

# PlayFair Cipher

Playfair Cipher using with different key values

**Name : GANESH D**

**RegNo: 212223240035**

## AIM:

To implement a program to encrypt a plain text and decrypt a cipher text using play fair Cipher substitution technique.

## DESIGN STEPS:

### Step 1:

Design of PlayFair Cipher algorithm

### Step 2:

Implementation using C or python code

### Step 3:

Testing algorithm with different key values.

**ALGORITHM DESCRIPTION:** The Playfair cipher uses a 5 by 5 table containing a key word or phrase. To generate the key table, first fill the spaces in the table with the letters of the keyword, then fill the remaining spaces with the rest of the letters of the alphabet in order (usually omitting "Q" to reduce the alphabet to fit; other versions put both "I" and "J" in the same space). The key can be written in the top rows of the table, from left to right, or in some other pattern, such as a spiral beginning in the upper-left-hand corner and ending in the centre. The keyword together with the conventions for filling in the 5 by 5 table constitutes the cipher key. To encrypt a message, one would break the message into digrams (groups of 2 letters) such that, for example, "HelloWorld" becomes "HE LL OW OR LD", and map them out on the key table. Then apply the following 4 rules, to each pair of letters in the plaintext:

1. If both letters are the same (or only one letter is left), add an "X" after the first letter. Encrypt the new pair and continue. Some variants of Playfair use "Q" instead of "X", but any letter, itself uncommon as a repeated pair, will do.
2. If the letters appear on the same row of your table, replace them with the letters to their immediate right respectively (wrapping around to the left side of the row if a letter in the original pair was on the right side of the row).
3. If the letters appear on the same column of your table, replace them with the letters immediately below respectively (wrapping around to the top side of the column if a letter in the original pair was on the bottom side of the column).
4. If the letters are not on the same row or column, replace them with the letters on the same row respectively but at the other pair of corners of the rectangle defined by the original pair. The order is important – the first letter of the encrypted pair is the one that lies on the same row as the first letter of the plaintext pair. To decrypt, use the INVERSE (opposite) of the last 3 rules, and the 1st as-is (dropping any extra "X"s, or "Q"s that do not make sense in the final message when finished).

## PROGRAM:



```
#include <stdio.h>
#include <string.h>
#include <ctype.h>

#define MX 5

void playfair(char ch1, char ch2, char key[MX][MX]) {
    int i, j, w, x, y, z;

    for (i = 0; i < MX; i++) {
        for (j = 0; j < MX; j++) {
            if (ch1 == key[i][j]) {
                w = i;
                x = j;
            }
            if (ch2 == key[i][j]) {
                y = i;
                z = j;
            }
        }
    }

    if (w == y) {
        x = (x + 1) % 5;
        z = (z + 1) % 5;
    } else if (x == z) {
        w = (w + 1) % 5;
        y = (y + 1) % 5;
    } else {
        int temp = x;
        x = z;
        z = temp;
    }

    printf("%c%c", key[w][x], key[y][z]); // Print directly
}

void remove_duplicates(char *str) {
    int hash[26] = {0}, i, j = 0;
    for (i = 0; str[i]; i++) {
        if (!hash[str[i] - 'A']) {
            str[j++] = str[i];
            hash[str[i] - 'A'] = 1;
        }
    }
    str[j] = '\0';
}

void prepare_input(char *str) {
    int i, j = 0;
    for (i = 0; str[i]; i++) {
        if (isalpha(str[i])) {
            str[j++] = toupper(str[i] == 'J' ? 'I' : str[i]);
        }
    }
}
```

```

    }
}
str[j] = '\0';
}

void generate_key_matrix(char key[MX][MX], char *keyst) {
    char alphabet[26] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
    char temp[26] = {0};
    int i, j, k = 0;

    strcpy(temp, keyst);
    remove_duplicates(temp);

    for (i = 0; i < 26; i++) {
        if (!strchr(temp, alphabet[i])) {
            temp[strlen(temp)] = alphabet[i];
        }
    }

    k = 0;
    for (i = 0; i < MX; i++) {
        for (j = 0; j < MX; j++) {
            key[i][j] = temp[k++];
            printf("%c ", key[i][j]);
        }
        printf("\n");
    }
}

void encrypt(char *str, char key[MX][MX]) {
    printf("\nCipher Text: ");
    int i;
    for (i = 0; str[i]; i++) {
        if (str[i + 1] == '\0')
            playfair(str[i], 'X', key);
        else {
            if (str[i] == str[i + 1]) {
                playfair(str[i], 'X', key);
            } else {
                playfair(str[i], str[i + 1], key);
                i++;
            }
        }
    }
    printf("\n");
}

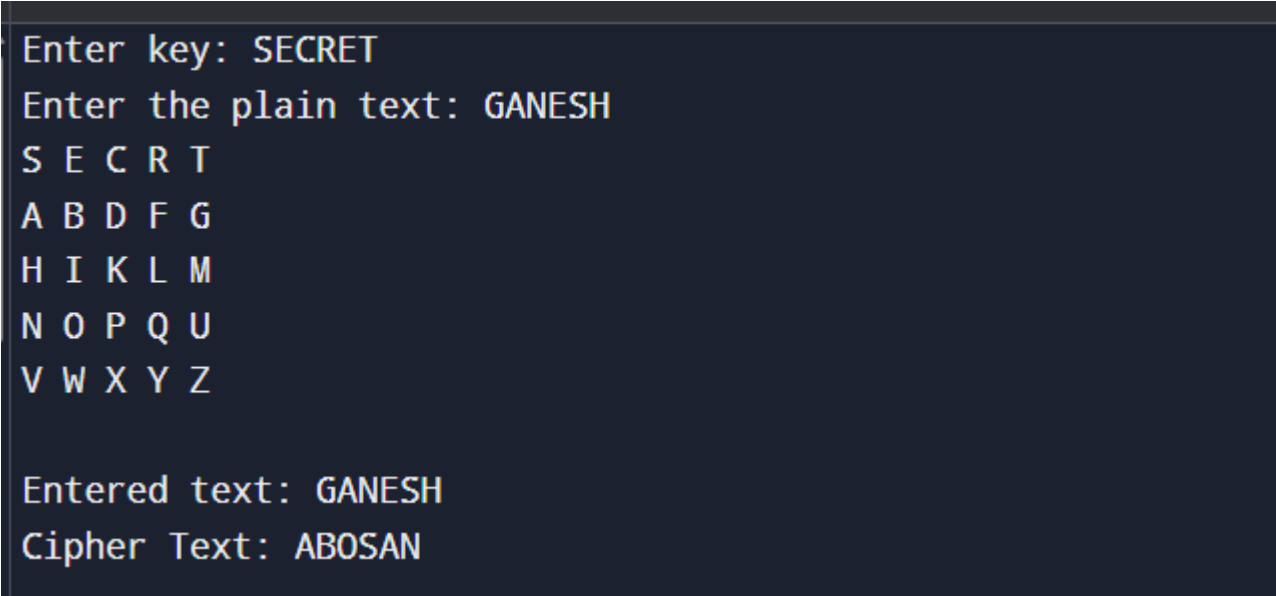
int main() {
    char key[MX][MX], keyst[20], str[100];

    printf("Enter key: ");
    fgets(keyst, sizeof(keyst), stdin);
    keyst[strlen(keyst)] = '\0';

```

```
printf("Enter the plain text: ");  
fgets(str, sizeof(str), stdin);  
str[strcspn(str, "\n")] = '\0';  
  
prepare_input(keystr);  
prepare_input(str);  
  
generate_key_matrix(key, keystr);  
  
printf("\nEntered text: %s", str);  
encrypt(str, key);  
  
return 0;  
}
```

## OUTPUT:



```
Enter key: SECRET  
Enter the plain text: GANESH  
S E C R T  
A B D F G  
H I K L M  
N O P Q U  
V W X Y Z  
  
Entered text: GANESH  
Cipher Text: ABOSAN
```

## RESULT:

The program is executed successfully