Computer Engineering Department

MSKU

CENG 2010
HOME PRACTICE
CRYPTOGRAPHY LIBRARY CREATION

In this HOA, you are asked to create a simple cryptography library. As a task, you are going to implement Caesar encryption method.

In this library, you will have two header and two implementation files:

- Basics.h
- CipherMethods.h
- CipherMethods.c
- main.c

In Basics header file, include any necessary C standard library header files, e.g. stdio.h and stdlib.h and mathematical constants as required. For instance, include string and ctype header files since our tasks require some functions belonging to these headers.

The constant macros defined in Basics.h are:

- NUMBER_OF_LETTERS with the value of 26
- NUMBER_OF_DIGITS with the value of 10
- CIPHER_SPECIAL_CHARS with the value of " .!?,;:'()[]{}`"
- CAESAR_SHIFT with the value of 3

In CipherMethods header file, include Basics.h. This file is going to be used to declare any necessary cipher function declaration. Now, declare 2 functions, namely:

- encryptCaesar
- decryptCaesar

encryptCaesar takes a constant char pointer as parameter (the text to be encrypted) and returns a char pointer (cipher text).

decryptCaesar takes a constant char pointer as parameter (the cipher text) and returns a char pointer (original text).

In CipherMethods implementation file, include CipherMethods.h. You will use this file to implement the body of each declared cipher function. Now check the second page to see the implementation details of Caesar Cipher.

Encryption:

Caesar encryption method is one of the simplest encryption techniques which shifts the text in the given shift amount **s** which is defined as CAESAR_SHIFT in our program. If **E** is the “encrypted” text and **O** is the "original" text, then, for instance, with s = 3:

O: I AM CAESAR
E: L DP FDHVDU

This means that $E_{i=}O_i + s$ for the ith element of encryped and original texts. However, you need to be careful as you handle the turning points (after ‘z’ for lowercase letters, after ‘Z’ for uppercase letters, and after ‘9’ for digits).

$E_i = O_i$ for special chars. Do not shift them. Any case other than letters, digits and special characters will be considered as undefined.

Decryption:

$O_i = E_i - s$ for letters and digits

$O_i = E_i$ for special chars

Again be careful in turning points.

In main file, include CipherMethods.h. This is the file where our main function resides. At the top of your code, define a tracing macro: **TRACE_ORG_DEC** which is used for the conditional compilation of decrypted to original text match check.

Please see the sample output first to get the idea of the functionality of the main function. Please note that the screen is cleared in the second image in order to prevent anybody to see the original text without asking again:

```
** You are about to enter a very secret cryptography service called CENG 2002 C-
Secret Coded System **

Enter your text:

Genius without education is like a silver in the mine
```

Figure 1: Welcome screen and text entry

```
In order to see the encrypted message, enter your passcode:
ceng2002
Cipher: Jhqlxv zlwkrwx hgxfdwlrq lv olnh vloyhu lq wkh plqh

Would you like to convert the cipher to original text (Y/N)?
Y
Genius without education is like a silver in the mine

Original text and the decrypeted text match.

Process returned 0 (0x0)    execution time : 75.533  s

Press any key to continue.
```

Figure 2: The second screen after clearing the first one. The cipher is shown here.

Start your main function with the following declarations:

- char text[500];
- char passcode[9] = "ceng2002";

After variable declarations, print the welcome text to the screen. Ask the user to enter the text and scan it to text variable. Then call the encryption function and keep the encrypted text in some other char array.

Now clear the screen for security using:

system ("CLS");

Continue with asking the passcode. Please be careful that if the passcode is entered wrong for 3 times in order, program exits with code 1 and prints an error message (see Fig. 3).

```
In order to see the encrypted message, enter your passcode:
cen22
Wrong passcode. Enter again:
ceng222
Wrong passcode. Enter again:
ceng200
Number of allowed attempts has been reached without successful entry!
Your IP has been blacklisted by us. Good luck!!
```

Figure 3: Wrong passcode entry.

As shown in Fig. 2, print the cipher and ask the user about the conversion. If the answer is y or Y, decrypt the cipher text and print it.

If **TRACE_ORG_DEC** is defined, compare our original text with the decrypted one.

If they match, print

    Original text and the decrypted text match.

otherwise print:

    There is something wrong. Original text and the decrypted text do not match.

and end the conditional compilation.

DISCUSSION: Is breaking the Caesar's Cipher complicated? For 26 number of letters, how many possible decrypted strings you create?

Have you ever tried other cryptography techniques for encryption, e.g. ***RSA*** (***Rivest–Shamir–Adleman***)? Search how RSA works.