**Experiment no: 02 Date: /0 /19**

**TRANSPOSITION**

**Aim:** Write a program to implement Transposition.

**Theory:**

In cryptography, a **transposition cipher** is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed (the plaintext is reordered). Mathematically a bijective function is used on the characters' positions to encrypt and an inverse function to decrypt.

**Columnar transposition**

The columnar transposition cipher is a fairly simple, easy to implement cipher. It is a transposition cipher that follows a simple rule for mixing up the characters in the plaintext to form the ciphertext.

Although weak on its own, it can be combined with other ciphers, such as a substitution cipher, the combination of which can be more difficult to break than either cipher on it's own.

## Example

To encrypt a piece of text, e.g.

defend the east wall of the castle

taking a 6X6 matrix

d e f e n d

t h e e a s

t w a l l o

f t h e c a

s t l e x x

In the above example, the plaintext has been padded so that it neatly fits in a rectangle. This is known as a regular columnar transposition. An irregular columnar transposition leaves these characters blank, though this makes decryption slightly more difficult. The columns are now reordered such that the letters in the key word are ordered alphabetically.

n e d e d f

a h t e s e

l w t l o a

c t f e a h

x t s e x l

The ciphertext is read off along the columns:

nalcxehwttdttfseeleedsoaxfeahl

**Algorithm:**

1. Start.
2. Declare String message and ct (cipher text).
3. Input message
4. Store k = length ( message ).
5. Start encoding:
   1. Print message
   2. I = 0
   3. Repeat step 5:d to 5:h until I < 5
   4. J = i
   5. Repeat step 5:f to 5:g until j < k
   6. Ct = ct + message [ j ]
   7. J = j + 5
   8. I + +
6. Print ct
7. Start decoding:
   1. Pad empty places in matrix with ‘\0’
      1. I = length ( ct )
      2. Repeat 7:a:iii to 7:a:iv until I < 25
      3. Message [ i ] = ‘ \0 ’
      4. I + +
   2. X = 0
   3. I = 0
   4. Repeat 7:e to 7: i until I < 5
   5. J = i
   6. Repeat 7:h until j < 25
   7. If ( message [ j ] ! = ‘ \0 ‘ ) message [ j ] = ct [ x ] , X + +
   8. J = j + 5
   9. I + +
8. Print message.
9. Stop.

**Complexity:**

Encoding:

1. for( int I = 0; I < 5; I ++ )
2. for( int j = i; j < k; j + = 5 )
3. ct = ct + message [ j ];

Message is converted into a matrix of 5x5 dimension using two for loops and ct holds ct + the message at that position (3). So, O(n2).

Decoding:

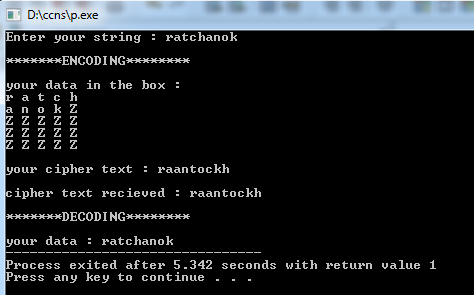
1. for( int I = ct. length ( ); I < 25; I ++ )
2. for( int I = 0; I < 5; I ++ )
3. for( int j = i; j < 25; j + = 5 )

For decoding, matrix is filled with '\0' if size of cipher is less than 25 (1). And using two for loops (2&3) for storing the cipher text column wise which gives original message. O(n2)

**Program:**

|  |
| --- |
| #include<iostream>  #include<string.h>  using namespace std;  class TranspositionAlgorithm  { string message,ct;  public:  void encode()  { int k;  cout<<"Enter your string : ";  cin>>message;  k=message.length();  cout<<"\n\*\*\*\*\*\*\*ENCODING\*\*\*\*\*\*\*\*\n";    cout<<"\nyour data in the box : \n";  print(message);      for(int i=0;i<5;i++) //cipher text is generated here  for(int j=i;j<k;j+=5)  ct=ct+message[j];    cout<<"\nyour cipher text : ";  cout<<ct;  }    void decode()  { cout<<"\n\ncipher text recieved : "<<ct;  cout<<"\n\n\*\*\*\*\*\*\*DECODING\*\*\*\*\*\*\*\*\n";  for(int i=ct.length();i<25;i++)  message[i]='\0'; //matrix is filled with '\0' if size of cipher is less than 25    int x=0;  for(int i=0;i<5;i++)  for(int j=i;j<25;j+=5)  if(message[j]!='\0')  { message[j]=ct[x]; //storing the cipher text column wise  x++;  }    cout<<"\nyour data : ";  cout<<message;  }      //function to print the message in the matrix form  void print(string data)  {  for(int i=0;i<25;i+=5)  { for(int j=i;j<i+5;j++)  if(j<data.length())  cout<<data[j]<<" ";  else  cout<<"Z ";  cout<<endl;  }  }  };  int main()  { TranspositionAlgorithm a;  a.encode();  a.decode();  return 1;  } |

**Result:**

****

**Conclusion:** Program to implementTransposition was written and executed successfully.