adaptability* [5 points]
    - Enhance your C++ program to seamlessly accommodate both positive and negative shift values.
        - A positive shift value implies moving to the right in the alphabet, while a negative value indicates a shift to the left.
        - For instance, with a shift value of -1, 'b' would be encrypted to 'a', and 'a' would loop to 'z'.
        - Ensure that the program can handle edge cases, such as when the alphabet loops from 'a' to 'z' or vice versa.
2. *Encoding Transformation States* [10 points]
    - This bonus challenge invites you to make our current cipher to be *lossless* and *fully recoverable* by encoding the state of your encryption transformations directly within your encrypted message.
    - Task Details:
        - For each additional rule applied during the encryption process, you are to mark its application with a unique symbol (encoding) within the encrypted message.
        - **The Vowel Enhancement Rule:** Each time a vowel is shifted, append an "*" immediately after the encrypted vowel.
        - **The Letter Range Rule:** Similarly, for letters within the specified range that are encrypted with their ASCII values being even, append

an “**” immediately after the encrypted letter. For letters within the range with odd ASCII values, append an “***” immediately after the encrypted letter.
        - **Example:** If “Cake” is encrypted with a shift value of 2, the encrypted message will look something like “Ed*k***h*”.
    - It is safe to assume that the original user message used for encryption does not contain the asterisk (“*”) symbol, as their presence could interfere with our specific encoding system.
    - Don’t forget that the decryption process must be redesigned to interpret these markers accurately.

Sample Output:

```
Welcome to the Oscar Academy of Magic Caesar Cipher Program!

Unraveling the secret acceptance decision letter from the Oscar Academy of
Magic…

Using a shift value of 20…

…

Drumroll please…

The acceptance letter says:

??? (write your own C++ program to figure it out!)



Now, let's start encoding and decoding some messages!

Do you want to encrypt or decrypt a message? (e/d) e
Enter the message to be encoded: Hello World
Enter the shift value (1-40): 2
Encrypted message: Jhnnr Yrtnf

Would you like to continue the program? (y/n): y

Do you want to encrypt or decrypt a message? (e/d) e
Enter the message to be encoded: The quick brown fox jumps over the lazy dog
```

```
Enter the shift value (1-40): 2
Encrypted message: Vjh sxiek dtryp hrz lxmru rxht vjh ndba frg

Do you want to encrypt or decrypt a message? (e/d) d
Enter the message to be decoded: Vjh sxiek dtryp hrz lxmru rxht vjh ndba frg
Enter the shift value (0-40): 2
Decrypted message: Thf qvgci brpwn fpx jvkps pvfr thf lbzy dpe

Would you like to continue the program? (y/n): y

Do you want to encrypt or decrypt a message? (e/d) d
Enter the message to be decoded: Wkh txlfn eurzq ira mxpsv ryhu wkh odcb grj
Enter the shift value (0-40): 3
Decrypted message: The quick brown fox jumps over the lazy dog

Would you like to continue the program? (y/n): n

Thanks for using the Oscar Academy of Magic Caesar Cipher Program!

Keep making magic!
```

Notes:

- You MUST NOT use functions for this assignment except for the main function.
- You are welcome to create as many variables as you wish, as long as you use them well.
- DO include input/range validations as long as they are appropriate. While input data type validation is not required for this assignment, it is good practice to validate the data values for range. For example, you should ensure that the shift value input by the user stays within the range of 1 to 40, inclusive. Another example would be to make sure that the user entered a correct value when prompting if they'd like to continue running or exiting out of the program.
- The program should be able to handle both upper-case and lower-case letters. In order to distinguish between upper and lower case alphabets, you can use the built-in functions isupper() and islower(). These functions return true if the character is an uppercase or lowercase alphabet, respectively, and false otherwise. For example:

```
char letter = 'A';
if (isupper(letter)) {
    cout << "The letter is uppercase" << endl;
} else {
    cout << "The letter is not uppercase" << endl;
}
```

  - You'll need to #include the <cctype> library to access the isupper() and islower() functions in C++.

- The program should be able to handle messages that contain punctuation, numbers, and spaces.
  - By including the <cctype> standard library, the isalpha() function then can be used to check if a character in the message is an alphabet or not. This function returns true if the character is an alphabet and false otherwise. This can be useful in handling messages that contain punctuation, numbers, and space
- Here's an example of how to iterate through a string in C++ using a for loop:

```
#include <iostream>
#include <string>
using namespace std;

int main() {
    string str = "Iterating through a string";
    for (int i = 0; i < str.length(); i++) {
        cout << str[i] << " ";
    }
    cout << endl;
    return 0;
}
```

- Test the program with different inputs. This will help you ensure that the program is functioning as expected and will also help you identify any bugs or errors.
- Read the error messages carefully if the program is not working as expected. Understanding the error messages will help you to identify and fix the problem.

- Don’t forget to use an adequate amount of comments to explain your code and make it easy to understand.
- You must use proper indentation (two whitespaces) to make the code more readable and easy to understand.
- Finally, don't hesitate to ask for help from the instructor or TA if you are having trouble with the assignment.

Grading Criteria:

1. Clarity and Understandability (20%)
    - Code should be easy to follow.
    - Variables are appropriately named and clearly defined.
    - Logical flow is evident.
2. Completeness (20%)
    - All components of the assignment are addressed.
    - Edge cases or unique scenarios are considered and handled appropriately.
3. Optimization and Efficiency (20%)
    - The solution takes advantage of efficient methods and avoids unnecessary steps.
4. Syntax and Structure (15%)
    - Code follows a consistent format and adheres to C++ standards.
    - Proper indentation is used to enhance code readability.
    - Use of appropriate C++ conventions and constructs.
5. Problem-Solving and Logic (15%)
    - The solution effectively addresses the problem.
    - Logic used in the code is sound and free of major errors.
6. Documentation and Comments (5%)
    - Important steps are commented for clarity.
    - Complex sections of code have accompanying explanations.
7. Creativity and Originality (5%)
    - Students showcase original thinking.
    - Unique and efficient solutions to problems are encouraged.