

# PlayFair ALgorthem

## IT4 Ibb UN

### Eng:Eissa AL-gumaei

Form1

PlainText

KeyText

Encrypt

Decrypt

CipherText

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace playFairAlgorthem
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```

}
private void btnEncrypt_Click(object sender, EventArgs e)
{
    string plaintext = txtInput.Text.ToUpper().Replace("J", "I").Replace(" ",
    "");
    string keyword = txtKey.Text.ToUpper().Replace("J", "I").Replace(" ",
    "");
    string ciphertext = EncryptPlayfair(plaintext, keyword);
    txtOutput.Text = ciphertext.Replace("X", "");
}

private void btnDecrypt_Click(object sender, EventArgs e)
{
    string ciphertext = txtInput.Text.ToUpper().Replace(" ", "");
    string keyword = txtKey.Text.ToUpper().Replace("J", "I").Replace(" ",
    "");
    string decryptedText = DecryptPlayfair(ciphertext, keyword);
    txtOutput.Text = decryptedText;
}

private string EncryptPlayfair(string plaintext, string keyword)
{
    char[,] matrix = GenerateMatrix(keyword);
    return ProcessText(plaintext, matrix, true);
}

private string DecryptPlayfair(string ciphertext, string keyword)
{
    char[,] matrix = GenerateMatrix(keyword);
    return ProcessText(ciphertext, matrix, false);
}

private string ProcessText(string text, char[,] matrix, bool encrypt)
{
    string result = "";

    for (int i = 0; i < text.Length; i += 2)
    {
        if (i + 1 >= text.Length || text[i] == text[i + 1])
        {
            text = text.Insert(i + 1, "X");
        }

        char a = text[i];
        char b = text[i + 1];

        (int row1, int col1) = FindPosition(matrix, a);
        (int row2, int col2) = FindPosition(matrix, b);
    }
}

```

```

        if (row1 == row2)
        {
            result += matrix[row1, (col1 + (encrypt ? 1 : 4)) % 5];
            result += matrix[row2, (col2 + (encrypt ? 1 : 4)) % 5];
        }
        else if (col1 == col2)
        {
            result += matrix[(row1 + (encrypt ? 1 : 4)) % 5, col1];
            result += matrix[(row2 + (encrypt ? 1 : 4)) % 5, col2];
        }
        else
        {
            result += matrix[row1, col2];
            result += matrix[row2, col1];
        }
    }

    return result;
}

private char[,] GenerateMatrix(string keyword)
{
    char[,] matrix = new char[5, 5];
    string usedChars = "";

    foreach (char c in keyword)
    {
        if (!usedChars.Contains(c) && c != 'J')
        {
            usedChars += c;
        }
    }

    string alphabet = "ABCDEFGHIKLMNOPQRSTUVWXYZ";
    foreach (char c in alphabet)
    {
        if (!usedChars.Contains(c))
        {
            usedChars += c;
        }
    }

    int index = 0;
    for (int row = 0; row < 5; row++)
    {
        for (int col = 0; col < 5; col++)
        {
            matrix[row, col] = usedChars[index++];
        }
    }
}

```

```
    }  
    return matrix;  
}  
  
private (int, int) FindPosition(char[,] matrix, char c)  
{  
    for (int row = 0; row < 5; row++)  
    {  
        for (int col = 0; col < 5; col++)  
        {  
            if (matrix[row, col] == c)  
            {  
                return (row, col);  
            }  
        }  
    }  
    return (-1, -1);  
}  
}
```