

**Walchand College Of Engineering, Sangli**  
**Department of Computer Science and Engineering**  
**Subject: C&NS Lab**

**Batch: B4**

**Name: Gayatri Sopan Gade**

**PRN:2020BTECS00210**

---

**Assignment 3**

**Title:** Implementation of Play Fair Cipher

**Introduction:**

The Playfair cipher was the first practical digraph substitution cipher. The scheme was invented in 1854 by Charles Wheatstone but was named after Lord Playfair who promoted the use of the cipher. In playfair cipher unlike traditional cipher we encrypt a pair of alphabets(digraphs) instead of a single alphabet. It was used for tactical purposes by British forces in the Second Boer War and in World War I and for the same purpose by the Australians during World War II. This was because Playfair is reasonably fast to use and requires no special equipment.

**Algorithm:**

**1. Generate the key Square(5×5):**

- The key square is a 5×5 grid of alphabets that acts as the key for encrypting the plaintext. Each of the 25 alphabets must be unique and one letter of the alphabet (usually J) is omitted from the table (as the table can hold only 25 alphabets). If the plaintext contains J, then it is replaced by I.
- The initial alphabets in the key square are the unique alphabets of the key in the order in which they appear followed by the remaining letters of the alphabet in order.

- 2. Algorithm to encrypt the plain text:** The plaintext is split into pairs of two letters. If there is an odd number of letters, a Z is added to the last letter.

## Rules:

- If both the letters are in the same column: Take the letter below each one (going back to the top if at the bottom)
- If both the letters are in the same row: Take the letter to the right of each one (going back to the leftmost if at the rightmost position)
- If neither of the above rules is true: Form a rectangle with the two letters and take the letters on the horizontal opposite corner of the rectangle.

**Code:**

The screenshot shows a C++ IDE with the following components:

- Top Bar:** File Edit Search View Project Execute Tools AStyle Window Help
- Toolbar:** Icons for file operations, compilation, and execution. A dropdown menu shows "ITM-GDC 4.9.2 64-bit Release".
- Project Explorer:** A list of files: ceasar.cpp, cryptanalysis.cpp, PlayFair.cpp, Vigenere.cpp, RailFence.cpp, and Columnar.cpp. "PlayFair.cpp" is selected.
- Code Editor:** Contains the following C++ code:
 

```

1 // ceasar.cpp
2 #include <string>
3 using namespace std;
4
5 class playfair {
6 public:
7     string msg;
8     char n[5][5];
9
10    char getChar( int a, int b )
11    {
12        //get the characters
13        return n[ (b + 5) % 5 ][ (a + 5) % 5 ];
14    }
15
16    bool getPos( char l, int &c, int &d )
17    {
18        //get the position
19        for( int y = 0; y < 5; y++ )
20        {
21            for( int x = 0; x < 5; x++ )
22            {
23                if( n[y][x] == l )
24                {
25                    c = x;
26                    d = y;
27                    return true;
28                }
29            }
30        }
31        return false;
32    }
33
34    void getText( string t, bool e )
35    {
36        //get the original message
37        msg = t;
38    }
39
40    void display() const {
41        for( int i = 0; i < 5; i++ )
42        {
43            for( int j = 0; j < 5; j++ )
44            {
45                cout << n[i][j] << " ";
46            }
47            cout << endl;
48        }
49    }
50
51    void encrypt() {
52        //get the original message
53        msg = t;
54        //get the key
55        key = k;
56        //get the key matrix
57        getPos( key[0], c, d );
58        for( int i = 0; i < 5; i++ )
59        {
60            for( int j = 0; j < 5; j++ )
61            {
62                n[i][j] = key[c + 5 * d + 5 * i + j];
63            }
64        }
65    }
66
67    void decrypt() {
68        //get the original message
69        msg = t;
70        //get the key
71        key = k;
72        //get the key matrix
73        getPos( key[0], c, d );
74        for( int i = 0; i < 5; i++ )
75        {
76            for( int j = 0; j < 5; j++ )
77            {
78                n[i][j] = key[c + 5 * d + 5 * i + j];
79            }
80        }
81    }
82
83    void run() {
84        //get the original message
85        msg = t;
86        //get the key
87        key = k;
88        //get the key matrix
89        getPos( key[0], c, d );
90        for( int i = 0; i < 5; i++ )
91        {
92            for( int j = 0; j < 5; j++ )
93            {
94                n[i][j] = key[c + 5 * d + 5 * i + j];
95            }
96        }
97    }
98
99    void show() {
100        //get the original message
101        msg = t;
102        //get the key
103        key = k;
104        //get the key matrix
105        getPos( key[0], c, d );
106        for( int i = 0; i < 5; i++ )
107        {
108            for( int j = 0; j < 5; j++ )
109            {
110                n[i][j] = key[c + 5 * d + 5 * i + j];
111            }
112        }
113    }
114
115    void clear() {
116        //get the original message
117        msg = t;
118        //get the key
119        key = k;
120        //get the key matrix
121        getPos( key[0], c, d );
122        for( int i = 0; i < 5; i++ )
123        {
124            for( int j = 0; j < 5; j++ )
125            {
126                n[i][j] = key[c + 5 * d + 5 * i + j];
127            }
128        }
129    }
130
131    void init() {
132        //get the original message
133        msg = t;
134        //get the key
135        key = k;
136        //get the key matrix
137        getPos( key[0], c, d );
138        for( int i = 0; i < 5; i++ )
139        {
140            for( int j = 0; j < 5; j++ )
141            {
142                n[i][j] = key[c + 5 * d + 5 * i + j];
143            }
144        }
145    }
146
147    void run() {
148        //get the original message
149        msg = t;
150        //get the key
151        key = k;
152        //get the key matrix
153        getPos( key[0], c, d );
154        for( int i = 0; i < 5; i++ )
155        {
156            for( int j = 0; j < 5; j++ )
157            {
158                n[i][j] = key[c + 5 * d + 5 * i + j];
159            }
160        }
161    }
162
163    void show() {
164        //get the original message
165        msg = t;
166        //get the key
167        key = k;
168        //get the key matrix
169        getPos( key[0], c, d );
170        for( int i = 0; i < 5; i++ )
171        {
172            for( int j = 0; j < 5; j++ )
173            {
174                n[i][j] = key[c + 5 * d + 5 * i + j];
175            }
176        }
177    }
178
179    void clear() {
180        //get the original message
181        msg = t;
182        //get the key
183        key = k;
184        //get the key matrix
185        getPos( key[0], c, d );
186        for( int i = 0; i < 5; i++ )
187        {
188            for( int j = 0; j < 5; j++ )
189            {
190                n[i][j] = key[c + 5 * d + 5 * i + j];
191            }
192        }
193    }
194
195    void init() {
196        //get the original message
197        msg = t;
198        //get the key
199        key = k;
200        //get the key matrix
201        getPos( key[0], c, d );
202        for( int i = 0; i < 5; i++ )
203        {
204            for( int j = 0; j < 5; j++ )
205            {
206                n[i][j] = key[c + 5 * d + 5 * i + j];
207            }
208        }
209    }
210
211    void run() {
212        //get the original message
213        msg = t;
214        //get the key
215        key = k;
216        //get the key matrix
217        getPos( key[0], c, d );
218        for( int i = 0; i < 5; i++ )
219        {
220            for( int j = 0; j < 5; j++ )
221            {
222                n[i][j] = key[c + 5 * d + 5 * i + j];
223            }
224        }
225    }
226
227    void show() {
228        //get the original message
229        msg = t;
230        //get the key
231        key = k;
232        //get the key matrix
233        getPos( key[0], c, d );
234        for( int i = 0; i < 5; i++ )
235        {
236            for( int j = 0; j < 5; j++ )
237            {
238                n[i][j] = key[c + 5 * d + 5 * i + j];
239            }
240        }
241    }
242
243    void clear() {
244        //get the original message
245        msg = t;
246        //get the key
247        key = k;
248        //get the key matrix
249        getPos( key[0], c, d );
250        for( int i = 0; i < 5; i++ )
251        {
252            for( int j = 0; j < 5; j++ )
253            {
254                n[i][j] = key[c + 5 * d + 5 * i + j];
255            }
256        }
257    }
258
259    void init() {
260        //get the original message
261        msg = t;
262        //get the key
263        key = k;
264        //get the key matrix
265        getPos( key[0], c, d );
266        for( int i = 0; i < 5; i++ )
267        {
268            for( int j = 0; j < 5; j++ )
269            {
270                n[i][j] = key[c + 5 * d + 5 * i + j];
271            }
272        }
273    }
274
275    void run() {
276        //get the original message
277        msg = t;
278        //get the key
279        key = k;
280        //get the key matrix
281        getPos( key[0], c, d );
282        for( int i = 0; i < 5; i++ )
283        {
284            for( int j = 0; j < 5; j++ )
285            {
286                n[i][j] = key[c + 5 * d + 5 * i + j];
287            }
288        }
289    }
290
291    void show() {
292        //get the original message
293        msg = t;
294        //get the key
295        key = k;
296        //get the key matrix
297        getPos( key[0], c, d );
298        for( int i = 0; i < 5; i++ )
299        {
300            for( int j = 0; j < 5; j++ )
301            {
302                n[i][j] = key[c + 5 * d + 5 * i + j];
303            }
304        }
305    }
306
307    void clear() {
308        //get the original message
309        msg = t;
310        //get the key
311        key = k;
312        //get the key matrix
313        getPos( key[0], c, d );
314        for( int i = 0; i < 5; i++ )
315        {
316            for( int j = 0; j < 5; j++ )
317            {
318                n[i][j] = key[c + 5 * d + 5 * i + j];
319            }
320        }
321    }
322
323    void init() {
324        //get the original message
325        msg = t;
326        //get the key
327        key = k;
328        //get the key matrix
329        getPos( key[0], c, d );
330        for( int i = 0; i < 5; i++ )
331        {
332            for( int j = 0; j < 5; j++ )
333            {
334                n[i][j] = key[c + 5 * d + 5 * i + j];
335            }
336        }
337    }
338
339    void run() {
340        //get the original message
341        msg = t;
342        //get the key
343        key = k;
344        //get the key matrix
345        getPos( key[0], c, d );
346        for( int i = 0; i < 5; i++ )
347        {
348            for( int j = 0; j < 5; j++ )
349            {
350                n[i][j] = key[c + 5 * d + 5 * i + j];
351            }
352        }
353    }
354
355    void show() {
356        //get the original message
357        msg = t;
358        //get the key
359        key = k;
360        //get the key matrix
361        getPos( key[0], c, d );
362        for( int i = 0; i < 5; i++ )
363        {
364            for( int j = 0; j < 5; j++ )
365            {
366                n[i][j] = key[c + 5 * d + 5 * i +
```

C:\Users\lenevo\Downloads\C&NS Assignments\C&NS Assignments\Experiment - 3\PlayFair.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug [\*] caesar.cpp cryptanalysis.cpp PlayFair.cpp Vigenere.cpp RailFence.cpp Columnnar.cpp

```
43         continue;
44         if (*it == 'J')
45             *it = 'I';
46         msg += *it;
47     }
48     if ( e )
49     {
50         string nmsg = ""; size_t len = msg.length();
51         for( size_t x = 0; x < len; x ++ )
52         {
53             nmsg += msg[x];
54             if( x + 1 < len )
55             {
56                 if( msg[x] == msg[x + 1] )
57                     nmsg += 'X';
58                 else
59                 {
60                     nmsg += msg[x + 1];
61                     x++;
62                 }
63             }
64         }
65         msg = nmsg;
66     }
67     if( msg.length() & 1 )
68         msg += 'X';
69 }
70
71 void createEncoder( string key)
72 {
73     //creation of the key table
74     string s = "";
75     vector<char> v;
76     for(int i=0;i<key.size();i++)
77     {
78         if(key[i] == 'J')
79             continue;
```

Compiler Resources Compile Log Debug Find Results

C:\Users\lenevo\Downloads\C&NS Assignments\C&NS Assignments\Experiment - 3\PlayFair.cpp - [Executing] - Dev-C++ 5.11

File Edit Search View Project Execute Tools AStyle Window Help

TM-GCC 4.9.2 64-bit Release

(globals)

Project Classes Debug [\*] caesar.cpp cryptanalysis.cpp PlayFair.cpp Vigenere.cpp RailFence.cpp Columnnar.cpp

```
82     else
83     {
84         if(find(v.begin(), v.end(), toupper(key[i])) != v.end())
85             continue;
86         v.push_back(toupper(key[i]));
87     }
88 }
89
90 for(int i=0;i<26;i++)
91 {
92     if('A'+i == 'J')
93     {
94         if(find(v.begin(), v.end(), 'I') != v.end())
95             continue;
96         v.push_back('I');
97     }
98     else
99     {
100         if(find(v.begin(), v.end(), 'A'+i) != v.end())
101             continue;
102         v.push_back('A'+i);
103     }
104 }
105
106 for(int i=0;i<v.size();i++)
107 {
108     s+=v[i];
109 }
110 copy( s.begin(), s.end(), &n[0][0] );
111
112 void play( int dir )
113 {
114     int j,k,p,q;
115     string nmsg;
116     for( string::const_iterator it = msg.begin(); it != msg.end(); it++ )
117     {
118         if( *it == 'J' )
119             continue;
```

Compiler Resources Compile Log Debug Find Results

```
C:\Users\lenevo\Downloads\C&NS Assignments\C&NS Assignments\Experiment - 3\PlayFair.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
[global]
Project Classes Debug [*] ceaser.cpp cryptanalysis.cpp PlayFair.cpp Vigenere.cpp RailFence.cpp Columnar.cpp

214
215 {
216     cout<<"Enter data to be Decrypted:\n";
217     cin.ignore();
218     getline(cin,sample);
219     cout<<"Enter the key: ";
220     getline(cin,key);
221     cout<<"Decrypted String:\n";
222     cout<<pf.play(key,sample,false)<<endl;;
223 }
224 else
225 {
226     cout<<"Enter File Name:\n";
227     cin.ignore();
228     getline(cin,sample);
229     cout<<"Enter the key: ";
230     getline(cin,key);
231     fstream myfile;
232     myfile.open(sample.c_str());
233     string str,s;
234     if(myfile.is_open())
235     {
236         cout << "Error while Opening File";
237         while(getline(myfile,str))
238         {
239             s+=str;
240             myfile.close();
241             s=pf.play(key,s,false);
242             myfile.open("PlainText.txt",ios_base::out);
243             if(myfile.is_open())
244             {
245                 myfile.write(s.data(),s.size());
246                 cout<<"File Decrypted\n";
247                 myfile.close();
248             }
249             break;
250         }
251     }
252     return 0;
253 }
```

## Output:

```
C:\Users\lenevo\Downloads\C&NS Assignments\C&NS Assignments\Experiment - 3\PlayFair.exe
PlayFair Cipher
1. Encryption
2. Decryption
3. Exit
Enter Choice: 1
Data is from
1. Manual Entering
2. File
Enter Choice: 1
Enter data to be Encrypted:
gayatri
Enter the key: good
Encrypted String:
OBAHUSFZ
PlayFair Cipher
1. Encryption
2. Decryption
3. Exit
Enter Choice: 2
Data is from
1. Manual Entering
2. File
Enter Choice: 1
Enter data to be Decrypted:
OBAHUSFZ
Enter the key: good
Decrypted String:
GAYATRIX
PlayFair Cipher
1. Encryption
2. Decryption
3. Exit
Enter Choice:
```