**Theory:**

Playfair cipher program in c is a manual symmetrical encryption technique that is used to encrypt or encode a message. As this technique uses the same key for encryption and decryption, this technique falls under the category of symmetrical encryption technique. It was the first literal digraph substitution cipher. Playfair Cipher was the successor of traditional monoalphabetic ciphers that were able to encrypt only a single letter. In this algorithm, two letters are paired and then encrypted.

**Encryption Algorithm:**

The Playfair Cipher technique is done in two steps which are given below. In this technique, some parameters are given and some parameters we need to find, in this algorithm. For example- The given parameters are the message which is the "plain-text" that we need to convert into the cipher-text and the second thing is the "keyword" that we need to write into the matrix structure of 5\*5.

**1) Generate the Key Square (5×5):**

First, a square box that consists of a 5\*5 grid is made in which we need to write the keyword. This text box works as the key using which we will encrypt the plain text. In this key, the alphabet is filled in each box. All the alphabets should be unique. Generally, the letter "J" is removed from the key because the key square can contain only 25 letters. But if there is "J" already present in plain text then it is replaced by "I".

**2) Algorithm to Encrypt the Plain Text:**

The plain text that we need to encrypt, is split into pairs of letters (digraphs). After splitting the plain text, if the number of letters is odd, then a 'Z' is added to the last letter. Also, the letters inside a pair should be unique. All the unique letters from left to right of Keyword are filled in the grid and then the rest grids are filled with the remaining letters of alphabets. In some cases, when there is the repetition of a letter in the text then a pair is not made with the same text. A bogus letter is added to each repeated text.

**Decryption Algorithm:**

**1) Generate the key Square (5×5) at the Receiver’s End:**

The square will be of order 5\*5 in which the letters of a key are entered one by one. All the alphabet present inside the text box should be unique. Since there are 26 letters in the alphabet, and the square has only 25 grids, the letters "J" is generally removed from the box. If the letter "J" is present in the cipher text, then "J' is replaced with "I" in the square box. Initially, all the unique letters from left to right of the key are filled in the boxes. After that, all the remaining letters of the alphabet are filled in order of appearing.

**2) Algorithm to Decrypt the Ciphertext:**

The ciphertext is split into two letters known as "Digraphs". The number of characters of a ciphertext is always even. After that, the key-matric is traversed by pair, and the corresponding encipher is searched for that pair.

**Programming Language: C**

**IDE: VS-Code**

**Code :**

#include <stdio.h> //header file

#include <stdlib.h> //header file

#include <string.h> //header file

#define SIZE 30

// this function will convert the string to lowercase

void toLowerCase(char plain[], int ps)

{

int i;

for (i = 0; i < ps; i++) {

if (plain[i] > 64 && plain[i] < 91)

plain[i] += 32;

}

}

// this function will remove all the spaces

int removeSpaces(char\* plain, int ps)

{

int i, count = 0;

for (i = 0; i < ps; i++)

if (plain[i] != ' ')

plain[count++] = plain[i];

plain[count] = '\0';

return count;

}

// this function will generate the 5x5 grid square

void generateKeyTable(char key[], int ks, char keyT[5][5])

{

int i, j, k, flag = 0, \*dicty;

// character hashmap of 26 character that will

// store count of the alphabet.

dicty = (int\*)calloc(26, sizeof(int));

for (i = 0; i < ks; i++) {

if (key[i] != 'j')

dicty[key[i] - 97] = 2;

}

dicty['j' - 97] = 1;

i = 0;

j = 0;

for (k = 0; k < ks; k++) {

if (dicty[key[k] - 97] == 2) {

dicty[key[k] - 97] -= 1;

keyT[i][j] = key[k];

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

for (k = 0; k < 26; k++) {

if (dicty[k] == 0) {

keyT[i][j] = (char)(k + 97);

j++;

if (j == 5) {

i++;

j = 0;

}

}

}

}

// this function will search for the characters of a digraph

// in the key and return position of key

void search(char keyT[5][5], char a, char b, int arr[])

{

int i, j;

if (a == 'j')

a = 'i';

else if (b == 'j')

b = 'i';

for (i = 0; i < 5; i++) {

for (j = 0; j < 5; j++) {

if (keyT[i][j] == a) {

arr[0] = i;

arr[1] = j;

}

else if (keyT[i][j] == b) {

arr[2] = i;

arr[3] = j;

}

}

}

}

// this function will find the modulus with 5

int mod5(int a) { return (a % 5); }

// this function will make the plain text length even

int prepare(char str[], int ptrs)

{

if (ptrs % 2 != 0) {

str[ptrs++] = 'z';

str[ptrs] = '\0';

}

return ptrs;

}

// encryption will done using this function

void encrypt(char str[], char keyT[5][5], int ps)

{

int i, a[4];

for (i = 0; i < ps; i += 2) {

search(keyT, str[i], str[i + 1], a);

if (a[0] == a[2]) {

str[i] = keyT[a[0]][mod5(a[1] + 1)];

str[i + 1] = keyT[a[0]][mod5(a[3] + 1)];

}

else if (a[1] == a[3]) {

str[i] = keyT[mod5(a[0] + 1)][a[1]];

str[i + 1] = keyT[mod5(a[2] + 1)][a[1]];

}

else {

str[i] = keyT[a[0]][a[3]];

str[i + 1] = keyT[a[2]][a[1]];

}

}

}

// this function will encrypt cipher text using Playfair Cipher algorithm

void encryptByPlayfairCipher(char str[], char key[])

{

char ps, ks, keyT[5][5];

ks = strlen(key);

ks = removeSpaces(key, ks);

toLowerCase(key, ks);

ps = strlen(str);

toLowerCase(str, ps);

ps = removeSpaces(str, ps);

ps = prepare(str, ps);

generateKeyTable(key, ks, keyT);

encrypt(str, keyT, ps);

}

// main code

int encryption()

{

char str[SIZE], key[SIZE];

// key text

strcpy(key, "Algorithm");

printf("Key text: %s\n", key);

// Plaintext

strcpy(str, "MBM");

printf("Plain text: %s\n", str);

// encryption using the "Playfair Cipher" algorithmn

encryptByPlayfairCipher(str, key);

printf("Cipher text: %s\n", str);

return 0;

}

// this function calls the decrypt function

void decrypt(char str[], char keyT[5][5], int ps)

{

int i, a[4];

for (i = 0; i < ps; i += 2) {

search(keyT, str[i], str[i + 1], a);

if (a[0] == a[2]) {

str[i] = keyT[a[0]][mod5(a[1] - 1)];

str[i + 1] = keyT[a[0]][mod5(a[3] - 1)];

}

else if (a[1] == a[3]) {

str[i] = keyT[mod5(a[0] - 1)][a[1]];

str[i + 1] = keyT[mod5(a[2] - 1)][a[1]];

}

else {

str[i] = keyT[a[0]][a[3]];

str[i + 1] = keyT[a[2]][a[1]];

}

}

}

// this function wil call the decrypt function

void decryptByPlayfairCipher(char str[], char key[])

{

char ps, ks, keyT[5][5];

// Key text

ks = strlen(key);

ks = removeSpaces(key, ks);

toLowerCase(key, ks);

// ciphertext

ps = strlen(str);

toLowerCase(str, ps);

ps = removeSpaces(str, ps);

generateKeyTable(key, ks, keyT);

decrypt(str, keyT, ps);

}

// main code

int decryption()

{

char str[SIZE], key[SIZE];

// Key text that needs to be encrypted

strcpy(key, "Algorithm");

printf("Key text: %s\n", key);

// Ciphertext that needs to be decrypted

strcpy(str, "pbnowv");

printf("Plain text: %s\n", str);

// encryption is done using Playfair Cipher algorithm

decryptByPlayfairCipher(str, key);

printf("Deciphered text: %s\n", str);

return 0;

}

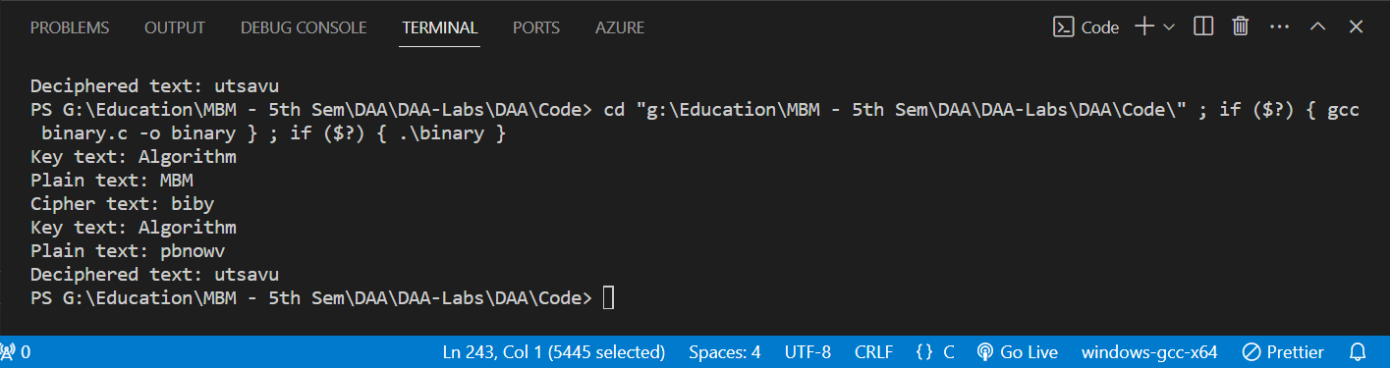
int main(){

encryption();

decryption();

}

**Output**

****