CNS-Programs

R. Aishwarya

192125001

1.Print the elements using the XOR.

Code:

#include<stdlib.h>

#include<stdio.h>

int main()

{

char str[]="WELCOME";

char str1[11];

int i,len;

len=strlen(str);

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

{

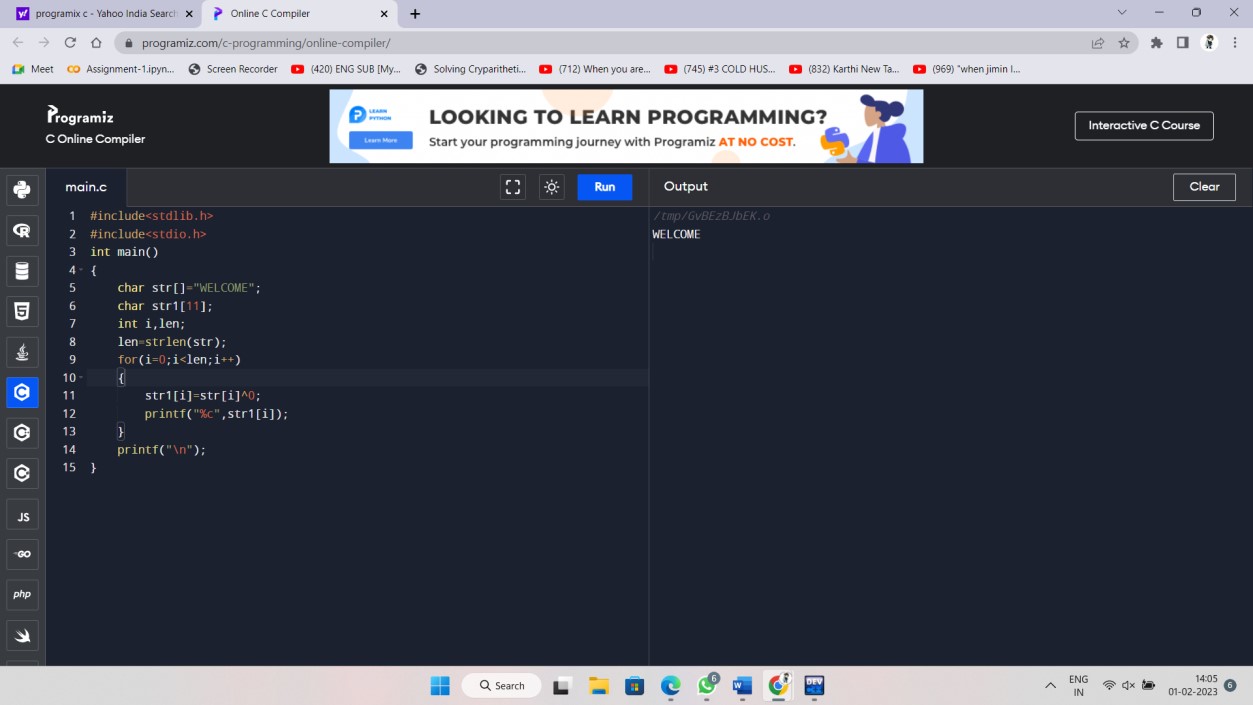
str1[i]=str[i]^0;

printf("%c",str1[i]);

}

printf("\n");

}



1. Write a C program for Caesar cipher involves replacing each letter of the alphabet with the letter stan places further down the alphabet, for k in the range 1 through 25.

Code:  
#include <stdio.h>

int main()

{

int i, x,n;

char str[100];

printf("\nPlease enter a string:\t");

gets(str);

printf("\nEnter the key value:");

scanf("%d",&n);

switch(x=1)

{

case 1:

for(i = 0; (i < 100 && str[i] != '\0'); i++)

str[i] = str[i] + n;

printf("\nEncrypted string: %s\n", str);

case 2:

for(i = 0; (i < 100 && str[i] != '\0'); i++)

str[i] = str[i] - n;

printf("\nDecrypted string: %s\n", str);

break;

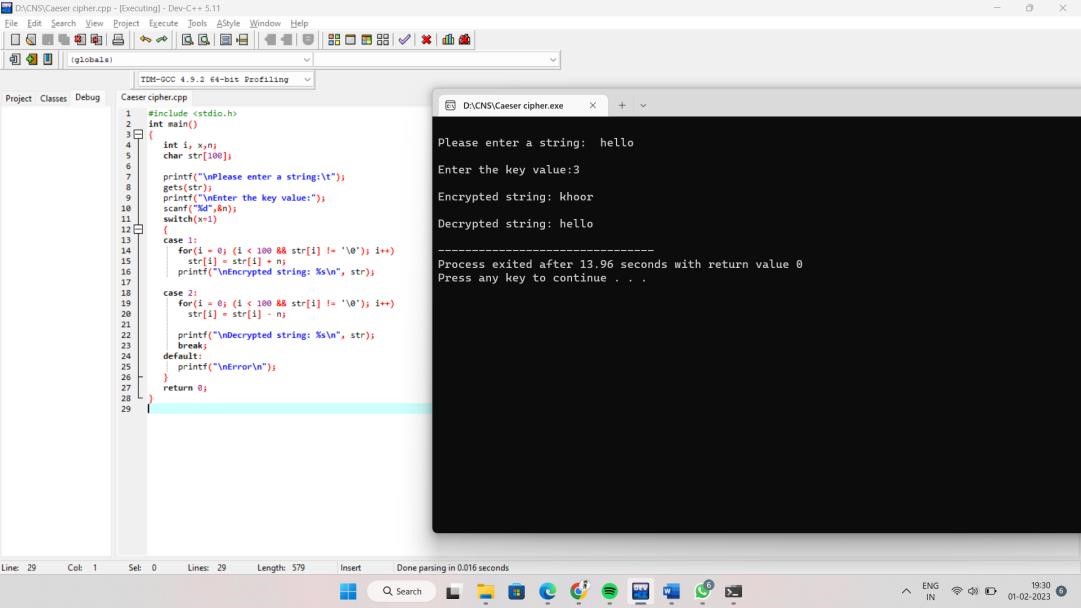
default:

printf("\nError\n");

}

return 0;

}



1. . Write a C program for Playfair algorithm is based on the use of a 5 X 5 matrix of letters constructed using a keyword. Plaintext is encrypted two letters at a time using this matrix.

Code:  
#include<stdio.h>

#include<string.h>

#include<ctype.h>

int removerepeated(int size,int a[]);

int insertelementat(int position,int a[],int size);

main()

{

int i,j,k,numstr[100],numcipher[100],numkey[100],lenkey,templen,tempkey[100],flag=-1,size,cipherkey[5][5],lennumstr,row1,row2,col1,col2;

char str[100],key[100];

printf("Enter a string\n");

gets(str);

for(i=0,j=0;i<strlen(str);i++)

{

if(str[i]!=' ')

{

str[j]=toupper(str[i]);

j++;

}

}

str[j]='\0';

printf("Entered String is %s\n",str);

size=strlen(str);

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

{

if(str[i]!=' ')

numstr[i]=str[i]-'A';

}

lennumstr=i;

printf("Enter the key (Non repeated elements if possible)\n");

gets(key);

//converting entered key to Capital letters

for(i=0,j=0;i<strlen(key);i++)

{

if(key[i]!=' ')

{

key[j]=toupper(key[i]);

j++;

}

}

key[j]='\0';

printf("%s\n",key);

k=0;

for(i=0;i<strlen(key)+26;i++)

{

if(i<strlen(key))

{

if(key[i]=='J')

{

flag=8;

printf("%d",flag);

}

numkey[i]=key[i]-'A';

}

else

{

if(k!=9 && k!=flag)

{

numkey[i]=k;

}

k++;

}

}

templen=i;

lenkey=removerepeated(templen,numkey);

printf("Entered key converted according to Play Fair Cipher rule\n");

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

{

printf("%c",numkey[i]+'A');

}

printf("\n");

k=0;

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

{

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

{

cipherkey[i][j]=numkey[k];

k++;

}

}

printf("Arranged key\n");

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

{

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

{

printf("%c ",cipherkey[i][j]+'A');

}

printf("\n");

}

for(i=0;i<lennumstr;i+=2)

{

if(numstr[i]==numstr[i+1])

{

insertelementat(i+1,numstr,lennumstr);

lennumstr++;

}

}

if(lennumstr%2!=0)

{

insertelementat(lennumstr,numstr,lennumstr);

lennumstr++;

}

printf("Entered String/Message After Processing according to Play fair cipher rule\n");

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

{

printf("%c",numstr[i]+'A');

}

for(k=0;k<lennumstr;k+=2)

{

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

{

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

{

if(numstr[k]==cipherkey[i][j])

{

row1=i;

col1=j;

}

if(numstr[k+1]==cipherkey[i][j])

{

row2=i;

col2=j;

}

}

}

if(row1==row2)

{

col1=(col1-1)%5;

col2=(col2-1)%5;

if(col1<0)

{

col1=5+col1;

}

if(col2<0)

{

col2=5+col2;

}

numcipher[k]=cipherkey[row1][col1];

numcipher[k+1]=cipherkey[row2][col2];

}

if(col1==col2)

{

row1=(row1-1)%5;

row2=(row2-1)%5;

if(row1<0)

{

row1=5+row1;

}

if(row2<0)

{

row2=5+row2;

}

numcipher[k]=cipherkey[row1][col1];

numcipher[k+1]=cipherkey[row2][col2];

}

if(row1!=row2&&col1!=col2)

{

numcipher[k]=cipherkey[row1][col2];

numcipher[k+1]=cipherkey[row2][col1];

}

}

printf("\nCipher Text is\n");

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

{

if((numcipher[i]+'A')!='X')//Should remove extra 'X' which were created during Encryption

printf("%c",numcipher[i]+'A');

}

printf("\n");

}

int removerepeated(int size,int a[])

{

int i,j,k;

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

{

for(j=i+1;j<size;)

{

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

{

for(k=j;k<size;k++)

{

a[k]=a[k+1];

}

size--;

}

else

{

j++;

}

}

}

return(size);

}

int insertelementat(int position,int a[],int size)

{

int i,insitem=23,temp[size+1];

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

{

if(i<position)

{

temp[i]=a[i];

}

if(i>position)

{

temp[i]=a[i-1];

}

if(i==position)

{

temp[i]=insitem;

}

}

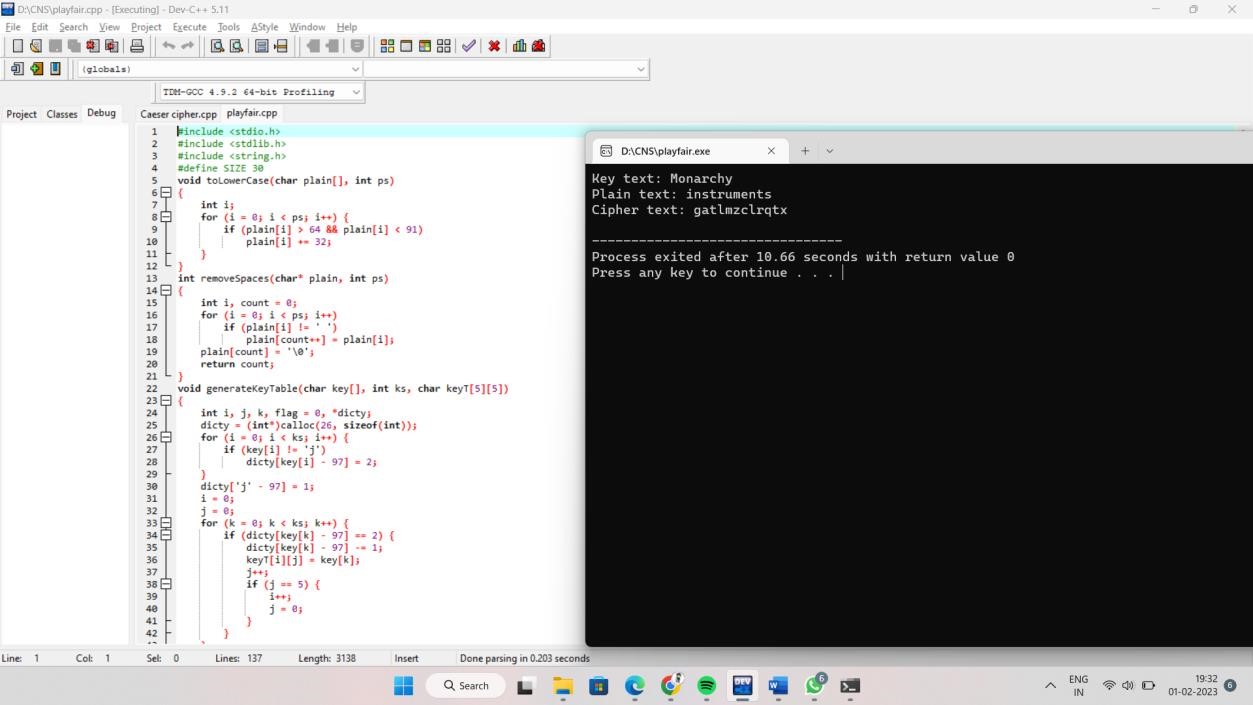
for(i=0;i<=size;i++)

{

a[i]=temp[i];

}

}



1. Write a C program for polyalphabetic substitution cipher uses a separate monoalphabetic substitution cipher for each successive letter of plaintext, depending on a key.

Code:

#include<stdio.h>

#include<conio.h>

#include<string.h>

int main()

{

char pt[20]={'\0'},ct[20]={'\0'},key[20]={'\0'},rt[20]={'\0'};

int i,j;

printf("\n enter the plain text:");

scanf("%s",pt);

printf("\n enter the key:");

scanf("%s",key);

j=0;

for(i=strlen(key);i<strlen(pt);i++)

{

if(j==strlen(key))

{

j=0;

}

key[i]=key[j];

j++;

}

for(i=0;i<strlen(pt);i++)

{

ct[i]=(((pt[i]-97)+(key[i]-97))%26)+97;

}

printf("\n \n cipher text is:%s",ct);

for(i=0;i<strlen(ct);i++)

{

if(ct[i]<key[i])

{

rt[i]=26+((ct[i]-97)-(key[i]-97))+97;

}

else

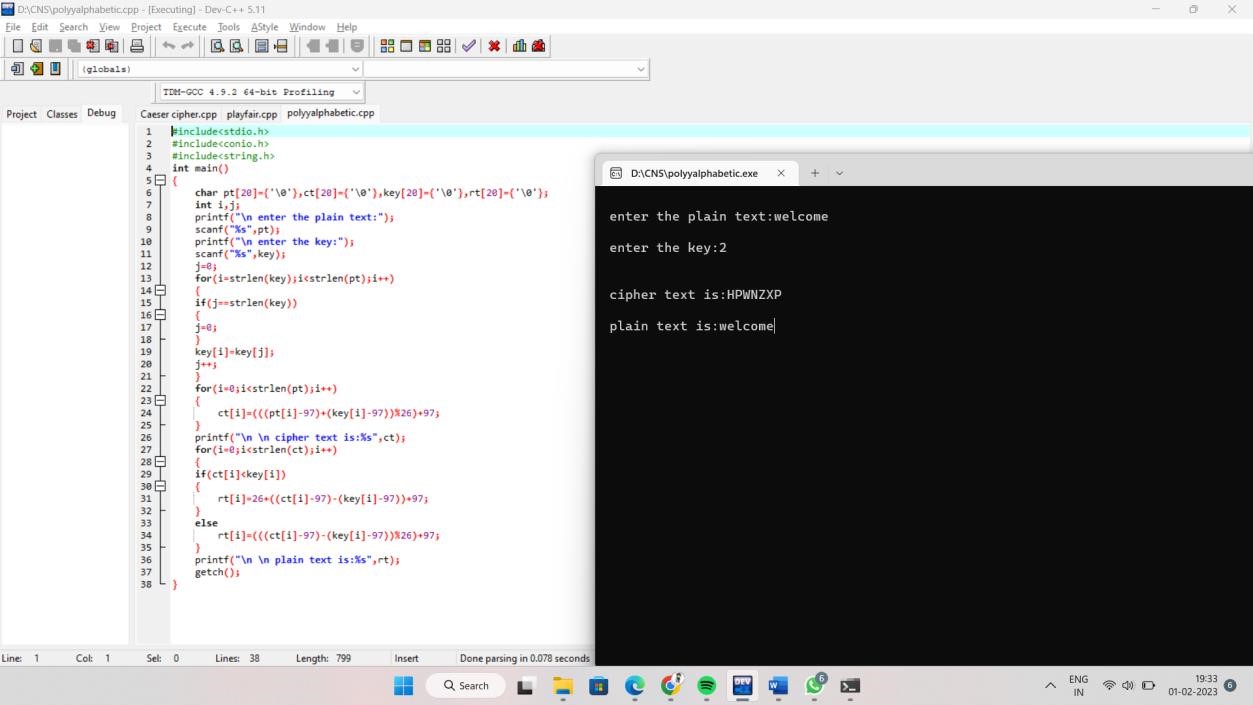
rt[i]=(((ct[i]-97)-(key[i]-97))%26)+97;

}

printf("\n \n plain text is:%s",rt);

getch();

}



5.Railfence c program.

Code:

#include<stdio.h>

#include<string.h>

void encryptMsg(char msg[], int key){

int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0;

char railMatrix[key][msgLen];

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

for(j = 0; j < msgLen; ++j)

railMatrix[i][j] = '\n';

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

railMatrix[row][col++] = msg[i];

if(row == 0 || row == key-1)

k= k \* (-1);

row = row + k;

}

printf("\nEncrypted Message: ");

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

for(j = 0; j < msgLen; ++j)

if(railMatrix[i][j] != '\n')

printf("%c", railMatrix[i][j]);

}

void decryptMsg(char enMsg[], int key){

int msgLen = strlen(enMsg), i, j, k = -1, row = 0, col = 0, m = 0;

char railMatrix[key][msgLen];

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

for(j = 0; j < msgLen; ++j)

railMatrix[i][j] = '\n';

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

railMatrix[row][col++] = '\*';

if(row == 0 || row == key-1)

k= k \* (-1);

row = row + k;

}

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

for(j = 0; j < msgLen; ++j)

if(railMatrix[i][j] == '\*')

railMatrix[i][j] = enMsg[m++];

row = col = 0;

k = -1;

printf("\nDecrypted Message: ");

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

printf("%c", railMatrix[row][col++]);

if(row == 0 || row == key-1)

k= k \* (-1);

row = row + k;

}

}

int main(){

char msg[] = "Hello World";

char enMsg[] = "Horel ollWd";

int key = 3;

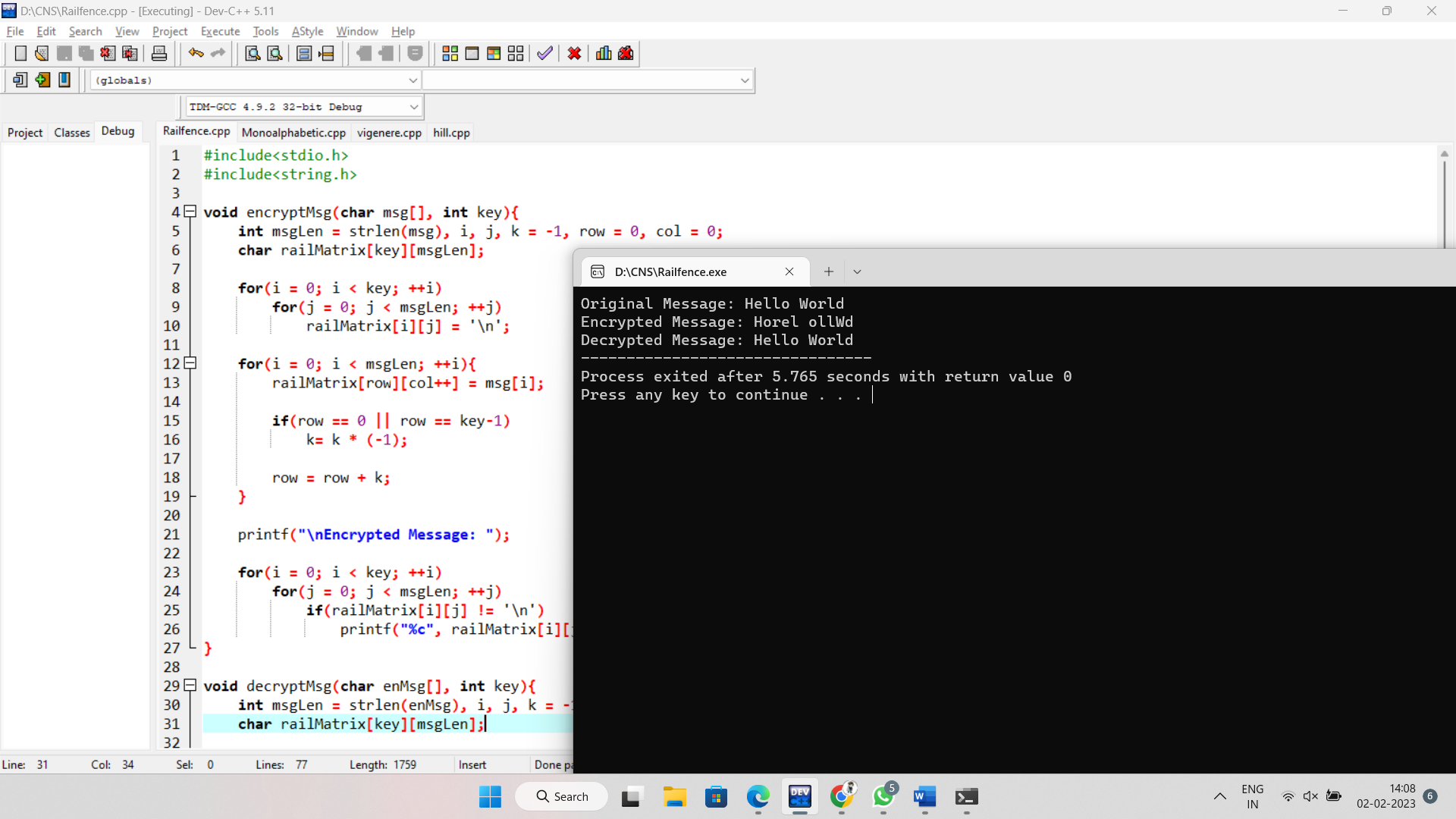
printf("Original Message: %s", msg);

encryptMsg(msg, key);

decryptMsg(enMsg, key);

return 0;

}



6.Monoalphabetic cipher c programming.

Code:

#include<stdio.h>

char monocipher\_encr(char);

char alpha[27][3] = { { 'a', 'f' }, { 'b', 'a' }, { 'c', 'g' }, { 'd', 'u' }, {

'e', 'n' }, { 'f', 'i' }, { 'g', 'j' }, { 'h', 'k' }, { 'i', 'l' }, {

'j', 'm' }, { 'k', 'o' }, { 'l', 'p' }, { 'm', 'q' }, { 'n', 'r' }, {

'o', 's' }, { 'p', 't' }, { 'q', 'v' }, { 'r', 'w' }, { 's', 'x' }, {

't', 'y' }, { 'u', 'z' }, { 'v', 'b' }, { 'w', 'c' }, { 'x', 'd' }, {

'y', 'e' }, { 'z', 'h' } };

char str[20];

int main() {

char str[20], str2[20];

int i;

printf("\n Enter String..");

gets(str);

for (i = 0; str[i]; i++) {

str2[i] = monocipher\_encr(str[i]);

}

str2[i] = '\0';

printf("\n Before Decryption..%s", str);

printf("\n After Decryption..%s\n", str2);

}

char monocipher\_encr(char a) {

int i;

for (i = 0; i < 27; i++) {

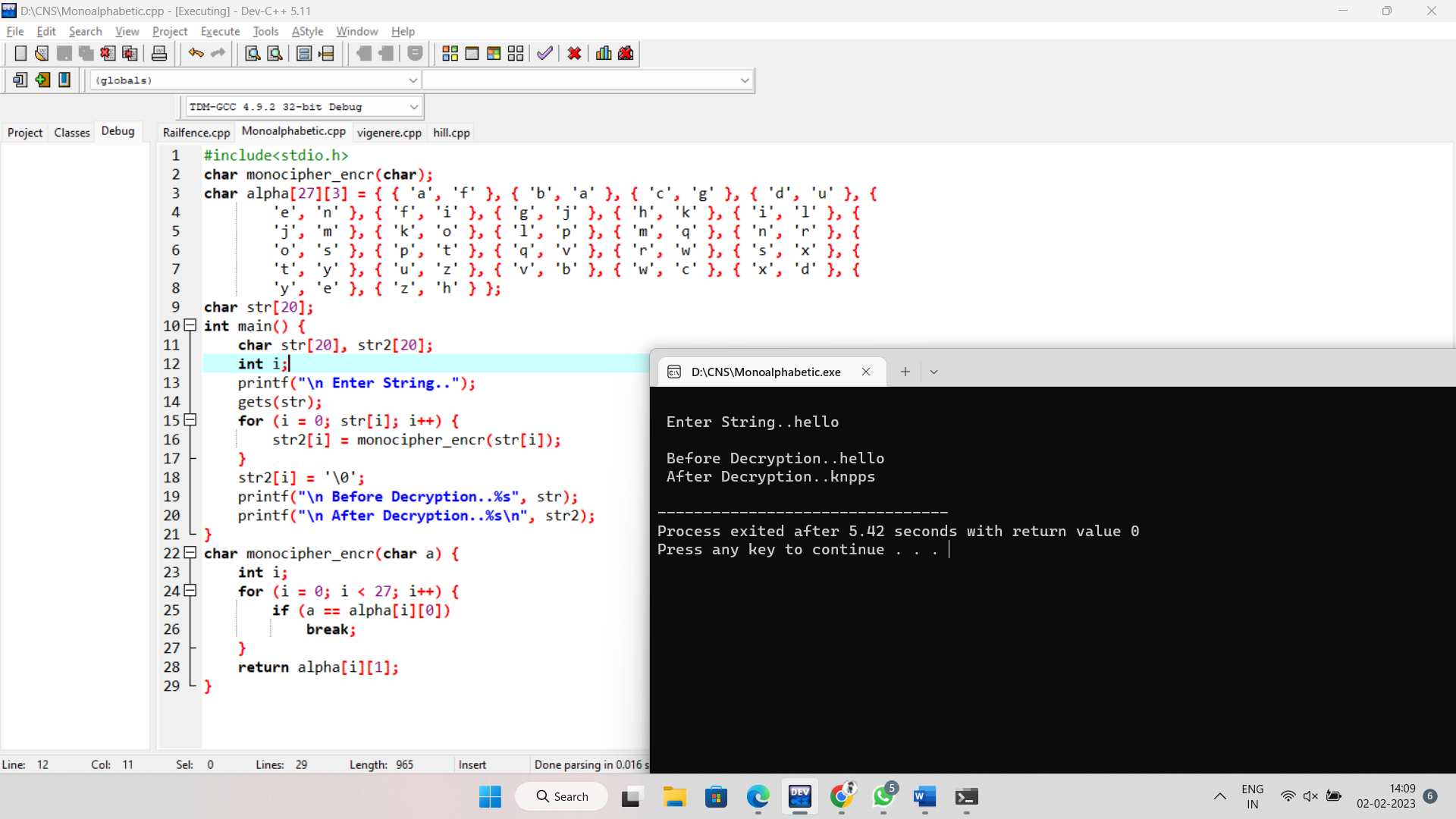
if (a == alpha[i][0])

break;

}

return alpha[i][1];

}



7. Vigenere Cipher

Code:

#include<stdio.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

int main()

{

int i,j,k,numstr[100],numkey[100],numcipher[100];

char str[100],key[100];

printf("Enter a string\n");

gets(str);

for(i=0,j=0;i<strlen(str);i++)

{

if(str[i]!=' ')

{

str[j]=toupper(str[i]);

j++;

}

}

str[j]='\0';

printf("Entered string is : %s \n",str);

for(i=0;i<strlen(str);i++)

{

numstr[i]=str[i]-'A';

}

printf("Enter a key\n");

gets(key);

for(i=0,j=0;i<strlen(key);i++)

{

if(key[i]!=' ')

{

key[j]=toupper(key[i]);

j++;

}

}

key[j]='\0';

for(i=0;i<strlen(str);)

{

for(j=0;(j<strlen(key))&&(i<strlen(str));j++)

{

numkey[i]=key[j]-'A';

i++;

}

}

for(i=0;i<strlen(str);i++)

{

numcipher[i]=numstr[i]+numkey[i];

}

for(i=0;i<strlen(str);i++)

{

if(numcipher[i]>25)

{

numcipher[i]=numcipher[i]-26;

}

}

printf("Vigenere Cipher text is\n");

for(i=0;i<strlen(str);i++)

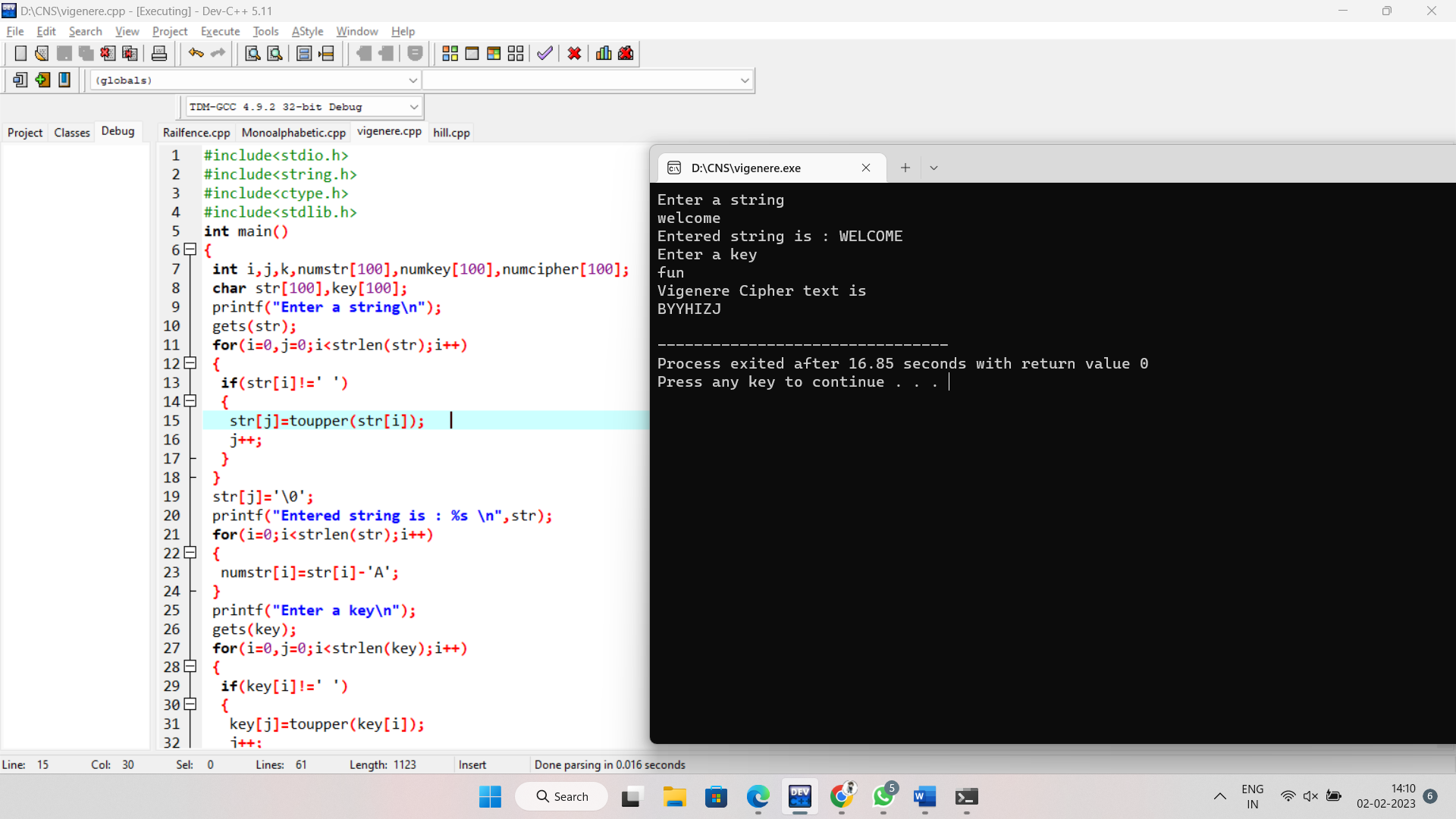
{

printf("%c",(numcipher[i]+'A'));

}

printf("\n");

}



8.Railfence technique.

Code:

#include<stdio.h>

#include<string.h>

void encryptMsg(char msg[], int key){

int msgLen = strlen(msg), i, j, k = -1, row = 0, col = 0;

char railMatrix[key][msgLen];

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

for(j = 0; j < msgLen; ++j)

railMatrix[i][j] = '\n';

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

railMatrix[row][col++] = msg[i];

if(row == 0 || row == key-1)

k= k \* (-1);

row = row + k;

}

printf("\nEncrypted Message: ");

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

for(j = 0; j < msgLen; ++j)

if(railMatrix[i][j] != '\n')

printf("%c", railMatrix[i][j]);

}

void decryptMsg(char enMsg[], int key){

int msgLen = strlen(enMsg), i, j, k = -1, row = 0, col = 0, m = 0;

char railMatrix[key][msgLen];

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

for(j = 0; j < msgLen; ++j)

railMatrix[i][j] = '\n';

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

railMatrix[row][col++] = '\*';

if(row == 0 || row == key-1)

k= k \* (-1);

row = row + k;

}

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

for(j = 0; j < msgLen; ++j)

if(railMatrix[i][j] == '\*')

railMatrix[i][j] = enMsg[m++];

row = col = 0;

k = -1;

printf("\nDecrypted Message: ");

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

printf("%c", railMatrix[row][col++]);

if(row == 0 || row == key-1)

k= k \* (-1);

row = row + k;

}

}

int main(){

char msg[] = "Hello";

char enMsg[] = "Horel ";

int key = 3;

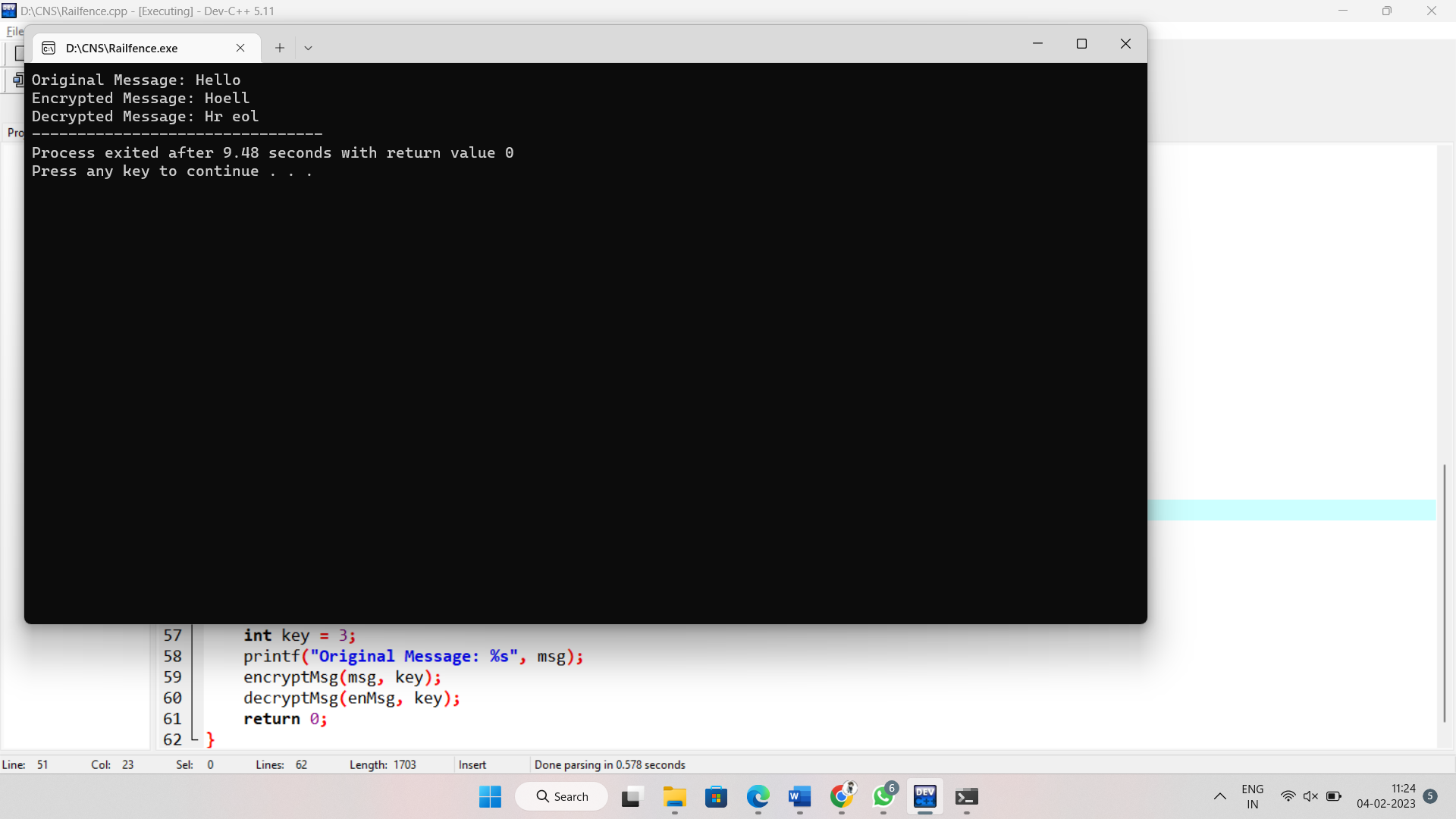
printf("Original Message: %s", msg);

encryptMsg(msg, key);

decryptMsg(enMsg, key);

return 0;

}



9.DSA:

Code:

#include <stdio.h>

#include <string.h>

static void display(int intArray[], int length)

{

int i=0;

printf("Array : [");

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

{

printf(" %d ", intArray[i]);

}

printf(" ]\n ");

}

int main()

{

int i = 0;

int intArray[8];

for ( i = 0; i < 8; i++ )

{

intArray[ i ] = 0;

}

display(intArray,8);

for(i = 0; i < 8; i++)

{

printf("Adding %d at index %d\n",i,i);

intArray[i] = i;

}

printf("\n");

printf("Array after adding data. ");

display(intArray,8);

int index = 5;

intArray[index] = 10;

printf("Array after updating element at index %d.\n",index);

display(intArray,8);

printf("Data at index %d:%d\n" ,index,intArray[index]);

int value = 4;

for(i = 0; i < 8; i++) {

if(intArray[i] == value ){

printf("value %d Found at index %d \n", intArray[i],i);

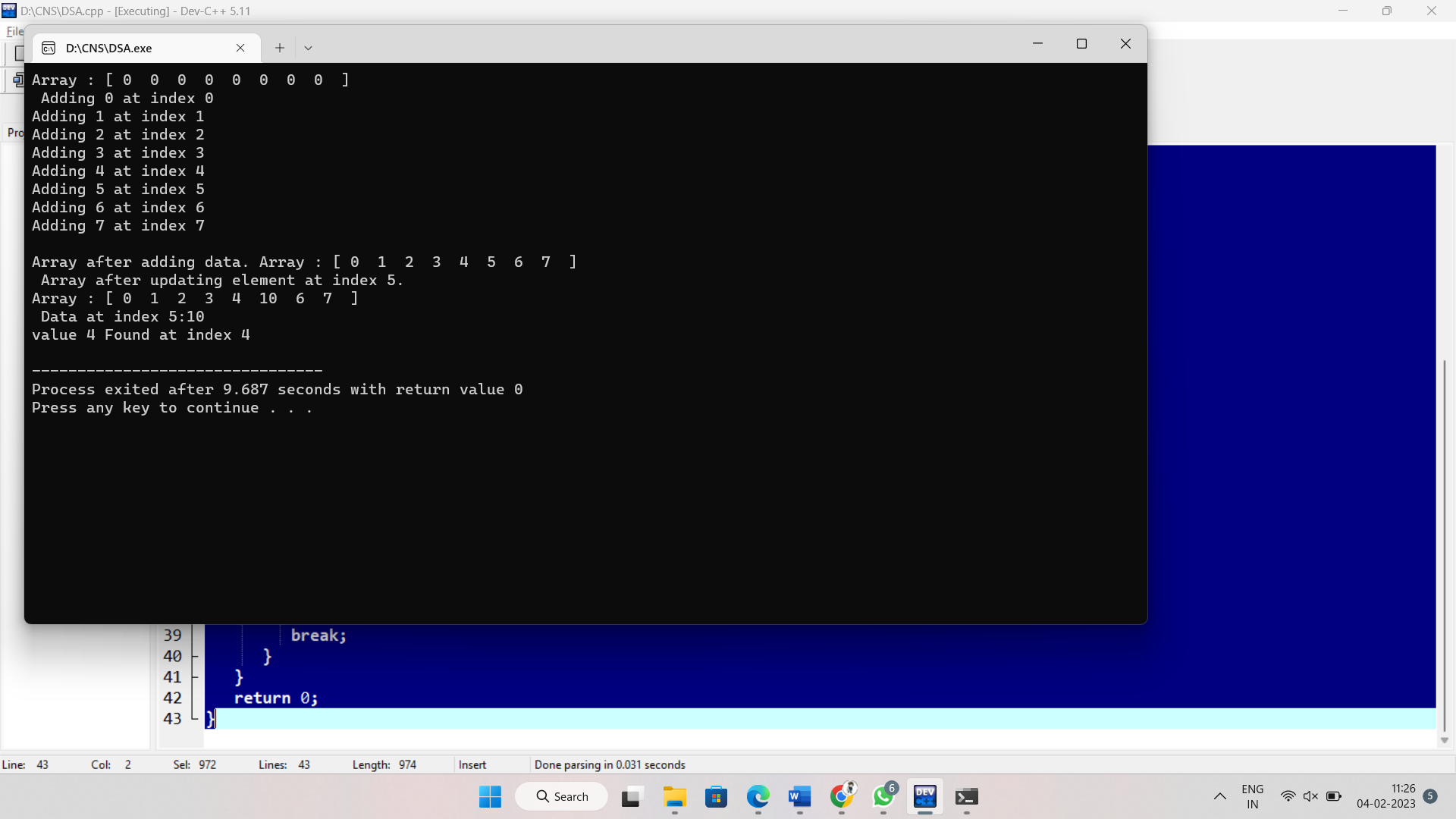
break;

}

}

return 0;

}



10.RSA:

Code:

#include<stdio.h>

#include<stdlib.h>

#include<math.h>

#include<string.h>

long int p,q,n,t,flag,e[100],d[100],temp[100],j,m[100],en[100],i;

char msg[100];

int prime(long int);

void ce();

long int cd(long int);

void encrypt();

void decrypt();

int main()

{

printf("\nENTER FIRST PRIME NUMBER\n");

scanf("%ld",&p);

flag=prime(p);

if(flag==0)

{

printf("\nWRONG INPUT\n");

exit(1);

}

printf("\nENTER ANOTHER PRIME NUMBER\n");

scanf("%ld",&q);

flag=prime(q);

if(flag==0||p==q)

{

printf("\nWRONG INPUT\n");

exit(1);

}

printf("\nENTER MESSAGE\n");

fflush(stdin);

scanf("%s",msg);

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

m[i]=msg[i];

n=p\*q;

t=(p-1)\*(q-1);

ce();

printf("\nPOSSIBLE VALUES OF e AND d ARE\n");

for(i=0;i<j-1;i++)

printf("\n%ld\t%ld",e[i],d[i]);

encrypt();

decrypt();

return 0;

}

int prime(long int pr)

{

int i;

j=sqrt(pr);

for(i=2;i<=j;i++)

{

if(pr%i==0)

return 0;

}

return 1;

}

void ce()

{

int k;

k=0;

for(i=2;i<t;i++)

{

if(t%i==0)

continue;

flag=prime(i);

if(flag==1&&i!=p&&i!=q)

{

e[k]=i; flag=cd(e[k]);

if(flag>0)

{

d[k]=flag;

k++;

}

if(k==99)

break;

}

}

}

long int cd(long int x)

{

long int k=1;

while(1)

{

k=k+t;

if(k%x==0)

return(k/x);

}

}

void encrypt()

{

long int pt,ct,key=e[0],k,len;

i=0;

len=strlen(msg);

while(i!=len)

{

pt=m[i];

pt=pt-96;

k=1;

for(j=0;j<key;j++)

{

k=k\*pt;

k=k%n;

}

temp[i]=k;

ct=k+96;

en[i]=ct;

i++;

}

en[i]=-1;

printf("\nTHE ENCRYPTED MESSAGE IS\n");

for(i=0;en[i]!=-1;i++)

printf("%c",en[i]);

}

void decrypt()

{

long int pt,ct,key=d[0],k;

i=0;

while(en[i]!=-1)

{

ct=temp[i];

k=1;

for(j=0;j<key;j++)

{

k=k\*ct;

k=k%n;

}

pt=k+96;

m[i]=pt;

i++;

}

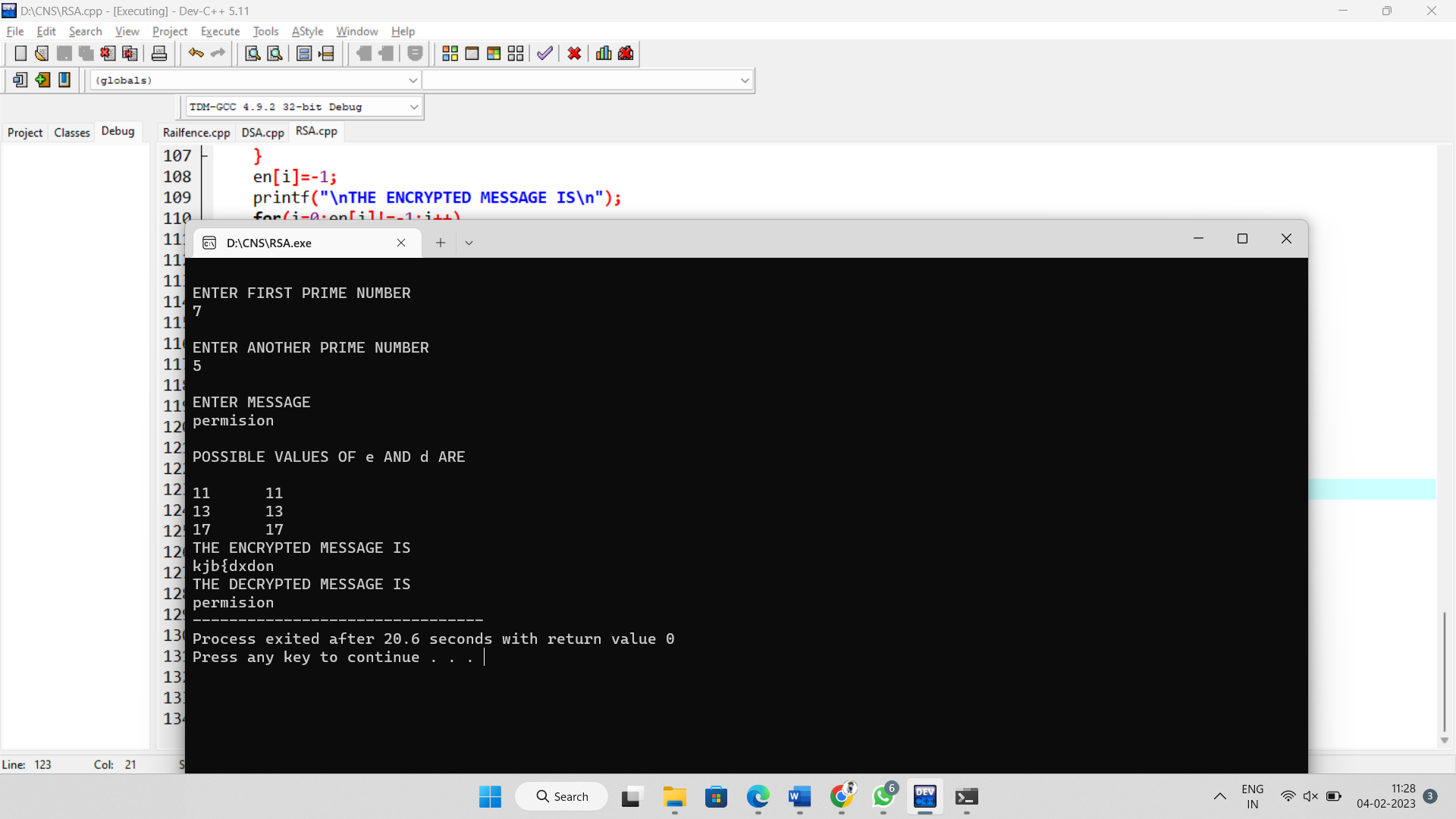
m[i]=-1;

printf("\nTHE DECRYPTED MESSAGE IS\n");

for(i=0;m[i]!=-1;i++)

printf("%c",m[i]);

}



11.Diffie- Hillman algorithm:

Code:

#include <stdio.h>

int compute(int a, int m, int n)

{

int r;

int y = 1;

while (m > 0)

{

r = m % 2;

if (r == 1) {

y = (y\*a) % n;

}

a = a\*a % n;

m = m / 2;

}

return y;

}

int main()

{

int p = 23;

int g = 5;

int a, b;

int A, B;

a = 6;

A = compute(g, a, p);

b = 15;

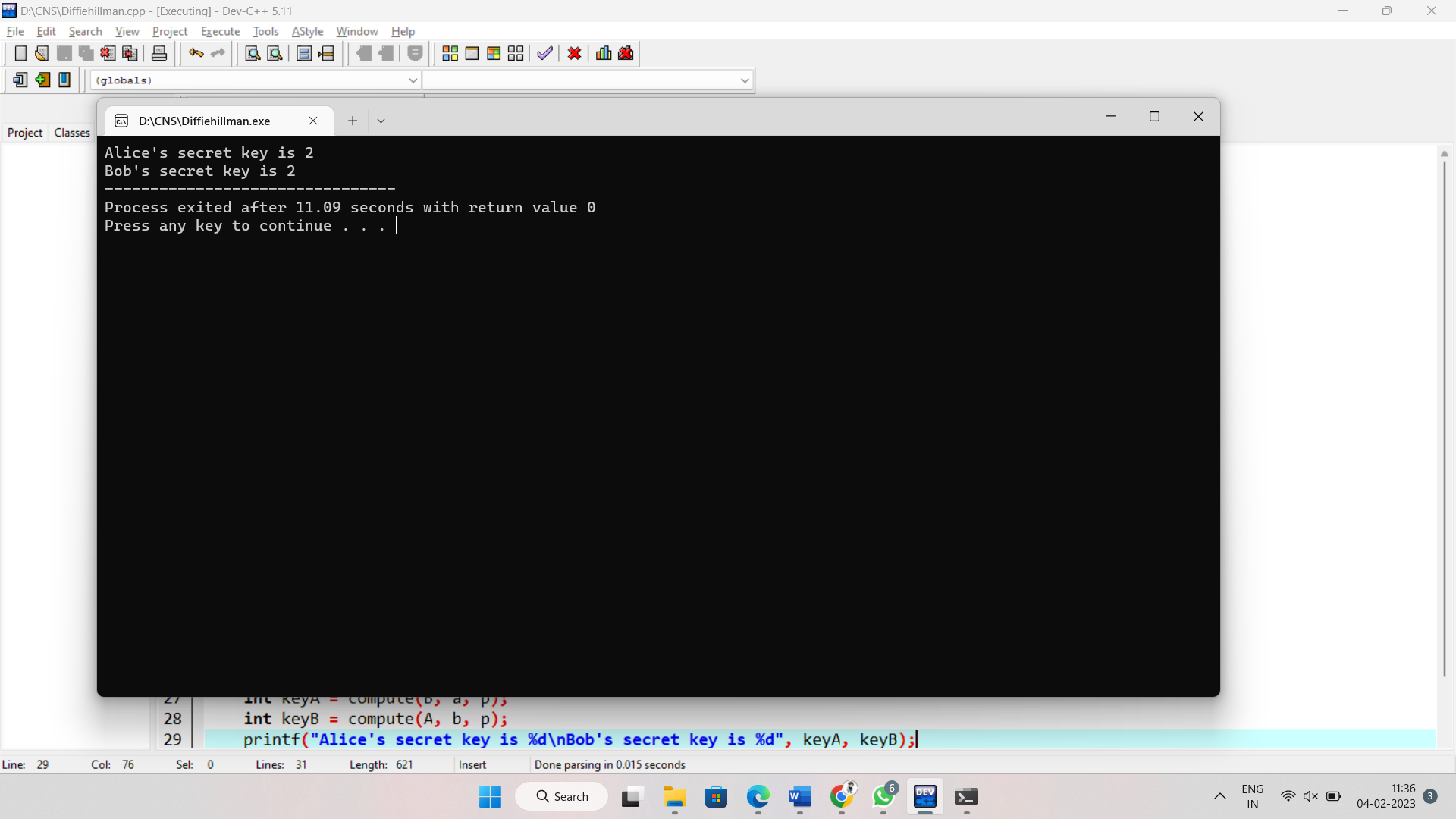
B = compute(g, b, p);

int keyA = compute(B, a, p);

int keyB = compute(A, b, p);

printf("Alice's secret key is %d\nBob's secret key is %d", keyA, keyB);

return 0;

}  


12.DES

Code:

#include <bits/stdc++.h>

using namespace std;

int permChoiceOne[] = {

57, 49, 41, 33, 25, 17, 9 ,

1 , 58, 50, 42, 34, 26, 18,

10, 2 , 59, 51, 43, 35, 27,

19, 11, 3 , 60, 52, 44, 36,

63, 55, 47, 39, 31, 23, 15,

7 , 62, 54, 46, 38, 30, 22,

14, 6 , 61, 53, 45, 37, 29,

21, 13, 5 , 28, 20, 12, 4 };

int permChoiceTwo[] = {

14, 17, 11, 24, 1 , 5 , 3 , 28,

15, 6 , 21, 10, 23, 19, 12, 4 ,

26, 8 , 16, 7 , 27, 20, 13, 2 ,

41, 52, 31, 37, 47, 55, 30, 40,

51, 45, 33, 48, 44, 49, 39, 56,

34, 53, 46, 42, 50, 36, 29, 32 };

int leftShiftTable[] = {1, 1, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1};

string rotateSubKey(string s , int rot)

{

return s.substr(rot, s.length()-rot) + s.substr(0, rot) ;

}

string firstPermute(string input)

{

string res = "" ;

for(int i=0 ; i<56 ; i++)

{

res += input[permChoiceOne[i]-1];

}

return res ;

}

string secondPermute(string input)

{

string res = "" ;

for(int i=0 ; i<48 ; i++)

{

res += input[permChoiceTwo[i]-1];

}

return res ;

}

void genKeys(string left, string right)

{

ofstream fout ;

fout.open("keygen.txt");

for (int i=0; i<16; i++)

{

left = rotateSubKey(left , leftShiftTable[i]);

right = rotateSubKey(right, leftShiftTable[i]);

string key = secondPermute(left+right);

cout << "key " << i+1 << " \t: " << key << endl;

fout << key << endl;

}

}

int main()

{

unsigned long long hexkey;

cout << "\nEnter 64-bit key in hexadecimal(16-digits) : " ;

cin >> hex >> hexkey;

string key = bitset<64>(hexkey).to\_string();

cout << "Binary key (k) \t: " << key << endl;

key = firstPermute(key) ;

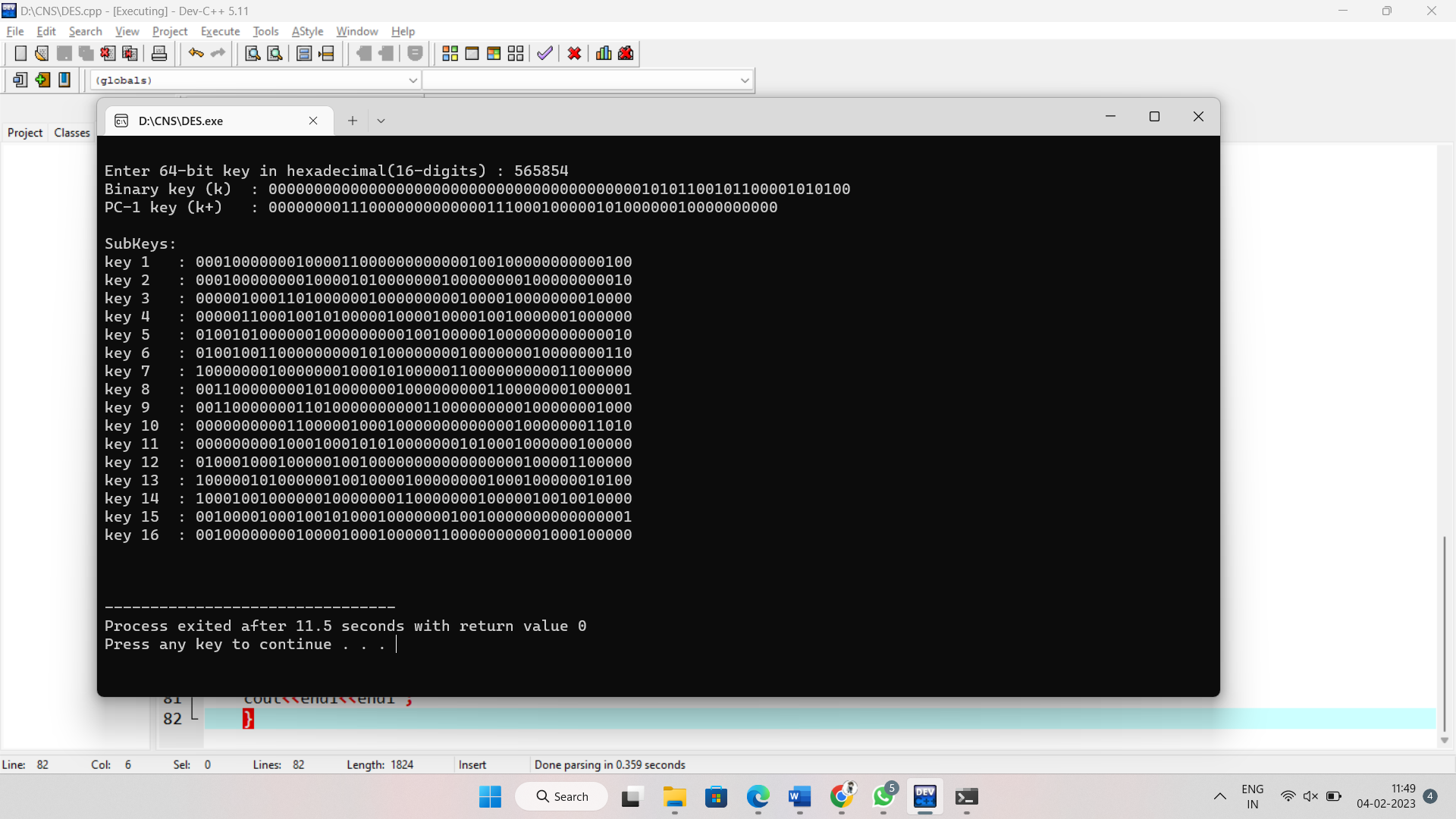
cout << "PC-1 key (k+) \t: " << key << endl;

cout << "\nSubKeys: " << endl;

genKeys(key.substr(0,28) , key.substr(28,28));

cout<<endl<<endl ;

}



13.Affine cipher:

Code:

#include<stdio.h>

#include<string.h>

#include<ctype.h>

#include<stdlib.h>

int CalcGCD(int);

main()

{

int i,j,k,gcd,alpha,beta,numstr[100],numcipher[100];

char str[100],cipher[100];

printf("Enter a string\n");

gets(str);

for(i=0,j=0;i<strlen(str);i++)

{

if(str[i]!=' ')

{

str[j]=toupper(str[i]);

j++;

}

else

{

str[j]=' ';

j++;

}

}

str[j]='\0';

printf("Entered string is : %s \n",str);

printf("Enter Alpha value and must be between 1 and 25 both included\n");

scanf("%d",&alpha);

if(alpha<1 || alpha>25)

{

printf("Alpha should lie in between 1 and 25\nSorry Try again !\n");

exit(0);

}

gcd=CalcGCD(alpha);

if(gcd!=1)

{

printf("gcd(alpha,26)=1 but \n gcd(%d,26)=%d\nSorry Try again !\n",alpha,gcd);

exit(0);

}

printf("Enter Beta value and must be between 0 and 25 both included\n");

scanf("%d",&beta);

if(beta<0 || beta>25)

{

printf("Beta value should lie between 0 and 25\nSorry Try again !\n");

exit(0);

}

for(i=0;i<strlen(str);i++)

{

if(str[i]!=' ')

numstr[i]=str[i]-'A';

else

numstr[i]=-20;

}

printf("Affine Cipher text is\n");

for(i=0;i<strlen(str);i++)

{

if(numstr[i]!=-20)

{

numcipher[i]=((alpha\*numstr[i])+beta)%26;

printf("%c",(numcipher[i]+'A'));

}

else

{

printf(" ");

}

}

printf("\n");

}

int CalcGCD(int alpha)

{

int x;

int temp1=alpha;

int temp2=26;

while(temp2!=0)

{

x=temp2;

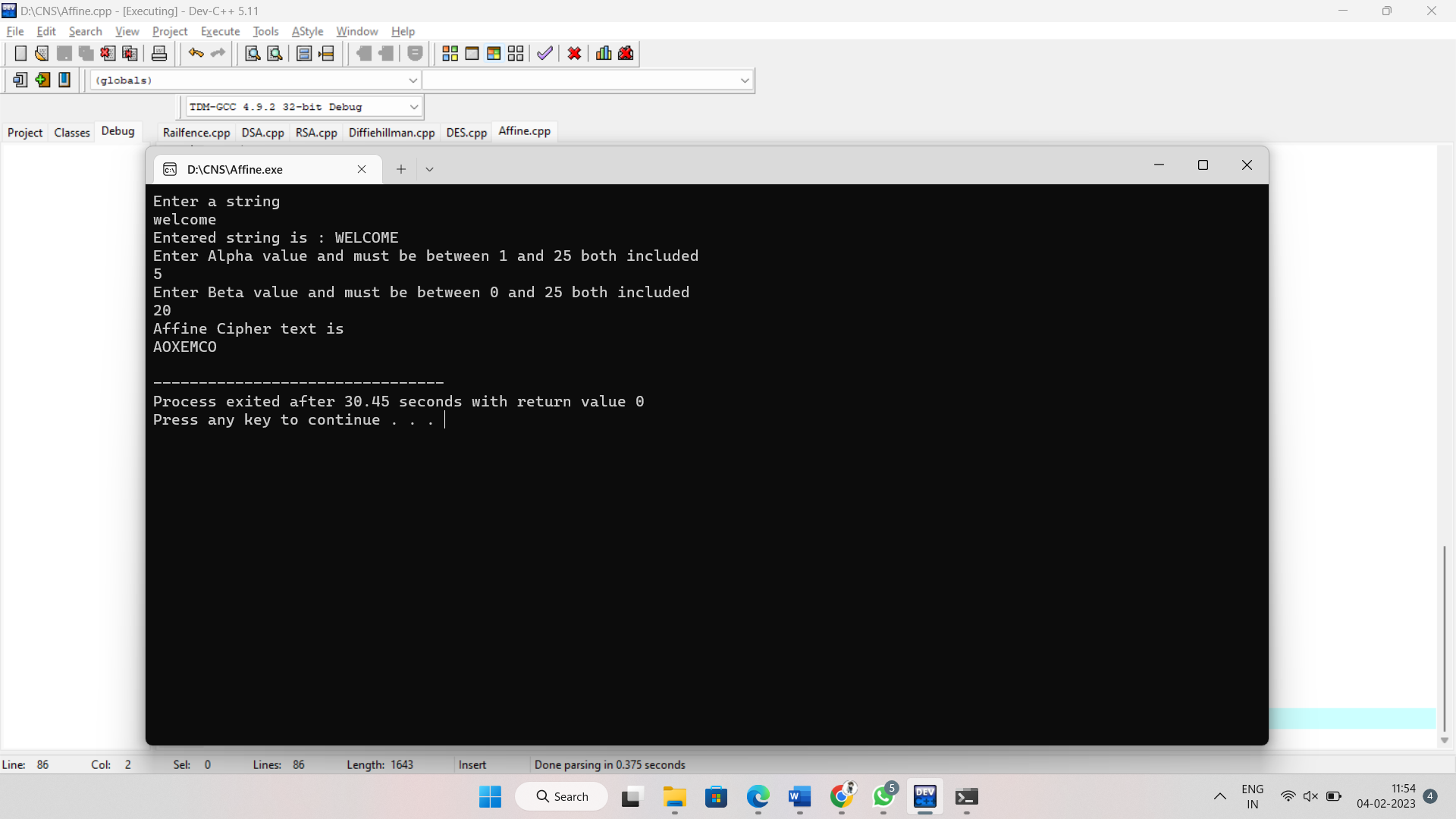
temp2=temp1%temp2;

temp1=x;

}

return(temp1);

}



14.TO convert the plain text:

Code:

#include <stdio.h>

#include <string.h>

int main()

{

char cipherText[] = "53‡‡†305))6\*;4826)4‡.)4‡);806\*;48†8¶60))85;;]8\*;:‡\*8†83(88)5\*†;46(;88\*96\*?;8)\*‡(;485);5\*†2:\*‡(;4956\*2(5\*—4)8¶8\*;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4)485†528806\*81(‡9;48;(88;4(‡?34;48)4‡;161;:188;‡?;";

int length = strlen(cipherText);

char plainText[length + 1];

int i;

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

{

char currentChar = cipherText[i];

switch (currentChar)

{

case '1':

plainText[i] = 'a';

break;

case '2':

plainText[i] = 'b';

break;

case '3':

plainText[i] = 'c';

break;

case '4':

plainText[i] = 'd';

break;

case '5':

plainText[i] = 'e';

break;

case '6':

plainText[i] = 'f';

break;

case '7':

plainText[i] = 'g';

break;

case '8':

plainText[i] = 'h';

break;

case '9':

plainText[i] = 'i';

break;

case '0':

plainText[i] = 'j';

break;

case ';':

plainText[i] = 'k';

break;

case ':':

plainText[i] = 'l';

break;

case '(':

plainText[i] = 'm';

break;

case ')':

plainText[i] = 'n';

break;

case '?':

plainText[i] = 'o';

break;

case '—':

plainText[i] = 'p';

break;

case '†':

plainText[i] = 'q';

break;

case '‡':

plainText[i] = 'r';

break;

case '¶':

plainText[i] = 's';

break;

case '\*':

plainText[i] = 't';

break;

case ']':

plainText[i] = 'u';

break;

case '[':

plainText[i] = 'v';

break;

}

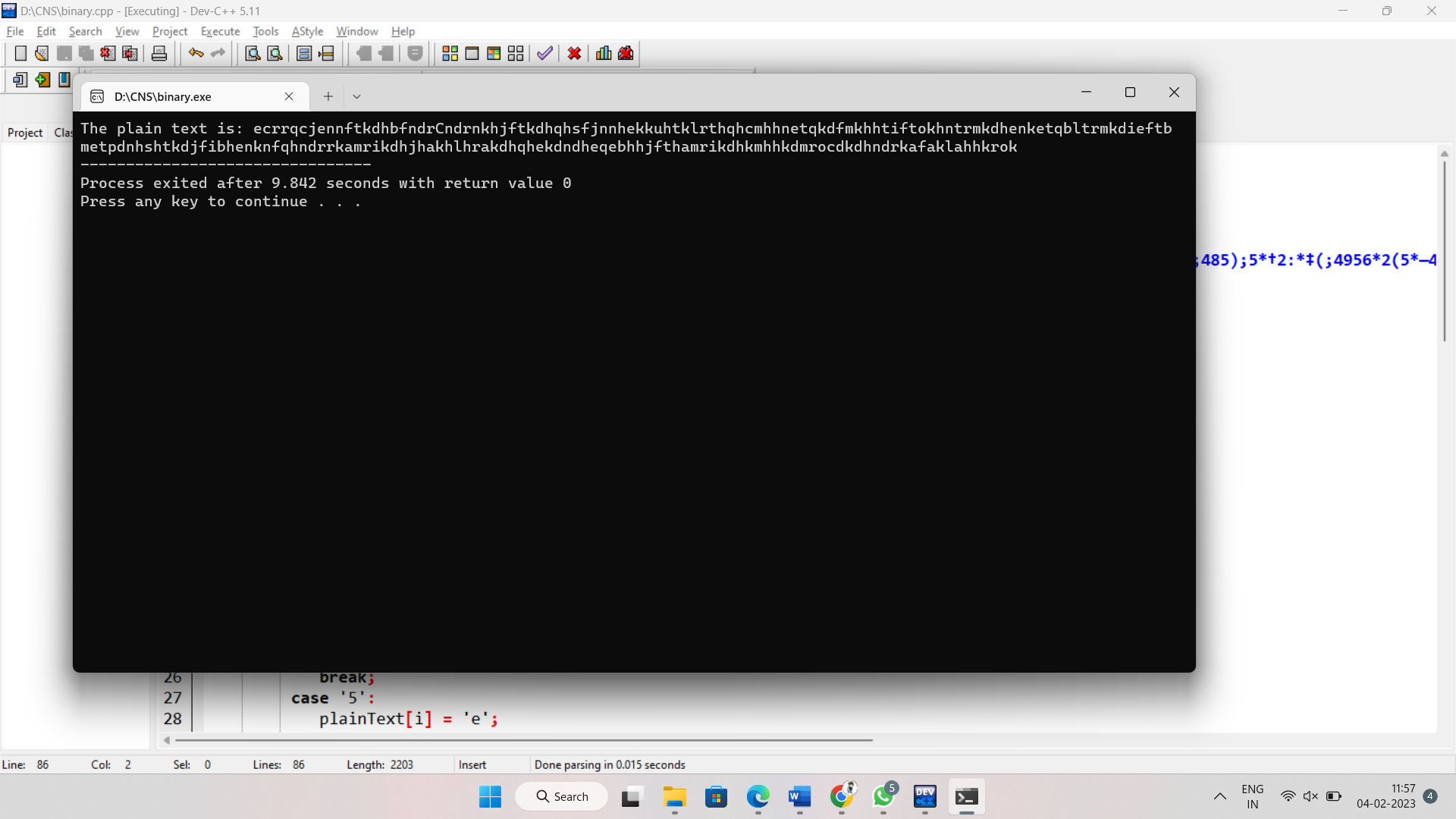
}

plainText[length] = '\0';

printf("The plain text is: %s", plainText);

return 0;

}



15.Converting the word cipher.

Code:

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define SIZE 30

void toLowerCase(char plain[], int ps)

{

int i;

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

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

plain[i] += 32;

}

}

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;

}

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

{

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

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;

}

}

}

}

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;

}

}

}

}

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

int prepare(char str[], int ptrs)

{

if (ptrs % 2 != 0) {

str[ptrs++] = 'z';

str[ptrs] = '\0';

}

return ptrs;

}

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]];

}

}

}

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);

}

int main()

{

char str[SIZE], key[SIZE];

strcpy(key, "CIPHER");

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

strcpy(str, "a b c d e f g h i j k l m n o p q r s t u v w x y z");

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

encryptByPlayfairCipher(str, key);

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

return 0;

}

