Universitatea din Craiova
Facultatea de Automatică,Calculatoare şi Electronică

03 June 2018

Project : Algorithm Design
Title : Crypting and Decrypting of a message
Teachers : Bădică Costin & Ionuţ Murareţu & Becheru Alex
Student : Lazăr Leonard Ruben
Section : Calculatoare Română
Year I
Group : 1.2B

Contents

1 Problem Statement

1.1 Title

Crypting and Decrypting of a message

1.2 Description

My project refers to an application that will contain an algorithm for encrypting the relevant characters appear in words with a count that represents a prime number, and an algorithm that will decrypt the encrypted messages in the same manner as the above. Any string that will appear in a nonprim number is irelevant and ignored by the decrypt function and will be generated randomly to the crypt function.

Dynamic allocation is the automatic allocation of memory in C/C++, Unlike declarations, which load data onto the programs data segment, dynamic allocation creates new usable space on the programs STACK (an area of RAM specifically allocated to that program).For this reason the matrix was dynamically allocated.

2 Pseudocode

2.1 prime no

```
$START$
$INT\ i, number$
if ($number == 0 || number == 1$) then
    $not\ a\ prime\ number\ > \ END$
end if
for ($i = 2$ TO number/2+1 do
    if ($number \% i == 0$) then
        $not\ a\ prime\ number\ > \ END$
    end if
end for
 $prime\ number\ > \ END$
```

2.2 generate letter

```
$START$
$int\ pick, var$
$pick = rand()\%2$
if ($pick == 0$) then
    $var = rand()\%26$
    $var = var\ + 97$
    $*letter = var$
else
    $var = rand()\%26$
    $var = var + 65$
    $*letter = var$
end if
$END$
```

2.3 generate prime

```
$START$
$INT\ i, pick$
$INT\ stock[6] = 2, 3, 5, 7, 11, 13$
$pick = rand()\%6$
$*n = stock[pick]$
$END$
```

2.4 generate non prime

```
$START$
$INT\ pick = 2$
while ($prime_n o(pick)$) do
```

```
    $pick = rand()\%14$
end while
$*n = pick$
$END$
```

2.5 generate print array

```
$START$
$INT\ i$
for $(i = 0\ TO\ array.size)$ do
    $print("\%c", array.text[i])$
end for
$END$
```

2.6 Message Decrypt

```
$START$
$char\ letter =' +'$
$char\ last_letter =' *'$
$INT\ i, apparitions = 1$
$d\_array.size = 0$
$d\_array.text = (char*)malloc(d\_array.size * sizeof(char))$
for $(i = 0\ TO\ array.size + 2)$ do
    if $(letter! = last\_letter)$ then
        if $(prime\_no(apparitions))$ then
            $d\_array.size + +$
            $d\_array.text = (char*)realloc(d\_array.text, d\_array.size*sizeof(char))$
            $d\_array.text[d\_array.size - 1] = last\_letter$
        end if
        $apparitions = 1$
        $last_letter = letter$
        $letter = array.text[i]$
    end if
end for
$END$
```

2.7 Message Crypt

```
$START$
$charletter$
$INT\ i,\ j = 0,\ letter\_num,\ flag$
for $(i = 0\ TO\_array.size$ do
    $flag = rand()\%2$
    while $(flag)$ do
        $generate\_letter(\&letter)$
        $generate\_non\_prime(\&letter\_num)$
        $c_array.size+ = letter_num$
        $c\_array.text = (char*)realloc(c\_array.text, c\_array.size)$
        for ($j$ $TO$ $c\_array.size$) do
            $c_array.text[j] = letter$
        end for
        $flag = rand()\%2$
    end while
    $generate_prime(\&letter\_num)$
    $c\_array.size+ = letter\_num$
    $c\_array.text = (char*)realloc(c\_array.text, c\_array.size)$
    for ($j$ $TO$ $c\_array.size$) do
        $c_array.text[j] = array.text[i]$
    end for
end for
$END$
```

3 Application Design

3.1 Main

This module contains only the main function. The user is required to introduce an input from the keyboard. First of all,we will need to count if a string represents a prime number or not,for this issue we will implement prime_no function that will check if a number is prime, we will implement generate_letter function that will generate a random letter ,we will implement generate_prime function that will generate a non prime number from 1 to 14 because otherwise the output couldn't have been human readable , we will implement generate_prime function that will generate a prime number from 2 to 13 and we will declare an array used to store the first 6 prime numbers , only those were used to ease the readability of crypted messages output. For this program I used the crypt and the decrypt function. For the encryption function I used the parameter c_array.size where I will stock the message that will be encrypted,with the help of the variables letter and last_letter I will store the current letter and the precedent letter,also I count the number of apparitions of a letter with the variable apparitions,also I used a flag variable like this : when (flag == 1) then there will be a non prime array of letters,and in the end I used the structure variable c_array witch will store the encrypted message. For the decryption function I used the parameter d_array.size where I will stock the message that will be decrypted,with the help of the variables letter and last_letter I will store the current letter and the precedent letter,also I count the number of apparitions of a letter with the variable apparitions,and in the end I used the structure variable d_array witch will store the decrypted message. There is a chance that in between every good sequence of letters a bad non-readable sequence will be added as it was specified in the problem requirements (a prime number of letters sequence), this chance is determined by the function rand() and the good and bad sequences will be generated by the following functions : generate_prime() , generate_non_prime() , generate_letter(). The main function contains only functions calls.

3.2 Input Data

In my implementation I decided to require the used to introduce from the keyboard an already encrypted message following the problem requirements (only prime rows of the same letter will be counted as a good set of data and decrypted with that specific letter) This message will be decrypted and crypted again in order to show that both of the functions work as intended .

3.3 Output Data

The output data of my implementation will be shown on the run screen to ease its readability for the user. The output will be represented by the decrypted message initially and then by the respective crypted message as I already specified in the input subsection.

3.4 Functions

The functions used in the program are presented in **Section 2**,where is presented and pseudocode.

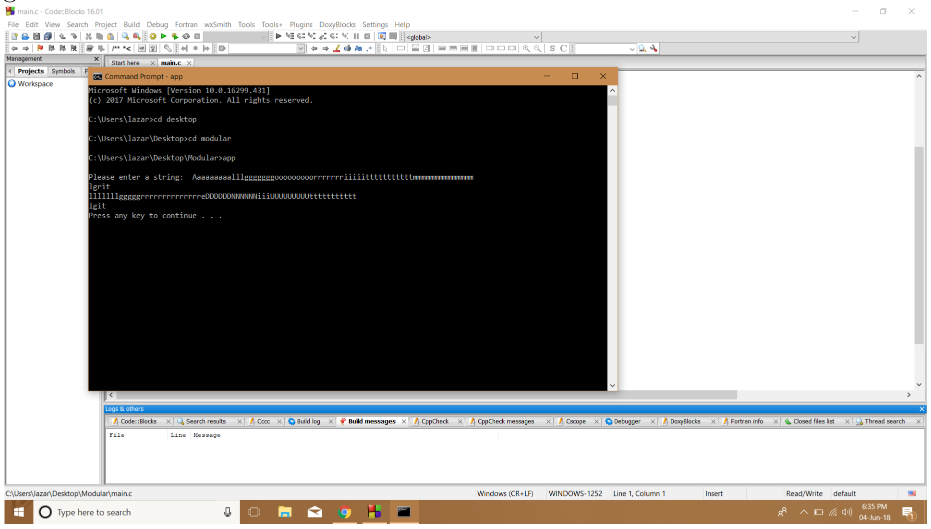
4 Source Code

My project is called "Crypting and Decrypting of a message" . The source code is created in programming language standard C99,that is compiled in one compiler. The compiler is in GNU GCC Compiler with help program Code Blocks 16.01

5 Experiments and results

5.1 GNU GCC Compiler

For the GCC compiler, will present a case. The method I used, running the program in the next subsection.



```
Microsoft Windows [Version 10.0.16299.431]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\lazar>cd desktop

C:\Users\lazar\Desktop>cd modular

C:\Users\lazar\Desktop\Modular>app

Please enter a string:  Aaaaaaaaalllgggggggooooooooorrrrrriiiiitttttttttttmmmmmmmmmmmmmm
lgrit
llllllgggggrrrrrrrrrrrreDDDDDDNNNNNNiiiUUUUUUUUUtttttttttt
lgit
Press any key to continue . . .
```

6 Conclusion

A problem about crypting and decrypting a text is an interesting approach over concepts used in real life in domains such as data bases or even banking , I believe that understanding this concept even at a base level as it was implemented in this module is very important in order to be able to develop and work in a more complex work environment that requires data security that can only be obtained through encryption.

7 References

Book:
Name : Totul despre C si C++
Year of publication :2005
Publishing :Teora
Author :Dr. Kris Jamsa Lars Klander
Web references:
1.$http : //www.geeksforgeeks.org$
2.$https : //www.sharelatex.com/learn/Main_Page$
Article:
1.$https : //en.wikipedia.org/wiki/Encryption$