

个人信息

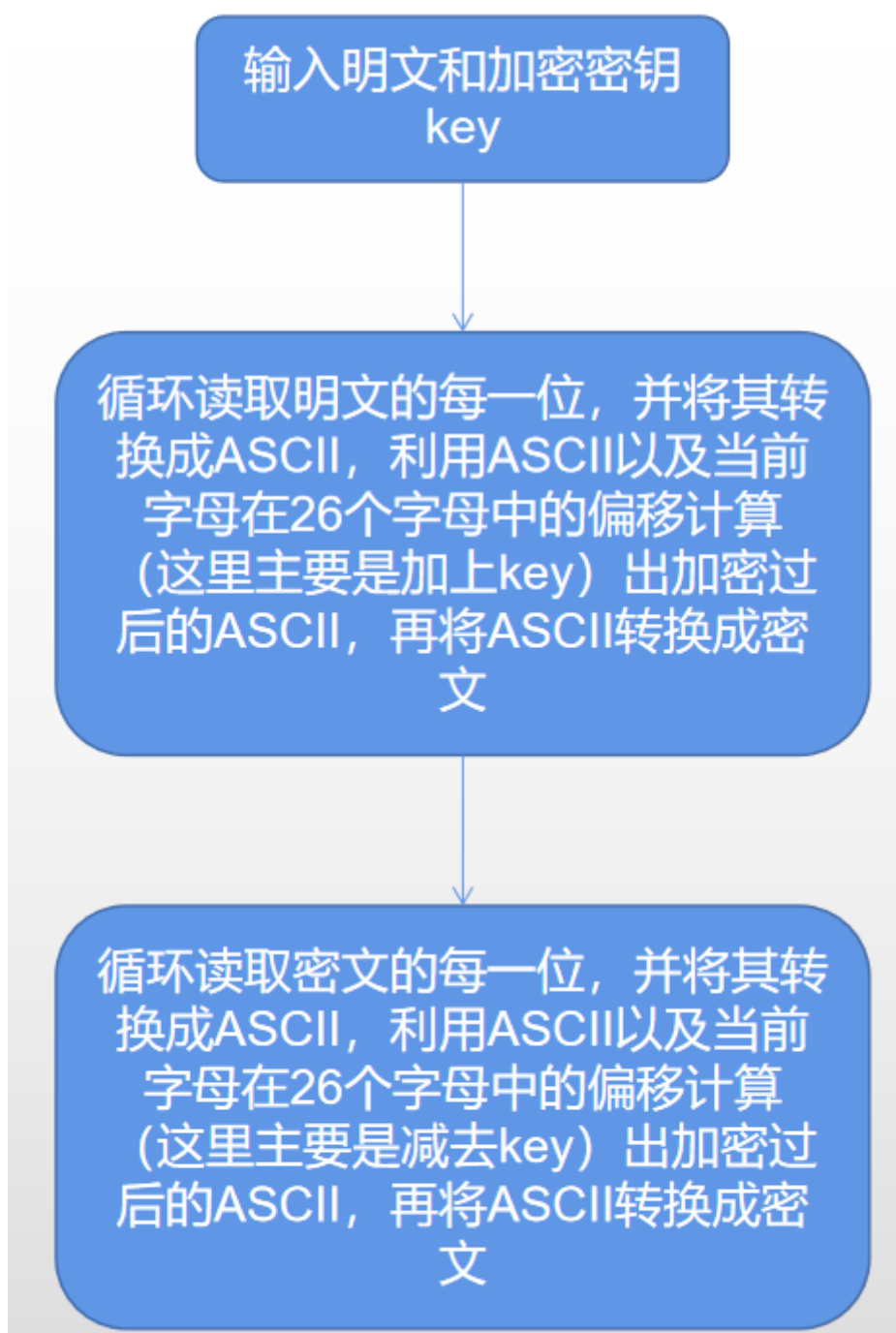
姓名：付文轩

学号：1911410

专业：信息安全

1、移位密码

算法流程



实现代码

```

1 void ShiftCipher() {
2     // 输入明文
3     cout << "Please input message:\n";
4     char* M = new char[MAX_SIZE_OF_M];
5     cin.getline(M, MAX_SIZE_OF_M);
6     //cout << M;
7
8     // 设置加密密钥key
9     int key;
10    cout << "Please input key(1-25):";
11    cin >> key;
12
13    // 密文结果初始化
14    char* C = new char[MAX_SIZE_OF_M];
15    // 移位加密
16    int index = 0;
17    while (M[index] != '\0') {
18        // 获取当前位置的ASCII
19        int ASCII = int(M[index]);
20        if (ASCII <= 90 && ASCII >= 65) {
21            // 如果是大写字母
22            int i = (ASCII - 65 + key) % 26;
23            C[index] = char(65 + i);
24        }
25        else {
26            if (ASCII <= 122 && ASCII >= 97) {
27                // 如果是小写字母
28                int i = (ASCII - 97 + key) % 26;
29                C[index] = char(97 + i);
30            }
31            else {
32                // 如果是非字母
33                C[index] = M[index];
34            }
35        }
36        // 下角标索引+1
37        index += 1;
38    }
39    // 结尾补充结束符
40    C[index] = '\0';
41
42    // 输出加密结果
43    cout << "Encrypted ciphertext is: ";
44    cout << C << endl;
45
46    // 开始解密
47    // 解密结果初始化
48    char* AM = new char[MAX_SIZE_OF_M];
49    index = 0;
50    while (C[index] != '\0') {
51        // 获取当前位置的ASCII
52        int ASCII = int(C[index]);
53        if (ASCII <= 90 && ASCII >= 65) {
54            // 如果是大写字母
55            int i = (ASCII - 65 - key + 26) % 26;
56            AM[index] = char(65 + i);
57        }

```

```

58         else {
59             if (ASCII <= 122 && ASCII >= 97) {
60                 // 如果是小写字母
61                 int i = (ASCII - 97 - key + 26) % 26;
62                 AM[index] = char(97 + i);
63             }
64             else {
65                 // 如果是非字母
66                 AM[index] = C[index];
67             }
68         }
69         // 下角标索引+1
70         index += 1;
71     }
72     // 结尾补充结束符
73     AM[index] = '\0';
74
75     // 输出解密结果
76     cout << "After decryption, the plaintext is: " << AM << endl;
77 }

```

实验结果

使用的明文为：public keys

使用的密钥为：3

加密后的密文为：sxeolf nhbv

解密得到的明文为：public keys

运行截图：

```

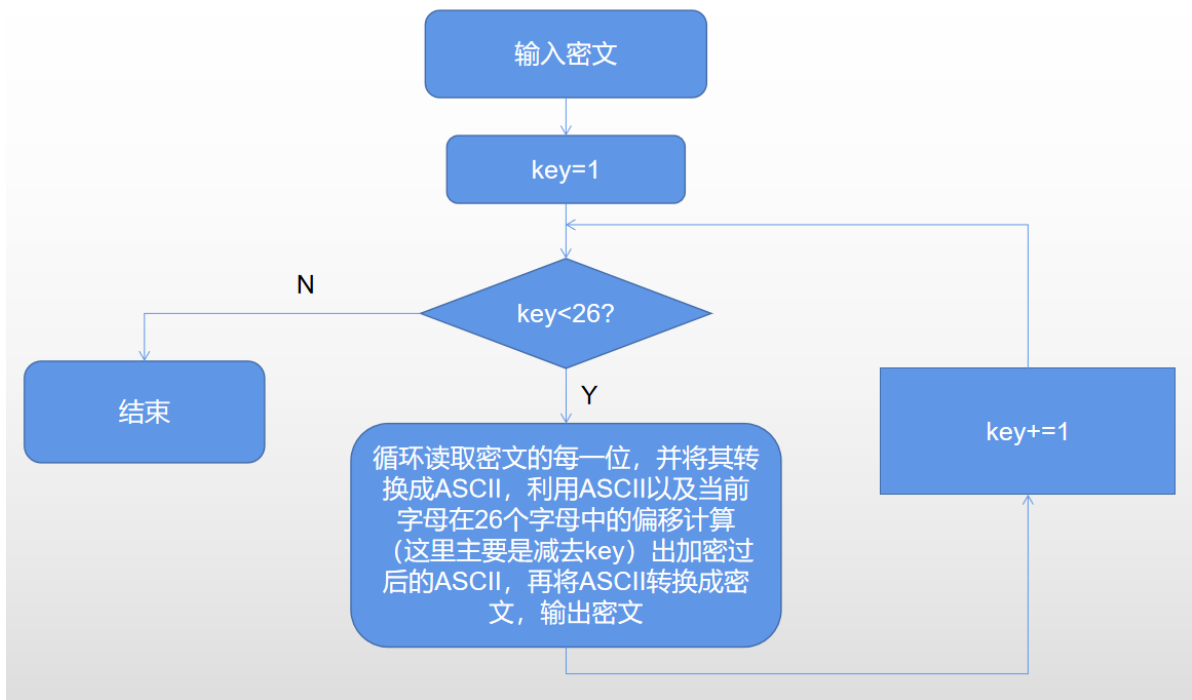
Microsoft Visual Studio 调试控制台
输入序号选择相应实验内容：
1、移位密码加密和解密
2、对移位密码的攻击
3、单表置换密码加密和解密
4、对单表置换密码的攻击方法
1
Please input message:
public keys
Please input key(1-25):3
Encrypted ciphertext is: sxeolf nhbv
After decryption, the plaintext is: public keys

D:\Study\terms\3. Junior\FirstSemester\密码学（古力）\Expertation\1\Project1\Debug\Project1.exe (进程 20344) 已退出，代码
为 0。
按任意键关闭此窗口。 . . .

```

2、对移位密码的攻击

算法流程



实现代码

```
1 void AttackShiftCipher() {
2     // 输入密文
3     cout << "Please input encrypted message:\n";
4     char* C = new char[MAX_SIZE_OF_M];
5     cin.getline(C, MAX_SIZE_OF_M);
6
7     for (int key = 1; key < 26; key++) {
8         // 明文结果初始化
9         char* M = new char[MAX_SIZE_OF_M];
10        // 移位解密
11        int index = 0;
12        while (C[index] != '\0') {
13            // 获取当前位置的ASCII
14            int ASCII = int(C[index]);
15            if (ASCII <= 90 && ASCII >= 65) {
16                // 如果是大写字母
17                int i = (ASCII - 65 - key + 26) % 26;
18                M[index] = char(65 + i);
19            }
20            else {
21                if (ASCII <= 122 && ASCII >= 97) {
22                    // 如果是小写字母
23                    int i = (ASCII - 97 - key + 26) % 26;
24                    M[index] = char(97 + i);
25                }
26                else {
27                    // 如果是非字母
28                    M[index] = C[index];
29                }
30            }
31            // 下角标索引+1
32            index += 1;
33        }
34        // 结尾补充结束符
```

```

35     M[index] = '\0';
36
37     // 输出解密结果
38     cout << "Key = " << key << "\tMessage = " << M << endl;
39 }
40 }

```

实验结果

使用的密文为：sxeolf nhbv

解密得到的明文为：public keys

运行截图：

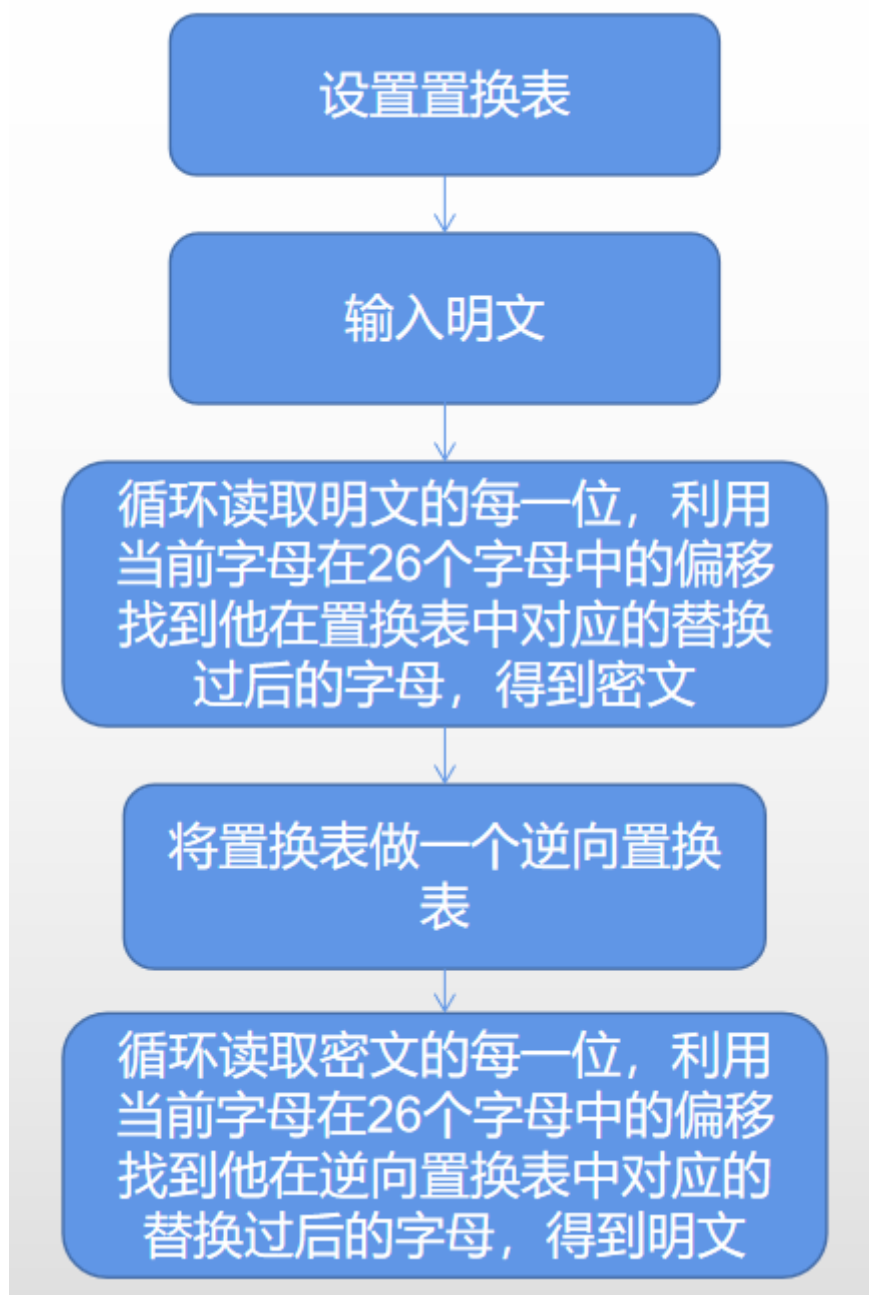
```

Microsoft Visual Studio 调试控制台
2
Please input encrypted message:
sxeolf nhbv
Key = 1 Message = rwdnke mgau
Key = 2 Message = qvcjmd lfzt
Key = 3 Message = public keys
Key = 4 Message = otakhb jdxr
Key = 5 Message = nszjga icwq
Key = 6 Message = mryifz hbvp
Key = 7 Message = lqxhey gauo
Key = 8 Message = kpwgdx fztn
Key = 9 Message = jovfcw eysm
Key = 10 Message = inuebv dxrl
Key = 11 Message = hmt dau cwqk
Key = 12 Message = glsczt bvpj
Key = 13 Message = fkrbys auoi
Key = 14 Message = ejqaxr ztnh
Key = 15 Message = dipzwq ysmg
Key = 16 Message = choypv xrlf
Key = 17 Message = bgnxuo wqke
Key = 18 Message = afmwtn vpjd
Key = 19 Message = zelvsm uoic
Key = 20 Message = ydkurl tn timer
Key = 21 Message = xcjtkq smga
Key = 22 Message = wbispi rlfz
Key = 23 Message = vahroi qkey
Key = 24 Message = uzgqnh pjdx
Key = 25 Message = tyfpmg oicw
D:\Study\terms\3. Junior\FirstSemester\密码学（古力）\Expertation\1\Project1\Debug\Project1.exe (进程 15804) 已退出。代码

```

3、单表置换密码

算法流程



实现代码

```
1 void SingleTableReplacement() {
2     // 设置置换表
3     // 置换之前的内容就使用ASCII进行索引，用ASCII减去第一个的位置来算相对偏移
4     // 如果是大写，ascii上进行处理就好
5     char ReplaceTable[26] = {
6         'h','k','w','t','x','y','s','g','b','p','q','e','j','a','z','m','l','n','o',
7         'f','c','i','d','v','u','r'};
8
9     // 输入明文
10    cout << "Please input message:\n";
11    char* M = new char[MAX_SIZE_OF_M];
12    cin.getline(M, MAX_SIZE_OF_M);
13
14    // 密文结果初始化
15    char* C = new char[MAX_SIZE_OF_M];
16
17    // 循环代换
18    int index = 0;
```

```

17 while (M[index] != '\0') {
18     // 获取当前位置的ASCII
19     int ASCII = int(M[index]);
20     if (ASCII <= 90 && ASCII >= 65) {
21         // 如果是大写字母
22         int i = ASCII - 65;
23         // 获取到的代换后的字母是小写字母，需要利用ASCII的关系转换成大写字母
24         C[index] = char(int(ReplaceTable[i]) - 32);
25     }
26     else {
27         if (ASCII <= 122 && ASCII >= 97) {
28             // 如果是小写字母
29             int i = ASCII - 97;
30             C[index] = ReplaceTable[i];
31         }
32         else {
33             // 如果是非字母
34             C[index] = M[index];
35         }
36     }
37     // 下角标索引+1
38     index += 1;
39 }
40 // 结尾补充结束符
41 C[index] = '\0';
42
43 // 输出加密结果
44 cout << "Encrypted ciphertext is: ";
45 cout << C << endl;
46
47
48 // 开始解密
49 // 首先对代换表进行一个逆转操作，预处理
50 char reverseTable[26];
51 for (int i = 0; i < 26; i++) {
52     // 首先获取当前位置字符的ASCII
53     int temp_ascii = int(ReplaceTable[i]);
54     // ASCII映射到26位置空间之后，填入到相关位置
55     reverseTable[temp_ascii - 97] = char(i + 97);
56 }
57
58 // 解密内容初始化
59 char* AM = new char[MAX_SIZE_OF_M];
60
61 index = 0;
62 while (C[index] != '\0') {
63     // 获取当前位置的ASCII
64     int ASCII = int(C[index]);
65     if (ASCII <= 90 && ASCII >= 65) {
66         // 如果是大写字母
67         int i = ASCII - 65;
68         // 找到的是对应的小写字母，利用ASCII转换成大写即可
69         AM[index] = char(int(reverseTable[i]) - 32);
70     }
71     else {
72         if (ASCII <= 122 && ASCII >= 97) {
73             // 如果是小写字母
74             int i = ASCII - 97;

```

```

75         // 找到他在逆转之后的表的位置
76         AM[index] = reverseTable[i];
77     }
78     else {
79         // 如果是非字母
80         AM[index] = C[index];
81     }
82 }
83 // 下角标索引+1
84 index += 1;
85 }
86 // 结尾补充结束符
87 AM[index] = '\0';
88
89 // 输出解密结果
90 cout << "After decryption, the plaintext is: " << AM << endl;
91 }

```

实验结果

输入的明文为：public keys

使用的置换表为：

```

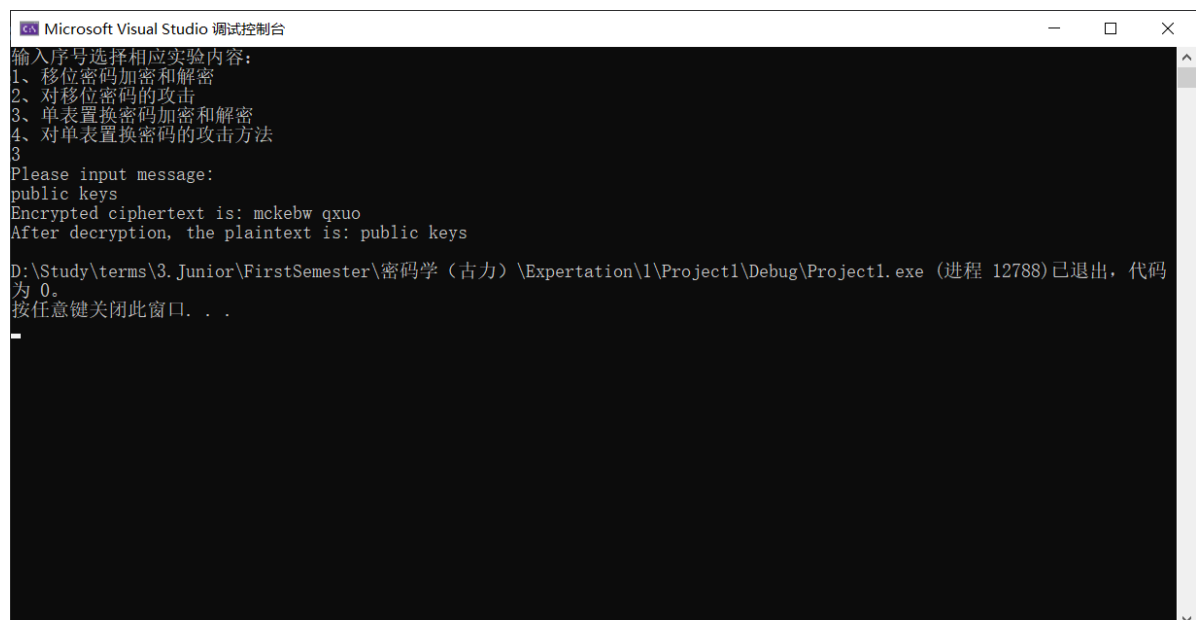
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
H K W T X Y S G B P Q E J A Z M L N O F C I D V U R

```

得到的密文为：mckebw qxuo

解密得到的明文为：public keys

运行截图：



```

Microsoft Visual Studio 调试控制台
输入序号选择相应实验内容：
1、移位密码加密和解密
2、对移位密码的攻击
3、单表置换密码加密和解密
4、对单表置换密码的攻击方法
3
Please input message:
public keys
Encrypted ciphertext is: mckebw qxuo
After decryption, the plaintext is: public keys
D:\Study\terms\3. Junior\FirstSemester\密码学（古力）\Expertation\1\Project1\Debug\Project1.exe (进程 12788) 已退出，代码
为 0。
按任意键关闭此窗口。 . . .

```

4、对单表置换密码的攻击方法

分析过程

1. 统计密文中各个字母出现的频次

2. 对单词长度为1的单词进行分析

1. 发现N出现的次数是6次；H出现的次数是1次
2. 观察这些N出现的间隔可以发现，有些N出现的间隔很短，根据人在叙述的时候一般不会对相隔很近的位置多次说“l”猜测N应该是由a替换而来，那么这样H就是由i替换而来

3. 对单词长度为2的单词进行分析

1. 发现长度为2的单词有：JB3次，JR1次，MF4次
2. 仅根据最常用统计单词中的统计：to 3.02 of 2.61 is 1.68 in 1.57，发现这里并不能确定什么信息，所以先跳过这里的分析

4. 对单词长度为3的单词进行分析

1. 首先看到第一个单词SIC就是长度为3，根据人们说话的习惯，一般来说放在开头并且是三个字母的单词就是the，那么这里就假设SIC是由the替换而来
2.

5.

之后便是依次对越来越长的单词串进行分析，同时在每次分析、猜测到替换过程后，将替换表进行修改并在原文段落修改（对替换表的修改需要一个一个进行手动修改），查看当前这些字母如果被替换了以后，密文会变成什么样，然后基于以上修改过后的结果，对剩下没有进行破解的内容进行猜测（看比较短的地方的位置什么样的单词比较合适）分析、替换。最终得到一个通顺的、较为可信的明文。

实现代码

```
1 void AttackSingleTableReplacement() {
2     // 输入要统计的密文
3     cout << "Please input encrypted message:\n";
4     char* C = new char[MAX_SIZE_OF_M];
5     cin.getline(C, MAX_SIZE_OF_M);
6
7     // 记录统计熟练的数组初始化
8     int frequency[26] = { 0 };
9
10    // 遍历密文，统计频次
11    int index = 0;
12    while (C[index] != '\0') {
13        // 获取当前位置的ASCII
14        int ASCII = int(C[index]);
15        if (ASCII <= 90 && ASCII >= 65) {
16            // 如果是大写字母
17            int i = ASCII - 65;
18            // 获取到的代换后的字母是小写字母，需要利用ASCII的关系转换成大写字母
19            frequency[i] += 1;
20        }
21        else {
22            if (ASCII <= 122 && ASCII >= 97) {
23                // 如果是小写字母
24                int i = ASCII - 97;
25                frequency[i] += 1;
26            }
27            else {
28                // 如果是非字母
29                // 暂时不做任何行为
30            }
31        }
32        // 下角标索引+1
33        index += 1;
34    }
```

```

34     }
35
36     // 输出统计的频次
37     cout << "After statistics, the occurrence frequency of each letter
is:";
38     for (int i = 0; i < 26; i++) {
39         if (i % 10 != 0)
40             cout << char(97 + i) << ": " << frequency[i] << '\t';
41         else
42             cout << endl << char(97 + i) << ": " << frequency[i] << '\t';
43     }
44
45     // 根据分析结果得到正向替换表
46     char replaceTable[26] = {
'n', 'h', 'g', 'd', 'c', 'f', 'e', 'i', 'j', '0', '0', 'a', 'q', 'b', 'm', 'x', '0', 'p', 'r',
's', 'z', 't', 'v', '0', 'y', 'o' };
47
48     // 开始解密
49     // 首先对替换表进行一个逆转操作，预处理
50     char reverseTable[26];
51     // 初始化
52     for (int i = 0; i < 26; i++) {
53         reverseTable[i] = '0';
54     }
55
56     for (int i = 0; i < 26; i++) {
57         // 首先获取当前位置字符的ASCII
58         int temp_ascii = int(replaceTable[i]);
59         // ASCII映射到26位置空间之后，填入到相关位置
60         // 检查ASCII是不是在正确的范围内
61         if (temp_ascii >= 97 && temp_ascii <= 122) {
62             // 如果是，则放入到对应的表中
63             reverseTable[temp_ascii - 97] = char(i + 97);
64         }
65         else {
66             // 如果不是，也就是说这是原本初始化时的内容，什么都不动
67         }
68     }
69
70     // 检查逆向替换表，如果位置上是0的就自己替换为自己，并且设置为大写，以和成功替换的区
分开
71     for (int i = 0; i < 26; i++) {
72         if (reverseTable[i] == '0')
73             reverseTable[i] = char(i + 65);
74     }
75
76     // 解密内容初始化
77     char* AM = new char[MAX_SIZE_OF_M];
78
79     index = 0;
80     while (C[index] != '\0') {
81         // 获取当前位置的ASCII
82         int ASCII = int(C[index]);
83         if (ASCII <= 90 && ASCII >= 65) {
84             // 如果是大写字母
85             int i = ASCII - 65;
86             AM[index] = reverseTable[i];
87         }

```

```

88         else {
89             if (ASCII <= 122 && ASCII >= 97) {
90                 // 如果是小写字母
91                 int i = ASCII - 97;
92                 // 找到他在逆转之后的表的位置
93                 AM[index] = reverseTable[i];
94             }
95             else {
96                 // 如果是非字母
97                 AM[index] = C[index];
98             }
99         }
100         // 下角标索引+1
101         index += 1;
102     }
103     // 结尾补充结束符
104     AM[index] = '\0';
105
106     // 输出解密结果
107     cout << "\nAfter decryption, the plaintext is:\n" << AM << endl;
108 }

```

实验结果

输入的密文为：SIC GCBSPNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE
 JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJHAY JBRCGZPC GINBBCA
 JB RZGI N VNY SINS SIC MPJEBNA QCRRNEC GNB MBAY HC PCGMTPCPD HY SIC PJEISFZA
 PCGJXJCBRS SIC XNPSJGJXNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEBNSMP MF SIC
 QCRRNEC HMH SIC PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRICR SM ENJB
 ZBNZSIMPJOCD GMBSPMA MF SIC QCRRNEC

经过统计得到的频次表为：

a: 10 b: 28 c: 36 d: 3 e: 9 f: 7 g: 14 h: 9 i: 18 j: 28
 k: 0 l: 0 m: 29 n: 31 o: 1 p: 23 q: 8 r: 21 s: 33 t: 2
 u: 0 v: 3 w: 0 x: 12 y: 7 z: 5

经过分析得到的置换表为：

a b c d e f g h i j k l m n o p q r s t u v w x y z
 n h g d c f e i j 0 0 a q b m x 0 p r s z t v 0 y o

其中标识为0的表示本段文字中出现频次是0，仅根据本文无法分析出来代换的内容

最终得到的明文为：the central problem in cryptography is that of transmitting information from a point a to a point b by means of a possibly insecure channel in such a way that the original message can only be recovered by the rightful recipients the participants in the transaction are alice the originator of the message bob the receiver and oscar a possible opponent who wishes to gain unauthorized control of the message

运行截图：

```
Microsoft Visual Studio 调试控制台
输入序号选择相应实验内容：
1、移位密码加密和解密
2、对移位密码的攻击
3、单表置换密码加密和解密
4、对单表置换密码的攻击方法
4
Please input encrypted message:
SIC GCBSPNA XPMHACQ JB GPYXSMEPNXIY JR SINS MF SPNBRQJSSJBE JBFMPQNSJMB FPMQ N XMJBS N SM N XMJBS H HY QCNBR MF N XMRRJH
AY JBRGZPC GINBBCA JB RZGI N VNY SINS SIC MPJEJBNA QCRRNEC GNB MBAY HC PCGMITCPCD HY SIC PJEISFZA PCGJXJCBSR SIC XNPSJGJ
XNBSR JB SIC SPNBRNGSJMB NPC NAJGC SIC MPJEJBNSMP MF SIC QCRRNEC HMH SIC PCGCJTCP NBD MRGNP N XMRRJHAC MXXMBCBS VIM VJRI
CR SM ENJB ZBNZSIMPJOC GMBSPMA MF SIC QCRRNEC
After statistics, the occurrence frequency of each letter is:
a: 10 b: 28 c: 36 d: 3 e: 9 f: 7 g: 14 h: 9 i: 18 j: 28
k: 0 l: 0 m: 29 n: 31 o: 1 p: 23 q: 8 r: 21 s: 33 t: 2
u: 0 v: 3 w: 0 x: 12 y: 7 z: 5
After decryption, the plaintext is:
the central problem in cryptography is that of transmitting information from a point a to a point b by means of a possib
ly insecure channel in such a way that the original message can only be recovered by the rightful recipients the partici
pants in the transaction are alice the originator of the message bob the receiver and oscar a possible opponent who wish
es to gain unauthorized control of the message
D:\Study\terms\3. Junior\FirstSemester\密码学（古力）\Expertation\1\Project1\Debug\Project1.exe (进程 7768)已退出，代码为
0。
按任意键关闭此窗口。 . . .
```