# **Caesar Cipher**

## **Coding**

#include<stdio.h>

#include<conio.h>

#include<math.h>

void encryption(char message[50]);

void decryption(char message[50]);

char alphabet[100]="abcdefghijklmnopqrstuvwxyz";

int main()

{

char message[50];

printf("Enter message:");

gets(message);

encryption(message);

}

void encryption(char message[])

{

char encryptmessage[50];

int k=3, i, j;

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

if (message[i] != ' '){

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

if (message[i]==alphabet[j]){

encryptmessage[i]=alphabet[(j+k)%26];

}

}

}else{

encryptmessage[i]=' ';

}

}

encryptmessage[i]=NULL;

printf("\n");

printf("Encrypted message:");

puts(encryptmessage);

decryption(encryptmessage);

getch();

}

void decryption(char message[50])

{

char decryptmessage[50];

int k=3, i, j;

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

if (message[i] != ' '){

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

if (message[i]==alphabet[j]){

decryptmessage[i]=alphabet[((j-k)+26)%26];

}

}

}else{

decryptmessage[i]=' ';

}

}

decryptmessage[i]=NULL;

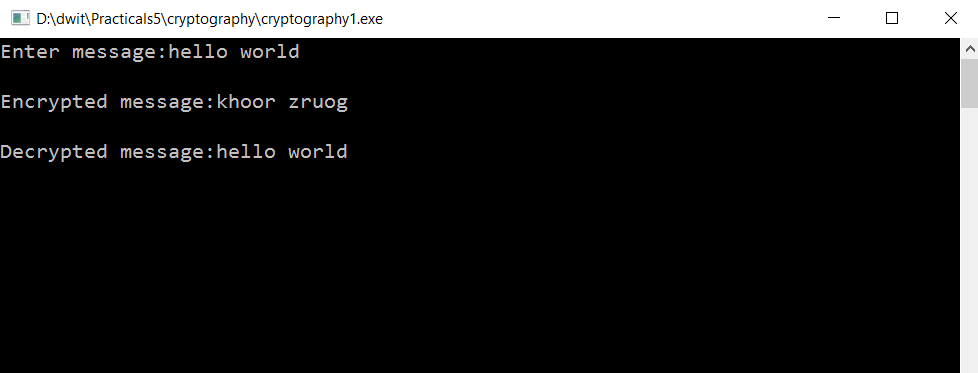
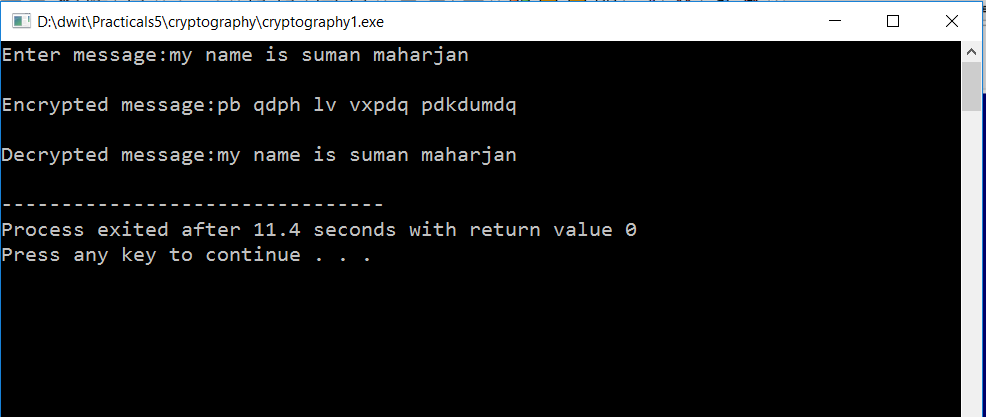
printf("\n");

printf("Decrypted message:");

puts(decryptmessage);

}

## **Output**



# **Attacking the Caesar Cipher (Statistical attack)**

## **Coding**

#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

#include<string.h>

int stringlength(char text[]);

int charactercount(char character, char text[]);

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

void sortindesc(float array[]);

void decryption(char message[50], int k, char alphabet[]);

void checkstring(char decrypted[]);

void splitstring(char text[]);

int wordcount(char text[]);

int printcount=0, checkcount=0;

int main()

{

char alpha[27]= "abcdefghijklmnopqrstuvwxyz";

float frequencies[27]={0.080, 0.015, 0.030, 0.040, 0.130, 0.020, 0.015, 0.060, 0.065, 0.005, 0.005, 0.035, 0.030, 0.070, 0.080, 0.020, 0.002, 0.065, 0.060, 0.090, 0.030, 0.010, 0.015, 0.005, 0.020, 0.002};

int length, i, j, k;

float frequency[27], probablekey[27], sorted[27];

char encrypted[50], modifiedtext[50];

puts("Enter the encrypted message:");

gets(encrypted);

length=stringlength(encrypted);

removesameletter(encrypted, modifiedtext);

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

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

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

frequency[j]=(float)charactercount(modifiedtext[i], encrypted)/length;

}

}

}

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

probablekey[i]=0;

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

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

if (modifiedtext[j]==alpha[k]){

probablekey[i]=probablekey[i]+(frequency[k] \* frequencies[(k-i+26)%26]);

}

}

}

sorted[i]=probablekey[i];

}

sortindesc(sorted);

printf("Decrypted Message:\n");

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

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

if (sorted[i]==probablekey[j]){

decryption(encrypted, j, alpha);

if (printcount==wordcount(encrypted)){

getch();

return 0;

}

}

}

}

if (checkcount>25)

puts("Word NOT FOUND in the Vocabulary");

}

void decryption(char message[50], int k, char alphabet[])

{

char decryptmessage[50];

int i, j;

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

if (message[i] != ' '){

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

if (message[i]==alphabet[j]){

decryptmessage[i]=alphabet[((j-k)+26)%26];

}

}

}else{

decryptmessage[i]=' ';

}

}

decryptmessage[i]=NULL;

checkcount++;

splitstring(decryptmessage);

}

int wordcount(char text[])

{

int word=1, i;

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

if (text[i]==' '){

word++;

}

}

return word;

}

void splitstring(char text[]){

int word,i,j,k,initial=0;

char text1[20];

word=wordcount(text);

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

k=0;

for (j=initial; text[j]!=' '; j++){

text1[k]=text[j];

k++;

if (text[j]==NULL){

break;

}

}

text1[k]='\0';

initial=j+1;

checkstring(text1);

}

return;

}

void checkstring(char decrypted[])

{

FILE \*ptr\_file;

char buf[100];

ptr\_file =fopen("dictionary.txt","r");

if (!ptr\_file){

puts("\nFILE DOESNOT EXISTS.\n");

exit(0);

}

while (fgets(buf,100, ptr\_file)!=NULL){

fscanf(ptr\_file,"%s",buf);

if (strcmp(buf, decrypted) == 0){

puts(buf);

printcount++;

checkcount=0;

break;

}

}

fclose(ptr\_file);

}

int stringlength(char text[])

{

int i, length=0;

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

if (text[i]>='A' && text[i<='z']){

length++;

}

}

return length;

}

int charactercount(char character, char text[])

{

int i,j, count=0;

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

if (character==text[i]){

count++;

}

}

return count;

}

char 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;

}

}

void sortindesc(float array[])

{

int i, j;

float temp;

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

for (j=i+1; j<26;j++){

if (array[i]<array[j]){

temp=array[j];

array[j]=array[i];

array[i]=temp;

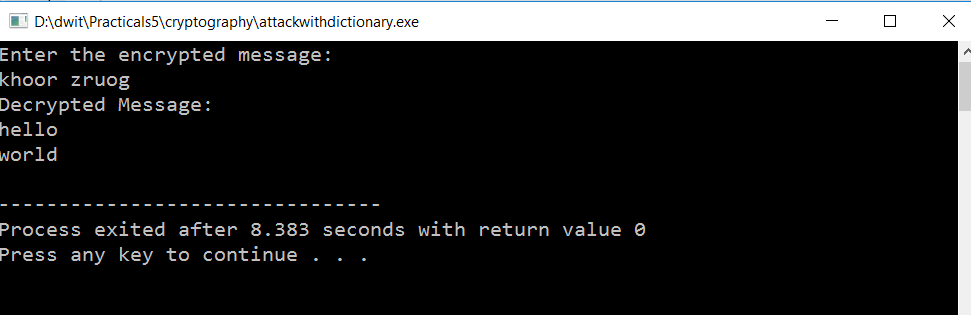
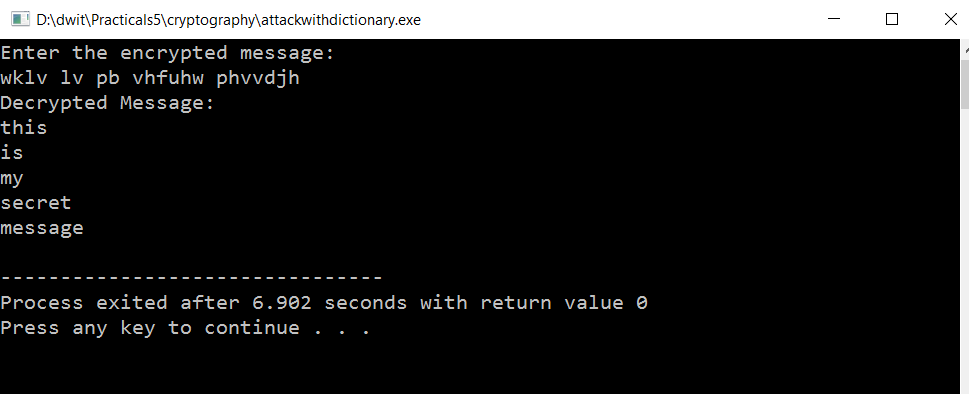
}

}

}

}

## **Output**



# **Vigenere Cipher: Substitution Cipher( Polyalphateic)**

## **Coding**

#include<stdio.h>

#include<conio.h>

#include<string.h>

void vigenerecipher(char message[], char key[]);

void vigeneredecipher(char encrypted[], char key[]);

void convertkeyandtext(char text[], char key[], char converted[]);

int main()

{

char key[20], message[100];

puts("Enter Message:");

gets(message);

puts("Enter key:");

gets(key);

vigenerecipher(message, key);

}

void convertkeyandtext(char text[], char key[], char converted[])

{

int i, k=0, index=0;

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

if (text[i]!=' '){

converted[index]=key[(k+strlen(key))%strlen(key)];

index++;

converted[index]=text[i];

index++;

k++;

}else{

converted[index]=' ';

index++;

}

}

converted[index]==NULL;

}

void vigenerecipher(char message[], char key[])

{

char alphabet[27]="abcdefghijklmnopqrstuvwxyz";

char beforencrypted[100], encrypted[100];

int i, j, k=0, keyindex, messageindex;

convertkeyandtext(message, key, beforencrypted);

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

if (beforencrypted[i] != ' '){

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

if (beforencrypted[i]==alphabet[j]){

keyindex=j;

}

if (beforencrypted[i+1]==alphabet[j]){

messageindex=j;

}

}

encrypted[k]=alphabet[(keyindex+messageindex)%26];

k++;

i++;

}else{

encrypted[k]=' ';

k++;

}

}

encrypted[strlen(message)]=NULL;

puts("Encrypted Message:");

puts(strupr(encrypted));

vigeneredecipher(strlwr(encrypted), key);

}

void vigeneredecipher(char encrypted[], char key[])

{

char alphabet[27]="abcdefghijklmnopqrstuvwxyz";

char beforedecrypted[100], decrypted[100];

int i, j, k=0, messageindex, keyindex;

convertkeyandtext(encrypted, key, beforedecrypted);

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

if (beforedecrypted[i] != ' '){

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

if (beforedecrypted[i]==alphabet[j]){

keyindex=j;

}

if (beforedecrypted[i+1]==alphabet[j]){

messageindex=j;

}

}

decrypted[k]=alphabet[(messageindex-keyindex+26)%26];

k++;

i++;

}else{

decrypted[k]=' ';

k++;

}

}

decrypted[strlen(encrypted)]=NULL;

puts("\nDecrypted Message:");

puts(decrypted);

}

## **Output**

