21.08.2020

Ex No: 2.A

IMPLEMENTATION OF RAIL FENCE – ROW & COLUMN

AIM:

To write a C program to implement the rail fence transposition technique.

DESCRIPTION:

In the rail fence cipher, the plain text is written downwards and diagonally on successive "rails" of an imaginary fence, then moving up when we reach the bottom rail. When we reach the top rail, the message is written downwards again until the whole plaintext is written out. The message is then read off in rows.

ALGORITHM:

STEP-1: Read the Plain text.

STEP-2: Arrange the plain text in row columnar matrix format.

STEP-3: Now read the keyword depending on the number of columns of the plain text.

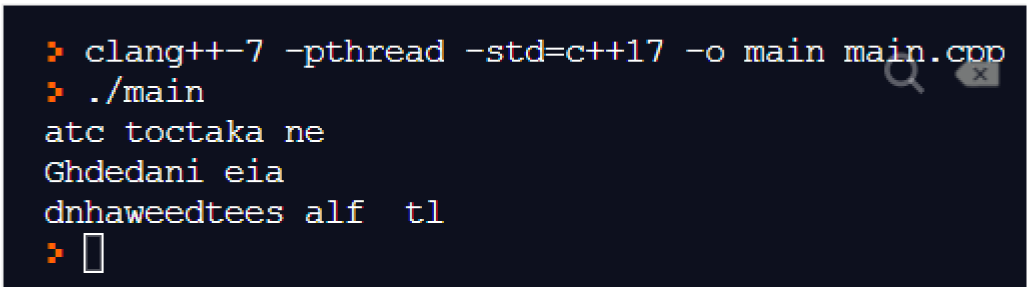
STEP-4: Arrange the characters of the keyword in sorted order and the corresponding columns of the plain text.

STEP-5: Read the characters row wise or column wise in the former order to get the cipher text.

PROGRAM: (Rail Fence)

```
#include <bits/stdc++.h>
using namespace std;

// function to encrypt a message
string encryptRailFence(string text, int key)
{
    // create the matrix to cipher plain text
    // key = rows , length(text) = columns
    char rail[key][(text.length())];

    // filling the rail matrix to distinguish filled
    // spaces from blank ones
    for (int i=0; i < key; i++)
        for (int j = 0; j < text.length(); j++)
            rail[i][j] = '\n';

    // to find the direction
    bool dir_down = false;
    int row = 0, col = 0;

    for (int i=0; i < text.length(); i++)
    {
        // check the direction of flow
        // reverse the direction if we've just
        // filled the top or bottom rail
        if (row == 0 || row == key-1)
            dir_down = !dir_down;

        // fill the corresponding alphabet
        rail[row][col++] = text[i];

        // find the next row using direction flag
        dir_down?row++ : row--;
    }

    //now we can construct the cipher using the rail matrix
    string result;
    for (int i=0; i < key; i++)
        for (int j=0; j < text.length(); j++)
            if (rail[i][j]!='\n')
                result.push_back(rail[i][j]);

    return result;
}

// This function receives cipher-text and key
// and returns the original text after decryption
string decryptRailFence(string cipher, int key)
{
    // create the matrix to cipher plain text
    // key = rows , length(text) = columns
    char rail[key][cipher.length()];

    // filling the rail matrix to distinguish filled
    // spaces from blank ones
    for (int i=0; i < key; i++)
        for (int j=0; j < cipher.length(); j++)
            rail[i][j] = '\n';

    // to find the direction
    bool dir_down;

    int row = 0, col = 0;

    // mark the places with '*'
    for (int i=0; i < cipher.length(); i++)
    {
        // check the direction of flow
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;

        // place the marker
        rail[row][col++] = '*';

        // find the next row using direction flag
        dir_down?row++ : row--;
    }

    // now we can construct the fill the rail matrix
    int index = 0;
    for (int i=0; i<key; i++)
        for (int j=0; j<cipher.length(); j++)
            if (rail[i][j] == '*' && index<cipher.length())
                rail[i][j] = cipher[index++];


    // now read the matrix in zig-zag manner to construct
    // the resultant text
    string result;

    row = 0, col = 0;
    for (int i=0; i< cipher.length(); i++)
    {
        // check the direction of flow
        if (row == 0)
            dir_down = true;
        if (row == key-1)
            dir_down = false;

        // place the marker
        if (rail[row][col] != '*')
            result.push_back(rail[row][col++]);

        // find the next row using direction flag
        dir_down?row++: row--;
    }
    return result;
}

//driver program to check the above functions
int main()
{
    cout << encryptRailFence("attack at once", 2) << endl;
    cout << encryptRailFence("Geedhaindia ", 3) << endl;
    cout << encryptRailFence("defend the east wall", 3) << endl;



    return 0;
}
```

OUTPUT:



```
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
atc toctaka ne
Ghdedani eia
dnhaweedtees alf  tl
>
```

RESULT:

Thus the rail fence algorithm had been executed successfully.

.