# **Playfair Cipher**

**Source Code**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void getinput(char key[], char plaintext[]);

void removesameletter(char text[], char modifiedtext[]);

int findindexofalphabet(char letter);

void createkeymatrix(int keymatrix[5][5], char key[]);

int checkmatrix(int keymatrix[5][5], int index);

void encryption(char plaintext[],int keymatrix[5][5], char encrypted[]);

void decryption(char encrypted[], int keymatrix[5][5], char decrypted[]);

int main()

{

char key[20], plaintext[100], modifiedtext[20], encrypted[100], decrypted[100];

int keymatrix[5][5]={30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30,30};

getinput(key, plaintext);

removesameletter(key, modifiedtext);

createkeymatrix(keymatrix, modifiedtext);

encryption(plaintext, keymatrix, encrypted);

decryption(strlwr(encrypted), keymatrix, decrypted);

}

void getinput(char key[], char plaintext[])

{

puts("Enter the Message:");

gets(plaintext);

strlwr(plaintext);

puts("KEY:");

gets(key);

strlwr(key);

}

void removesameletter(char text[], char modifiedtext[])

{

int i, j, flag=0, k=0;

for (i=0; text[i]!=NULL; i++){

for (j=0; modifiedtext[j]!=NULL;j++){

if (text[i]==modifiedtext[j]){

flag=1;

}

}

if (flag==0){

modifiedtext[k]=text[i];

k++;

modifiedtext[k]='\0';

}

flag=0;

}

}

int findindexofalphabet(char letter)

{

char alpha[27]= "abcdefghijklmnopqrstuvwxyz";

int i;

for (i=0; alpha[i]!=NULL;i++){

if (letter==alpha[i]){

return i;

}

}

}

int checkmatrix(int keymatrix[5][5], int index)

{

int i, j, flag=0;

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

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

if (index==keymatrix[i][j]){

flag=1;

break;

}

}

}

if (flag==1){

return 0;

}else{

return 1;

}

}

void createkeymatrix(int keymatrix[5][5],char key[])

{

int i, j, k, index=0;

char alpha[27]= "abcdefghiklmnopqrstuvwxyz";

char alphas[27]= "abcdefghijklmnopqrstuvwxyz";

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

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

if (index != strlen(key)){

keymatrix[i][j]=findindexofalphabet(key[index]);

index++;

}else{

break;

}

}

}

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

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

for (k=0; alpha[k]!=NULL;k++){

if (keymatrix[i][j]==30 && checkmatrix(keymatrix,findindexofalphabet(alpha[k]))){

keymatrix[i][j]=findindexofalphabet(alpha[k]);

}

}

printf("%c\t", alphas[keymatrix[i][j]]);

}

printf("\n");

}

}

void encryption(char plaintext[],int keymatrix[5][5], char encrypted[])

{

int i, j, k, index=0, length, index1, j1, k1, j2, k2, encryptindex=0;

char alpha[27]= "abcdefghijklmnopqrstuvwxyz";

char modifiedtext[50];

for (i=0;plaintext[i]!=NULL;i++){

if (plaintext[i]==plaintext[i+1]){

modifiedtext[index]=plaintext[i];

index++;

modifiedtext[index]='X';

index++;

}else{

modifiedtext[index]=plaintext[i];

index++;

modifiedtext[index]=plaintext[i+1];

index++;

i++;

}

}

if (index%2==0){

modifiedtext[index]='\0';

}

length=strlen(modifiedtext);

modifiedtext[length]=NULL;

if (length%2!=0){

modifiedtext[length]='X';

modifiedtext[length+1]='\0';

}

// puts(modifiedtext);

strlwr(modifiedtext);

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

index=findindexofalphabet(modifiedtext[i]);

index1=findindexofalphabet(modifiedtext[i+1]);

if (index==9){

index--;

}

if (index1==9){

index1--;

}

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

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

if (keymatrix[j][k]==index){

j1=j;

k1=k;

}

if (keymatrix[j][k]==index1){

j2=j;

k2=k;

}

}

}

if (j1==j2){

encrypted[encryptindex]=alpha[(keymatrix[j1][((k1+1)%5)])];

encryptindex++;

encrypted[encryptindex]=alpha[(keymatrix[j2][((k2+1)%5)])];

encryptindex++;

}else{

if (k1==k2){

encrypted[encryptindex]=alpha[(keymatrix[((j1+1)%5)][k1])];

encryptindex++;

encrypted[encryptindex]=alpha[(keymatrix[((j2+1)%5)][k2])];

encryptindex++;

}else{

encrypted[encryptindex]=alpha[(keymatrix[j1][k2])];

encryptindex++;

encrypted[encryptindex]=alpha[(keymatrix[j2][k1])];

encryptindex++;

}

}

i++;

}

encrypted[encryptindex]=NULL;

puts("ENCRYPTED Message:");

puts(strupr(encrypted));

}

void decryption(char encrypted[], int keymatrix[5][5], char decrypted[])

{

int i, j, k, length, j1, j2, k1, k2, index, index1, decryptindex=0;

char alpha[27]= "abcdefghijklmnopqrstuvwxyz";

length=strlen(encrypted);

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

index=findindexofalphabet(encrypted[i]);

index1=findindexofalphabet(encrypted[i+1]);

if (index==9){

index--;

}

if (index1==9){

index1--;

}

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

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

if (keymatrix[j][k]==index){

j1=j;

k1=k;

}

if (keymatrix[j][k]==index1){

j2=j;

k2=k;

}

}

}

if (j1==j2){

decrypted[decryptindex]=alpha[(keymatrix[j1][((k1-1)%5)])];

decryptindex++;

decrypted[decryptindex]=alpha[(keymatrix[j2][((k2-1)%5)])];

decryptindex++;

}else{

if (k1==k2){

decrypted[decryptindex]=alpha[(keymatrix[((j1-1)%5)][k1])];

decryptindex++;

decrypted[decryptindex]=alpha[(keymatrix[((j2-1)%5)][k2])];

decryptindex++;

}else{

decrypted[decryptindex]=alpha[(keymatrix[j1][k2])];

decryptindex++;

decrypted[decryptindex]=alpha[(keymatrix[j2][k1])];

decryptindex++;

}

}

i++;

}

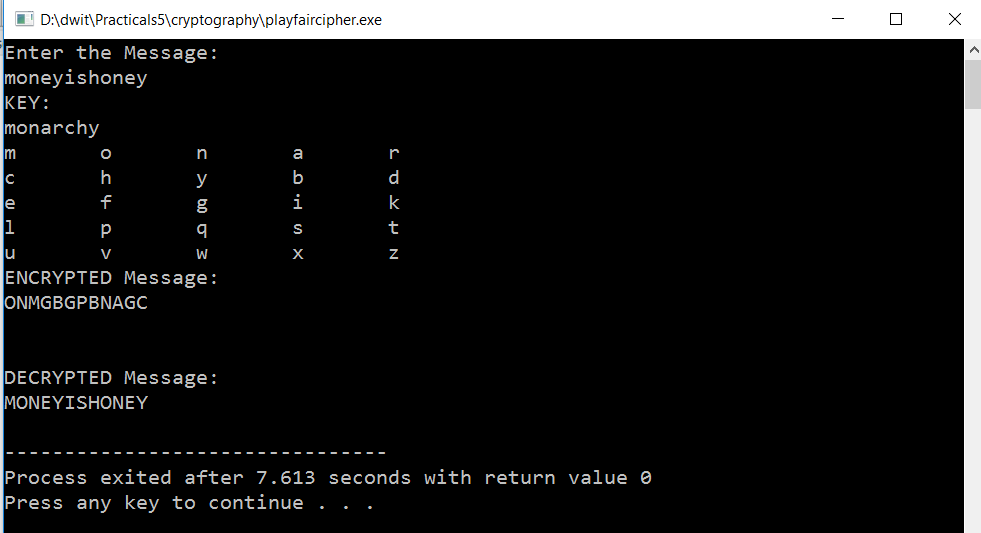
decrypted[decryptindex]=NULL;

puts("\n\nDECRYPTED Message:");

puts(strupr(decrypted));

}

**Output**

****

****

# **Hill Cipher**

**Source Code**

#include<stdio.h>

#include<conio.h>

#include<string.h>

#include<math.h>

int findindex(char letter);

void encryption(char message[], int key[3][3]);

void matrixmultiplication(int key[3][3],int msgmatrix[5][5],char text[],int msgcolumn);

void decryption(char encrypted[], int key[3][3]);

int greatestcommondivisor(int a, int b);

int row,column;

int main()

{

char message[50], alphabet[]="abcdefghijklmnopqrstuvwxyz";

int key[3][3];

int i,j;

puts("Enter the Message:");

gets(message);

puts("Enter row and column size:");

scanf("%d%d", &row, &column);

puts("Enter the Key:");

for (i=0; i<row;i++){ //takes input

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

scanf("%d", &key[i][j]);

}

}

printf("\n");

encryption(message, key);

//decryption(message, key);

}

void encryption(char message[], int key[3][3])

{

int msgmatrix[5][5], i,j, messageindex=0, msgcolumn;

char encryptedtext[50];

if (row==2 && column==2){

if (strlen(message)%2!=0){

message[strlen(message)]='x'; //balances string if it is odd

message[strlen(message)+1]='\0';

}

}else{

if (row==3 && column == 3){

if (strlen(message)%3==2){

message[strlen(message)]='x';

message[strlen(message)+1]='\0';

}

if (strlen(message)%3==1){

message[strlen(message)]='x';

message[strlen(message)+1]='x';

message[strlen(message)+2]='\0';

}

}

}

msgcolumn=strlen(message)/row;

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

for (j=0; j<row;j++){ //converting plaintext into matrix

msgmatrix[j][i]=findindex(message[messageindex]);

messageindex++;

}

}

// puts(message);

matrixmultiplication(key, msgmatrix, encryptedtext, msgcolumn);

puts("Encrypted:");

puts(strupr(encryptedtext)); //display encryptedtext

decryption(strlwr(encryptedtext), key);

}

void decryption(char encrypted[], int key[3][3])

{

int i, j,x, detkey, detkeyinverse, temp, encryptindex=0, encryptedmatrix[5][5], encryptcolumn, keyinverse[3][3];

char decryptedtext[50];

// puts(encrypted);

if (row==2 && column==2){ //finds determinant of 2\*2 matrix

detkey=key[0][0]\*key[1][1] - key[1][0]\*key[0][1];

}else{

if (row==3 && column==3){ //finds determinant of 3\*3 matrix

detkey = key[0][0]\*((key[1][1]\*key[2][2]) - (key[2][1]\*key[1][2])) -key[0][1]\*(key[1][0]\*key[2][2] - key[2][0]\*key[1][2]) + key[0][2]\*(key[1][0]\*key[2][1] - key[2][0]\*key[1][1]);

}

}

if (detkey>25 || detkey < 0){

detkey=detkey%26;

}

if (detkey<0){

detkey=detkey+26;

}

if (greatestcommondivisor(detkey, 26)==1){

for (x=0; x<1000; x++){

if ((detkey\*x)%26==1){ //finds determinant inverse

detkeyinverse=x;

//printf("%d", detkeyinverse);

break;

}

}

if (row==2 && column==2){

keyinverse[0][0]=((key[row-1][column-1])\*detkeyinverse)%26;

keyinverse[row-1][column-1]=(key[0][0]\*detkeyinverse)%26;

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

for (j=0; j<column;j++){ //calculating inverse matrix of key of 2\*2 matrix

if (i!=j){

keyinverse[i][j]=((-key[i][j]+26)\*detkeyinverse)%26;

}

}

}

}else{

if (row==3 && column==3){

for(i=0;i<3;i++){

for(j=0;j<3;j++){

keyinverse[j][i]=((key[(i+1)%3][(j+1)%3] \* key[(i+2)%3][(j+2)%3])-(key[(i+1)%3][(j+2)%3]\*key[(i+2)%3][(j+1)%3]))%26; //calculates adjoint of 3\*3 matrix

if (keyinverse[j][i]<0){

keyinverse[j][i]=keyinverse[j][i]+26;

}

keyinverse[j][i]=(keyinverse[j][i]\*detkeyinverse)%26; //calculates inverse matrix of key of 3\*3 matrix

if (keyinverse[j][i]<0){

keyinverse[j][i]=keyinverse[j][i]+26;

}

}

}

}

}

encryptcolumn=strlen(encrypted)/row;

for (i=0; i<encryptcolumn;i++){ //converting encrypted text into matrix

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

encryptedmatrix[j][i]=findindex(encrypted[encryptindex]);

encryptindex++;

}

}

matrixmultiplication(keyinverse, encryptedmatrix, decryptedtext, encryptcolumn);

puts("Decrypted:");

puts(strupr(decryptedtext));

}else{

printf("There is no multiplicative inverse of %d and 26.", detkey);

}

}

void matrixmultiplication(int key[3][3],int msgmatrix[5][5],char text[],int msgcolumn)

{

int multipliedmatrix[5][5];

int i, j, k, textindex=0;

char alphabet[]="abcdefghijklmnopqrstuvwxyz";

for (i=0; i<row; i++){ //multiplies any order of matrix

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

multipliedmatrix[i][j]=0;

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

multipliedmatrix[i][j]+=key[i][k]\*msgmatrix[k][j];

multipliedmatrix[i][j]=multipliedmatrix[i][j]%26;

}

}

}

for (i=0; i<msgcolumn;i++){ //convert multiplied matrix into text

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

text[textindex]=alphabet[multipliedmatrix[j][i]];

textindex++;

}

}

text[textindex]='\0';

}

int findindex(char letter) //returns an index of an alphabet

{

char alphabet[]="abcdefghijklmnopqrstuvwxyz";

int i, j;

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

if (letter==alphabet[i]){

return i;

}

}

}

int greatestcommondivisor(int a, int b) //check if gcd of two number is 1 or not

{

int i;

for (i=2; i<=b; i++){

if (a%i==0 && b%i==0){

return i;

}else{

return 1;

}

}

}

**Output**

