Encryption and Decryption Of a Message

June 3, 2018

Student: Calcan Traian Marius

Computers and Information Technology
(Romanian Language)

Group C.R.1.1 A

1st Year

Problem statement

Two brothers, Alex and John, decide to exchange encrypted messages. In order to encrypt the messages they decided to move their keyboard letters to the right in a circular fashion for each row, with a given displacement. They have identical keyboards, the first row of letters is: qwertyuiop, the second is: asdfghjkl and the third is: zxcvbnm. Space and punctuation characters remain in the same position. In order to communicate the displacement used by each of them upon writing their message, they decided that the first letter of each message should indicate the position of letter a after the displacement, followed by a white space. For example the encrypted message s sms str zrtr should be decrypted as a ana are mere. The displacement is 1 as the letter s is the next letter from a. The letter m becomes z while e becomes r, due to the circular displacement on every row. White an application that implements an algorithm for encrypting a message and one for decrypting the message.

Pseudocode of Encrypt application

First, we will use an adjacency matrix for the directed graph representation, being an easier method.
We will use functions for each operation required by the application, with integer parameters, which will be called by the main program.
Here are the important functions employed by the program:

```
search_row(char key ,char row[11])
1.      if key = ' '
1.1      return 12
2.      for i <- 1 to lenght row
2.1     if key = row[i]
2.1.1       return i
```

```
int identify (char key ,char row_1[] ,char row_2[] ,char row_3[] ,int
    *row_number)
1. j <- search_row(key,row_1)
2. *row_number <- 1
3. if j = 11
3.1.    j <- search_row(key,row_2)
3.1.1   *row_number <- 2
4. if j = 11
4.1.    j <- search_row(key,row_3)
4.1.1   *row_number <- 3
```

```
encrypt_row(char row[] ,int position)
1. cpy[] <- row[]
2. aux <- position
3. for i <- 0 to lenght row - position
3.1     row[i] <- cpy[aux]
3.2     aux <- aux + 1
4. j <- 0
5. for i <- lenght row - position to lenght row
5.1     row[i] <- cpy[j]
5.2     j <- j + 1
6. row[i + 1] <- cpy [0]
```

```
print_encrypted(char message [] , char row_1[] ,char row_2[] ,char
    row_3[] ,int message_id [] ,int row_number [])
1. for i <- 0 to lenght message
1.1     if message_id = 12
1.1.1       message[i] <- ' '
1.1.2       continue
1.2     if row_number[i] = 1
1.2.1       message[i] <- row_1[message_id[i]]
1.2.2       continue
```

```
1.3     if row_number[i] = 2
1.3.1       message[i] <- row_2[message_id[i]]
1.3.2       continue
1.4     if row_number[i] = 3
1.4.1       message[i] <- row_3[message_id[i]]
1.4.2       continue
```

```
main ()
1.row_1[] <- "qwertyuiop"
2.row_2[] <- "asdfghjkl"
3.row_3[] <- "zxcvbnm"
4. print "Introdu mesajul :";
5. read message
6. if message[0] != ' ' and message[1] = ' '
6.1     key <- message[0]
7. else
7.1.    print "prima litera din mesaj trebuie urmata de un spatiu "
7.2.    exit
6. position <- identify(key ,row_1 ,row_2 ,row_3 ,&row_number[0])
7. for i <- 0 to lenght message
7.1.   message_id[i] <- identify (message [i] ,row_1 ,row_2 ,row_3
    ,&row_number[i])
8. if position = 8
8.1.   row_1="opqwertyui"
8.2.   row_2="lasdfghjk"
8.3.   row_3="xcvbnmz"
9. else
9.1.   encrypt_row(row_1,position)
9.2.   encrypt_row(row_2,position)
9.3.   encrypt_row(row_3,position)
10. print_encrypted(message,row_1,row_2,row_3,message_id,row_number)
11. print message
12. exit
```

Application design

The library contains the header ***functions.h*** which has all the function prototypes to compute the required operations. These are all of them:
—int search_row(char key,char row[11])
—int identify(char key ,char row_1[] ,char row_2[] ,char row_3[], int *row_number)
—void encrypt_row(char row[], int position)
—void print_encrypted(char message[] ,char row_1[] ,char row_2[] ,char row_3[] ,int message_id[] ,int row_number[])

The source file ***functions.c*** has all the function implementations.
Function search_row(char key ,char row[11]) includes some steps, like:
1)We associate value 12 for ' '
2) We search the key in row and return the iterator(i)
3) If key is not found in row[] the function return 11 ***2***.

Function identify (char key ,char row_1[] ,char row_2[] ,char row_3[] ,int*row_number) search the key in all 3 rows and give us the number of the row(row_number) for the key.

Function encrypt_row(char row[] ,int position) encrypt the row following those steps:
1)we copy the row[] string in a new string cpy[]

2)we copy the in a new variable called aux

3)in the first for we use i= 0 to length of row[] - position to give row[i] the value of cpy[aux] with aux= position to the end of the row

4)we use a second iterator(j) = 0

5)in the second for we use i = length of the row - position to give row[i] the value of cpy[j] with j = 0 to length of the row - position

6)we give the last element of row[] the value of the first element of cpy[]

Function print_encrypted(char message [] , char row_1[] ,char row_2[] ,char row_3[] ,int message_id [] ,int row_number []) rewrite message[] using the encrypted rows and the id of the elements of the message(message_id) .

The final source file, ***main.c*** itself, uses all of the functions in five main modules that form the "menu screen".

The first module ask us to enter the message that will be encrypted.

The main have the following steps:
1)We initiate the first row of the keyboard

2)We initiate the second row of the keyboard

3)We initiate the third row of the keyboard

4)We get the message that will be encrypted

5)We get the key that we will use to encrypt the message

6)We search the position of the key on the second row

7)We search the message id and store it in message_id[]

8)We check is the position of the key is 8 , if the position of the key is 8 we need change the rows with the encrypted rows . This encryption algorithm will not work for key = 8 because the third row have only 7 elements.
9)If key is not 8 we encrypt the rows

10)We encrypt the message

11)We print the message

Source Code

```
//----------------------------search_row.c-------------------------------
#include "functions.h"

 int search_row(char key,char row[11]){
     int i;

     if (key==' '){
        return 12;

     }
     for(i=0;i<strlen(row);i++){
        if (key == row[i]){
            return i;
        }
     }

     return 11 ;
}
```

```
//----------------------------identify.c-------------------------------
#include "functions.h"

int identify( char key ,char row_1[] ,char row_2[] ,char row_3[], int
     *row_number){
    int j;

    j = search_row(key,row_1);
        *row_number=1;

    if(j == 11){
        j=search_row(key,row_2);
        *row_number=2;
    }

    if(j == 11){
       j = search_row(key,row_3);
       *row_number=3;
       }

    return j;

}
```

```
//----------------------------encrypt_row.c-------------------------------
#include "functions.h"
```

```
void encrypt_row(char row[], int position){
       int i;
       int aux;
       int j;
       char cpy[12];

       strcpy(cpy,row);

       aux = position;

           for(i = 0 ; i < strlen(row) - position; i++){
               row[i] = cpy[aux];
               aux++;
           }

           j = 0;

           for(i = strlen(row) - position; i < strlen(row); i++){
               row[i] = cpy[j];
               j++;
           }

           row[i + 1] = cpy[0];
       }
```

```
//-----------------------------print_encrypted.c-------------------------------
#include "functions.h"

void print_encrypted(char message[] ,char row_1[] ,char row_2[] ,char
    row_3[] ,int message_id[] ,int row_number[]){
   int i;
   for(i = 0; i < strlen(message); i++){
       if (message_id[i] == 12){
           message[i] = ' ';
           row_number[i] = 0;
           continue;
       }

       if(row_number[i] == 1){
           message[i] = row_1[message_id[i]];
           continue;
       }

       if(row_number[i] == 2){
           message[i] = row_2[message_id[i]];
           continue;
       }
```

```
        if(row_number[i] == 3){
            message[i] = row_3[message_id[i]];
            continue;
        }
    }
}
```

```
//---------------------------------main.c-------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "functions.h"


int main()
{
    char message[100];
    char row_1[12] = "qwertyuiop";
    char row_2[11] = "asdfghjkl";
    char row_3[9] = "zxcvbnm";
    char key;

    int i ;
    int position;
    int message_id[100];
    int row_number[100];


    printf("introdu mesajul :");
    gets(message);
    if(message[0] != ' ' && message[1]==' '){
        key = message[0];
    }else{
        printf("Prima litera din mesaj trebuie urmata de un spatiu\ns");
               system("pause");
        return 0;
    }

    position = identify(key ,row_1 ,row_2 ,row_3 ,&row_number[0]);

    for(i = 0; i < strlen(message) ; i++){
        message_id[i] =
            identify(message[i],row_1,row_2,row_3,&row_number[i]);
    }

    if(position==8){
        strcpy(row_1,"opqwertyui");
        strcpy(row_2,"lasdfghjk");
        strcpy(row_3,"xcvbnmz");
```

```
    }else{
    encrypt_row(row_1,position);
    encrypt_row(row_2,position);
    encrypt_row(row_3,position);
        }


    print_encrypted(message,row_1,row_2,row_3,message_id,row_number);
    printf("Mesaj encriptat :%s",message);
        printf("\n");
        system("pause");

    return 0;
}
```

Experiments and results

```
introdu mesajul :s ana are mere
Mesaj encriptat :d sms str zrtr
```

Experiments and results

```
introdu mesajul :d ana are mere
Mesaj encriptat :g dzd dyt xtyt
```

Experiments and results

```
introdu mesajul :l ana are mere verzi
Mesaj encryptat :k lml lwq zqwq bqwxy
```

Pseudocode of Decrypt application

First, we will use an adjacency matrix for the directed graph representation, being an easier method.
We will use functions for each operation required by the application, with integer parameters, which will be called by the main program.
Here are the important functions employed by the program:

```
search_row(char key ,char row[11])
1.      if key = ' '
1.1      return 12
2.      for i <- 1 to lenght row
2.1     if key = row[i]
2.1.1       return i
```

```
int identify (char key ,char row_1[] ,char row_2[] ,char row_3[] ,int
    *row_number)
1. j <- search_row(key,row_1)
2. *row_number <- 1
3. if j = 11
3.1.    j <- search_row(key,row_2)
3.1.1   *row_number <- 2
4. if j = 11
4.1.    j <- search_row(key,row_2)
4.1.1   *row_number <- 2
```

```
print_decrypted(char message[] ,char row_1[] ,char row_2[] ,char row_3[]
    ,int message_id[] ,int row_number[],int position)
1. for i <- 0 to lenght message
1.1     if message_id = 12
1.1.1.      message[i] <- ' '
1.1.2.      row_number[i] <- 0
1.1.2       continue
1.2     if row_number[i] = 1
1.2.1       message[i] <- row_1[message_id[i]]
1.2.2       continue
1.3     if row_number[i] = 2
1.3.1       message[i] <- row_2[message_id[i]]
1.3.2       continue
1.4     if row_number[i] = 3
1.4.1       if position = 8 and message[i] = 'z'
1.4.1.1         message[i]='m'
1.4.1.2         continue
1.4.2       message[i] <- row_3[message_id[i]]
1.4.3       continue
```

```
main ()
```

```
1.row_1[] <- "qwertyuiop"
2.row_2[] <- "asdfghjkl"
3.row_3[] <- "zxcvbnm"
4. print "Introdu mesajul :";
5. read message
6. if message[0] != ' ' and message[1] = ' '
6.1     key <- message[0]
7. else
7.1.    print "prima litera din mesaj trebuie urmata de un spatiu "
7.2.    exit
8. position <- identify(key ,row_1 ,row_2 ,row_3 ,&row_number[0])
9. for i <- 0 to lenght message
9.1.   message_id[i] <- identify (message [i] ,row_1 ,row_2 ,row_3
    ,&row_number[i])
9.2.   if message_id[i] != 12
9.2.1.     message_id[i] <- message_id[i] - position
9.2.2      if message_id < 0
9.2.2.1.       if row_number = 1
9.2.2.1.1.        message_id[i] = message_id[i] + 10
9.2.2.2.       if row_number = 2
9.2.2.2.1.        message_id[i] = message_id[i] + 9
9.2.2.3.       if row_number = 3
9.2.2.3.1.        message_id[i] = message_id[i] + 7

10. print_decrypted(message,row_1,row_2,row_3,message_id,row_number)
11. print message
12. exit
```

Application design

The functions search_row(char key ,char row[11]) and identify (char key ,char row_1[] ,char row_2[] ,char row_3[] ,int*row_number) are the same ones I used in the Encrypt application .

The function print_decrypted(char message[] ,char row_1[] ,char row_2[] ,char row_3[] ,int message_id[] ,int row_number[],int position) is based on print_encrypted(char message[] ,char row_1[] ,char row_2[] ,char row_3[] ,int message_id[] ,int row_number[]) the only difference is the instruction 1.4.1(pseudocode) when we check if position of key is 8 and i the message[i] is 'z' ,if it is true message[i] = 'm'.

The final source file, ***main.c*** itself, uses all of the functions in five main modules that form the "menu screen".

The main() function is based on the main() function from Encrypt application. The functions are the same until the instruction 9.2.(pseudocode).The differences consist in the following steps:
1)If the message is not 12 (12 is the ID for the char ' ') we get the ID of the encrypted (message_id[i] - position).

2)If the ID of the encrypted message ¡ 0 we change the message_id[i] with the message_id[i] + the number of elements of the row (10 for the first row ,9 for the second row ,7 for the last row).

3)We decrypt the message.

4)We print the message

Source Code

```
//----------------------------search_row.c-------------------------------
#include "functions.h"

 int search_row(char key,char row[11]){
     int i;

     if (key==' '){
        return 12;

     }
     for(i=0;i<strlen(row);i++){
        if (key == row[i]){
            return i;
        }
     }

     return 11 ;
}
```

```
//----------------------------identify.c-------------------------------
#include "functions.h"

int identify( char key ,char row_1[] ,char row_2[] ,char row_3[], int
     *row_number){
    int j;

    j = search_row(key,row_1);
        *row_number=1;

    if(j == 11){
        j=search_row(key,row_2);
        *row_number=2;
    }

    if(j == 11){
       j = search_row(key,row_3);
       *row_number=3;
       }

    return j;

}
```

```
//----------------------------print_decrypted.c-------------------------------
#include "functions.h"
```

```
void print_decrypted(char message[] ,char row_1[] ,char row_2[] ,char
    row_3[] ,int message_id[] ,int row_number[],int position){
    int i;
    for(i = 0; i < strlen(message); i++){
        if (message_id[i] == 12){
            message[i] = ' ';
            row_number[i] = 0;
            continue;
        }

        if(row_number[i] == 1){
            message[i] = row_1[message_id[i]];
            continue;
        }

        if(row_number[i] == 2){
            message[i] = row_2[message_id[i]];
            continue;
        }

        if(row_number[i] == 3){
                        if(position == 8 && message[i]=='z'){
                                message[i]='m';
                                continue;
                        }
            message[i] = row_3[message_id[i]];
            continue;
        }
    }
}
```

```
//--------------------------------main.c--------------------------------
#include <stdio.h>
#include <stdlib.h>
#include <strings.h>
#include "functions.h"

int main()
{
    char message[100];
    char row_1[12] = "qwertyuiop";
    char row_2[11] = "asdfghjkl";
    char row_3[9] = "zxcvbnm";
    char key;

    int i ;
    int position;
    int message_id[100];
```

```
    int row_number[100];


    printf("introdu mesajul :");
    gets(message);
    if(message[0] != ' ' && message[1]==' '){
        key = message[0];
    }else{
        printf("Prima litera din mesaj trebuie urmata de un spatiu");
               system("pause");
        return 0;
    }

    position = identify(key ,row_1 ,row_2 ,row_3 ,&row_number[0]);

    for(i = 0; i < strlen(message) ; i++){
        message_id[i] =
            identify(message[i],row_1,row_2,row_3,&row_number[i]);
        if(message_id[i] != 12){
            message_id[i]=message_id[i]-position;
            if(message_id[i] < 0){
                if(row_number[i]==1){
                    message_id[i] = message_id[i] + 10;
                }

                if(row_number[i]==2){
                    message_id[i] = message_id[i] + 9;
                }

                if(row_number[i]==3){
                    message_id[i] = message_id[i] + 7;
                }

            }
        }
    }


    print_decrypted(message,row_1,row_2,row_3,message_id,row_number,position);
        printf("%s",message);

        printf("\n");
        system("pause");
    return 0;
}
```

Experiments and results

```
introdu mesajul :s sms str zrtr
Mesaj decriptat :a ana are mere
```

Experiments and results

```
introdu mesajul :d dzd dyt xty
Mesaj decriptat :a ana are mere
```

Experiments and results

```
introdu mesajul :l lml lwq zqwq bqwxy
Mesaj decriptat :a ana are mere verzi
```

Conclusions

Working on this project, I have gained a better understanding of the strings in c ,it helped me to learn how to use them

References

1)http://www.sharelatex.com