**4CS372 : Cryptography and Network Security**

**B.Tech. (CSE) – I [ 2022-23 ]**

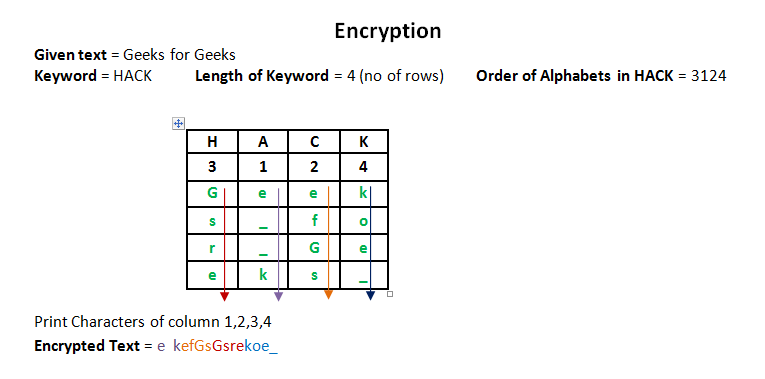
**Assignment No - 5**

**Implementation of Transposition Cipher**

**Aim: Implementation of Transposition Cipher in cpp**

**Theory: Transposition Cipher is another type of encryption technique . In which it contains 2 ciphers**

1. **Railfence Cipher-> We arrange string in zigzag manner .where depth is given .and then read string row by row from cipher string .**
2. **Columnar cipher -> In this cipher 2 strings are given plain and key . where our table length same as key length .After than arrange plain text in table row by row fashion. Then read column by column of table as alphabetical order of key .**



**Code:**

**#include<bits/stdc++.h>**

**using namespace std;**

**string getText(string text)**

**{**

**string x="";**

**for(int i=0;i<text.length();i++)**

**{**

**if(text[i] >='a' && text[i]<='z')**

**x += (text[i]-'a')+'A';**

**else**

**x += text[i];**

**}**

**return x;**

**}**

**string encryption(string text,int key)**

**{**

**char rail[key][(text.length())];**

**text = getText(text);**

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

**{**

**for(int j=0;j<text.length();j++)**

**rail[i][j] = '^';**

**}**

**bool dir\_down = false;**

**int row=0,col=0;**

**for(int i=0;i<text.length();i++)**

**{**

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

**dir\_down = !dir\_down;**

**rail[row][col++] = text[i];**

**dir\_down ? row++ : row--;**

**}**

**string result;**

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

**{**

**for(int j=0;j<text.length();j++)**

**{**

**if(rail[i][j] != '^')**

**result.push\_back(rail[i][j]);**

**}**

**}**

**return result;**

**}**

**string decryption(string text,int key)**

**{**

**char rail[key][(text.length())];**

**text = getText(text);**

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

**{**

**for(int j=0;j<text.length();j++)**

**rail[i][j] = '\n';**

**}**

**bool dir\_down;**

**int row=0,col=0;**

**for(int i=0;i<text.length();i++)**

**{**

**if(row==0)**

**dir\_down = true;**

**if(row == key-1)**

**dir\_down = false;**

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

**dir\_down ? row++ : row--;**

**}**

**int index=0;**

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

**{**

**for(int j=0;j<text.length();j++)**

**{**

**if(rail[i][j] == '\*' && index<text.length())**

**rail[i][j] = text[index++];**

**}**

**}**

**string result;**

**row = 0;**

**col = 0;**

**for(int i=0;i<text.length();i++)**

**{**

**if(row == 0)**

**dir\_down = true;**

**if(row == key-1)**

**dir\_down = false;**

**if(rail[row][col] != '\*')**

**result.push\_back(rail[row][col++]);**

**dir\_down ? row++ : row--;**

**}**

**return result;**

**}**

**int main()**

**{**

**int choice;**

**int datachoice;**

**string sample;**

**int key;**

**while(1)**

**{**

**cout << "Rail Fence Cipher\n 1. Encryption \n 2. Decryption\n 3. Exit\nEnter Choice: ";**

**cin>>choice;**

**if(choice>2)**

**break;**

**switch(choice)**

**{**

**case 1:**

**cout << "Data is from\n 1. Manual Entering \n 2. File \nEnter Choice: ";**

**cin>>datachoice;**

**if(datachoice == 1)**

**{**

**cout<<"Enter data to be Encrypted:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**cin>>key;**

**cout<<"Encrypted String:\n";**

**cout<<encryption(sample,key)<<endl;**

**}**

**else**

**{**

**cout<<"Enter File Name:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**cin>>key;**

**fstream myfile;**

**myfile.open(sample.c\_str());**

**string str,s;**

**if(!myfile.is\_open())**

**cout << "Error while Opening File";**

**while(getline(myfile,str))**

**s+=str;**

**myfile.close();**

**s=encryption(s,key);**

**myfile.open("CipherText.txt",ios\_base::out);**

**if(myfile.is\_open())**

**myfile.write(s.data(),s.size());**

**cout<<"File Encrypted\n";**

**myfile.close();**

**}**

**break;**

**case 2:**

**cout << "Data is from\n 1. Manual Entering \n 2. File \nEnter Choice: ";**

**cin>>datachoice;**

**if(datachoice == 1)**

**{**

**cout<<"Enter data to be Decrypted:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**cin>>key;**

**cout<<"Decrypted String:\n";**

**cout<<decryption(sample,key)<<endl;;**

**}**

**else**

**{**

**cout<<"Enter File Name:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**cin>>key;**

**fstream myfile;**

**myfile.open(sample.c\_str());**

**string str,s;**

**if(!myfile.is\_open())**

**cout << "Error while Opening File";**

**while(getline(myfile,str))**

**s+=str;**

**myfile.close();**

**s=decryption(s,key);**

**myfile.open("PlainText.txt",ios\_base::out);**

**if(myfile.is\_open())**

**myfile.write(s.data(),s.size());**

**cout<<"File Decrypted\n";**

**myfile.close();**

**}**

**break;**

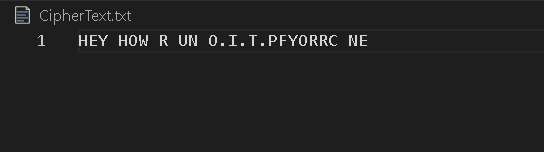
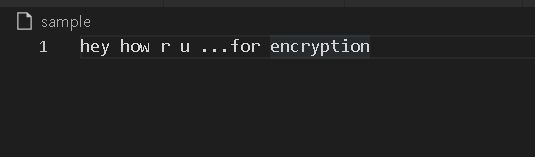
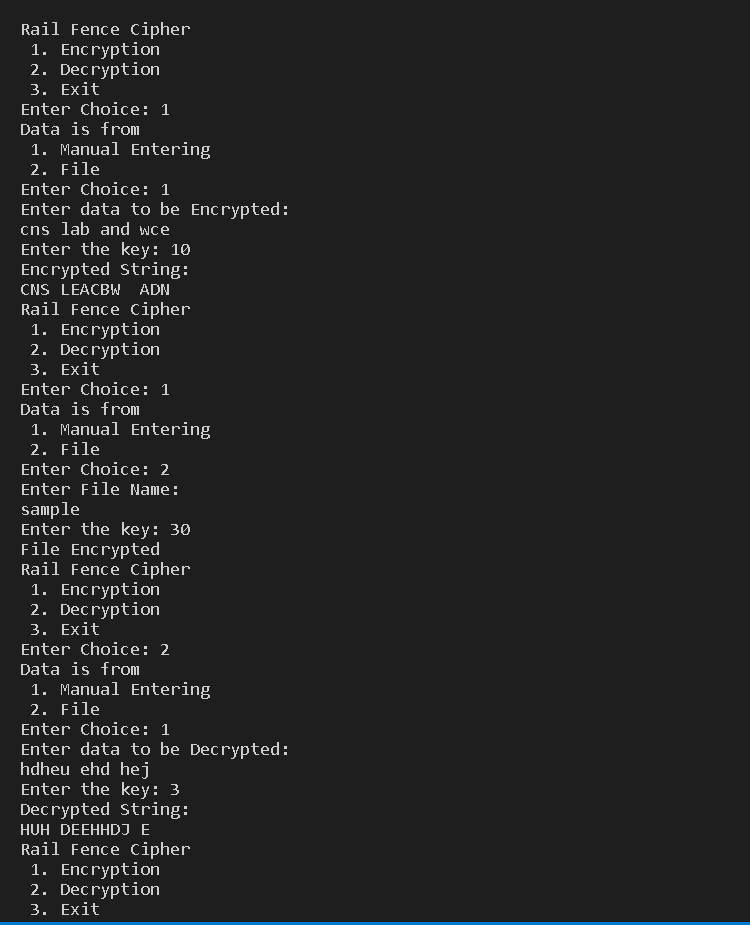
**}**

**}**

**return 0;**

**}**

**Output:**

****

**Implementation of Columnar Cipher**

**Code:**

**#include<bits/stdc++.h>**

**using namespace std;**

**map<int,int> keyMap;**

**string getText(string text)**

**{**

**string x="";**

**for(int i=0;i<text.length();i++)**

**{**

**if(text[i] >='a' && text[i]<='z')**

**x += (text[i]-'a')+'A';**

**else**

**x += text[i];**

**}**

**return x;**

**}**

**void setPermutationOrder(string key)**

**{**

**keyMap.clear();**

**for(int i=0;i<key.length();i++)**

**keyMap[key[i]] = i;**

**}**

**string encryption(string msg,string key)**

**{**

**key = getText(key);**

**setPermutationOrder(key);**

**msg = getText(msg);**

**int row,col,j;**

**string cipher="";**

**col = key.length();**

**row = msg.length()/col;**

**if(msg.length() % col)**

**row += 1;**

**char matrix[row][col];**

**for(int i=0,k=0;i<row;i++)**

**{**

**for(int j=0;j<col;)**

**{**

**if(msg[k] == '\0')**

**{**

**matrix[i][j] = '\_';**

**j++;**

**}**

**if(isalpha(msg[k]) || msg[k] == ' ')**

**{**

**matrix[i][j] = msg[k];**

**j++;**

**}**

**k++;**

**}**

**}**

**for(map<int,int>::iterator ii = keyMap.begin(); ii != keyMap.end(); ++ii)**

**{**

**j = ii->second;**

**for(int i=0;i<row;i++)**

**{**

**if(isalpha(matrix[i][j]) || matrix[i][j] == ' ' || matrix[i][j] == '\_')**

**cipher += matrix[i][j];**

**}**

**}**

**return cipher;**

**}**

**string decryption(string cipher,string key)**

**{**

**key = getText(key);**

**setPermutationOrder(key);**

**cipher = getText(cipher);**

**int col = key.length();**

**int row = cipher.length()/col;**

**char cipherMatrix[row][col];**

**for(int j=0,k=0;j<col;j++)**

**{**

**for(int i=0;i<row;i++)**

**cipherMatrix[i][j] = cipher[k++];**

**}**

**int index=0;**

**for(map<int,int>::iterator ii = keyMap.begin(); ii != keyMap.end(); ++ii)**

**{**

**ii->second = index++;**

**}**

**char decCipher[row][col];**

**map<int,int>::iterator ii = keyMap.begin();**

**int k=0;**

**for(int l=0,j; key[l] != '\0'; k++)**

**{**

**j = keyMap[key[l++]];**

**for(int i = 0; i < row; i++)**

**decCipher[i][k] = cipherMatrix[i][j];**

**}**

**string msg="";**

**for(int i=0; i<row; i++)**

**{**

**for(int j=0; j<col; j++)**

**{**

**if(decCipher[i][j] != '\_')**

**msg += decCipher[i][j];**

**}**

**}**

**return msg;**

**}**

**int main()**

**{**

**int choice;**

**int datachoice;**

**string sample,key;**

**int shift;**

**while(1)**

**{**

**cout << "Columnar Cipher\n 1. Encryption \n 2. Decryption\n 3. Exit\nEnter Choice: ";**

**cin>>choice;**

**if(choice>2)**

**break;**

**switch(choice)**

**{**

**case 1:**

**cout << "Data is from\n 1. Manual Entering \n 2. File \nEnter Choice: ";**

**cin>>datachoice;**

**if(datachoice == 1)**

**{**

**cout<<"Enter data to be Encrypted:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**getline(cin,key);**

**cout<<"Encrypted String:\n";**

**cout<<encryption(sample,key)<<endl;**

**}**

**else**

**{**

**cout<<"Enter File Name:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**getline(cin,key);**

**fstream myfile;**

**myfile.open(sample.c\_str());**

**string str,s;**

**if(!myfile.is\_open())**

**cout << "Error while Opening File";**

**if(getline(myfile,str))**

**{**

**while(1)**

**{**

**s+=str;**

**if(getline(myfile,str))**

**s+=" ";**

**else**

**break;**

**}**

**myfile.close();**

**s=encryption(s,key);**

**myfile.open("CipherText1.txt",ios\_base::out);**

**if(myfile.is\_open())**

**myfile.write(s.data(),s.size());**

**cout<<"File Encrypted\n";**

**myfile.close();**

**}**

**else**

**{**

**myfile.close();**

**cout<<"Noothing in the File"<<endl;**

**}**

**}**

**break;**

**case 2:**

**cout << "Data is from\n 1. Manual Entering \n 2. File \nEnter Choice: ";**

**cin>>datachoice;**

**if(datachoice == 1)**

**{**

**cout<<"Enter data to be Decrypted:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**getline(cin,key);**

**cout<<"Decrypted String:\n";**

**cout<<decryption(sample,key)<<endl;;**

**}**

**else**

**{**

**cout<<"Enter File Name:\n";**

**cin.ignore();**

**getline(cin,sample);**

**cout<<"Enter the key: ";**

**getline(cin,key);**

**fstream myfile;**

**myfile.open(sample.c\_str());**

**string str,s;**

**if(!myfile.is\_open())**

**cout << "Error while Opening File";**

**while(getline(myfile,str))**

**s+=(str+" ");**

**myfile.close();**

**s=decryption(s,key);**

**myfile.open("PlainText1.txt",ios\_base::out);**

**if(myfile.is\_open())**

**myfile.write(s.data(),s.size());**

**cout<<"File Decrypted\n";**

**myfile.close();**

**}**

**break;**

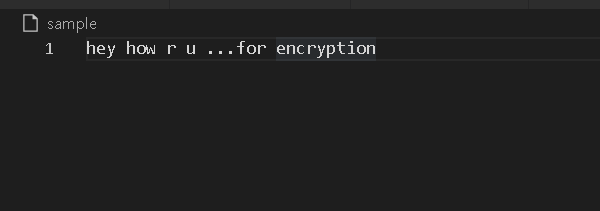
**}**

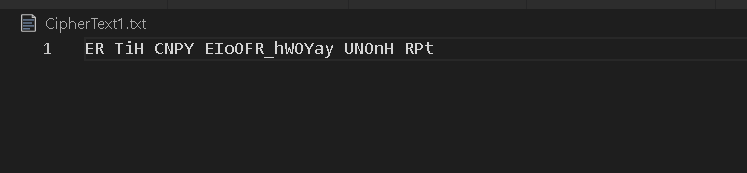
**}**

**return 0;**

**}**

**Output:**

****

****

**Conclusion:**

**So these transposition cipher are complicated than substitution ones and more secure.**