密码学基础实验报告

东北大学

实验名称	Vigenere 和 Column permutation cipher 的编程实现
班　级	软信-1503
学　号	20155362
姓　名	薛旗
日　期	2017.04.20
成　绩	
评 阅 人	

软件学院

一、 实验目的与意义

实验目的：

(1)通过编写程序设计实现古典密码体制中的 Vigenere Cipher，并能够加密/解密一个字符串或文件；

(2)通过编写程序设计实现古典密码体制中的 Column Permutation Cipher,并能够加密/解密一个字符串或文件；

(3)能够通过本实验了解古典密码的基本知识，能够掌握算法的基本原理，了解算法的详细步骤；

实验意义：

(1)通过实践，了解 Polyalphabetic Cipher 和 Transposition Cipher 的区别，加深对古典密码体制的理解，为深入学习密码学奠定基础；

(2)通过实践，不仅能够验证理论知识，还能通过实践加强实验手段和实践技能，培养分析问题、解决问题、应用知识的能力和创新精神，提高综合素质。

二、 实验环境

操作系统：Windows 10 专业版

调试软件：Microsoft Visual Studio Community 2017

版本号：4.7.02046

上机地点：信息学馆 B405

机器台号：随机

三、 实验的预习内容

Vigenere Cipher:

预习内容：

(1)替代密码原理：首先构造一个或多个密文字母表，然后用密文字母表中的字母或字母组来代替明文字母或字母组，各字母或字母组的相对位置不变，但其本身改变了。这样编成的密码称为代替密码。按替代所使用的密文字母表的个数可将代替密码分为单表替代密码和多表替代密码。Vigenere Cipher 属于多表替代密码。

(2)Vigenere Cipher 数学表达式：

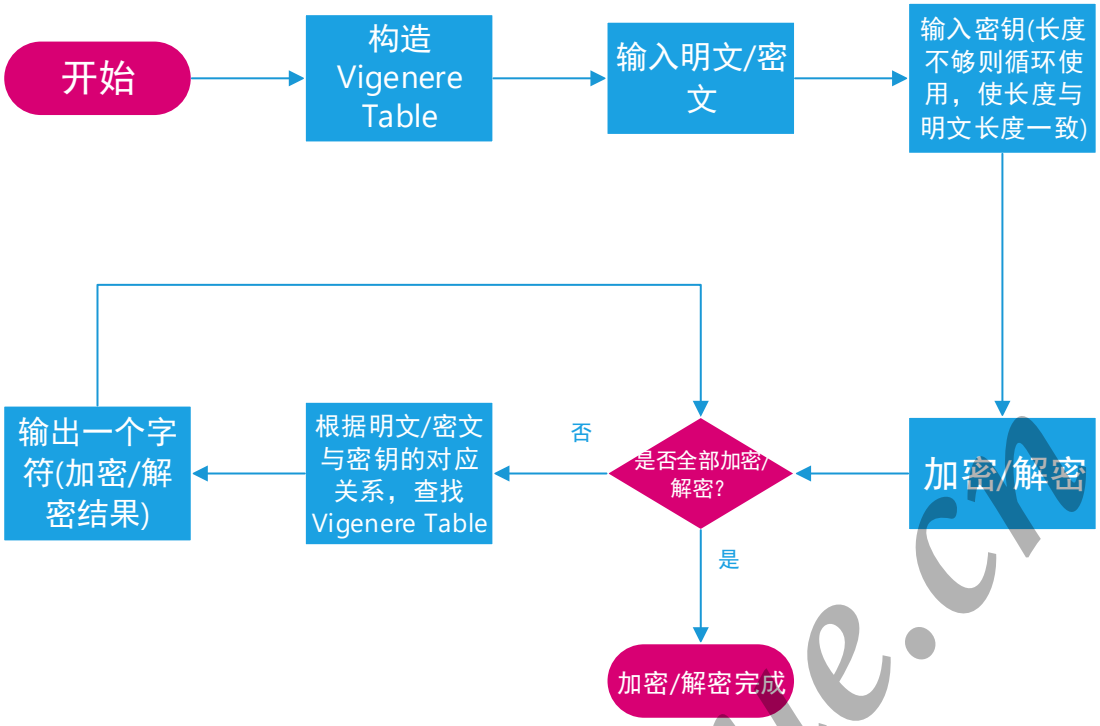
设 m 是一个正整数，定义$\mathrm{P}=\mathrm{C}=\mathrm{K}=(Z_{26})^{m}$ 对任意的密钥$\mathrm{K}=(k_1,k_2,\ldots,k_m)$,定义$e_k(x_1,x_2,\ldots,x_m)=(x_1+k_1,x_2+k_2,\ldots,x_m+k_m)$和$d_K(y_1,y_2,\ldots,y_m)=(y_1-k_1,y_2-k_2,\ldots,y_m-k_m)$,以上所有运算都是在$Z_{26}$上进行。

(3)Vigenere Cipher 加密/解密原理：在一个具有密钥字长度为 m 的 Vigenere Cipher 中，一个字母可以被映射为 m 个字母中的某一个(假定密钥字包含 m 个不同的字母)。Vigenere Cipher 使用一个词组作为密钥，密钥中的每一个字母确定一个代替表，每一个密钥字母用来加密一个明文字母。等所有密钥字母都使用玩后，密钥再循环使用。Vigenere Cipher 的替代规则是用明文字母在 Vigenere 方阵中的列和密钥字母在 Vigenere 方阵中的行的交点处的字母来代替该明文字母。其解密就是利用 Vigenere Table 进行反替代。

实验思路：

要实现对 Vigenere Cipher,关键是找到对应关系。因此，其中最重要的一项工作是构造 Vigenere Table,之后便可根据密钥与明文一一对应的关系在表中找对应的字符。假定密钥确定的是行替代表，那么密钥所确定的行和明文所确定的列的交叉位置处的字符，即为一个明文字符加密得到的密文字符。对于解密过程则与加密过程相反，由密钥先确定一个替代表（假定密钥确定的行替代表），然后在该表中找到密文字符，密文字符所在列对应的字符即为明文。这里还需要注意的是密钥是循环使用的，密钥长度与明文一致。

程序框图：(见下页)



Column Permutation Cipher:

预习内容：

(1)置换密码原理：把明文中的字母重新排列，字母本身不变，但其位置改变了，这样编成的密码称为置换密码。置换密码又称为移位密码，因为对照明密文来看，字母的位置被移动了。

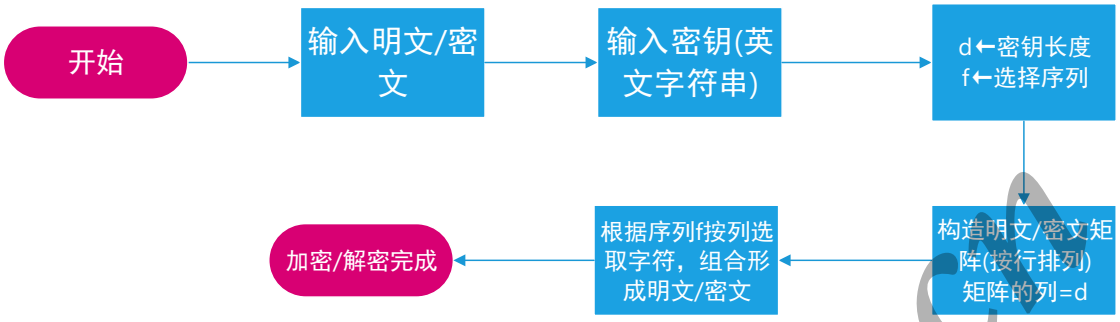
(2)Column Permutation Cipher 原理：Column Permutation Cipher 属于置换密码的一种，其加密过程大致如下：把明文按某一顺序排列成一个矩阵，其中不足部分用符号Φ填充。然后按另一顺序选出矩阵中的字母以形成密文，最后截成固定长度的字母组成密文。由加密过程可知，改变矩阵的大小和选出顺序可以得到不同形式的密码。

实验思路：

要实现 Column Permutation Cipher 的加密/解密，关键是构造明文/密文矩阵并能够得到一个合适的选出规则。在编程中通过输入一串英文字符来完成以上操作，具体步骤为：首先选用一串英文字符作为密钥，构造的矩阵的列数与字符串长度保持一致，明文/密文的长度不足以构造矩阵的则用Φ填充。按照字母的字典顺序给密钥字母编号，对于重复的字母，谁的位置靠前，谁的编

号就靠前。于是我们就得到一组与密钥词语对应的数字序列。最后据此数字序列中的数字顺序按列选出明文/密文。

程序框图：



四、 实验的步骤与调试方法

实验步骤：

步骤	说明	调用函数
1	编制标头	main()
2	加密/解密选择菜单	mainMenue()
3	算法选择菜单	menueOptions()
4	加密/解密方式(文件/键盘)	options()
5	字符检验(输入是否为纯英文)	Check(string s)
6	文件读取	readFileIntoString(char*)
7	打印 Vigenere Table	(*vigenereTable(char(&)[26][26]))[26]
8	Vigenere(键盘读取加密)	encryVigenereKeyboard()
9	Vigenere(键盘读取解密)	decryVigenereKeyboard()
10	Vigenere(文件读取加密)	encryVigenereFile()
11	Vigenere(文件读取解密)	decryVigenereFile()
12	Vigenere(加密主函数)	encryVigenere(string, string, char(*)[26])
13	Vigenere(解密主函数)	decryVigenere(string, string, char(*)[26])
14	Column(键盘读取加密)	encryColumnKeyboard()
15	Column(键盘读取解密)	decryColumnKeyboard()
16	Column(文件读取加密)	encryColumnFile()
17	Column(文件读取解密)	decryColumnFile()
18	Column(加密主函数)	encryColumn(string, string)
19	Column(解密主函数)	decryColumn(string, string)
20	Column 密钥构造	wordKey(string, int [20])

调试方法：

古典密码多为字符的替换和移位，故在调试过程中采用单步执行调试观察变量赋值，查找错误。在加密/解密过程中，可以先模拟选取一组简单的加密/解密字符，先通过人工验证，再通过程序实现，比较是否一致，从而能够快速发现问题，定位出错位置。具体调试步骤如下：

调试步骤：

（1）从错误的外部表现形式入手，确定程序出错位置；

（2）研究相关程序，找出错误原因；

（3）修改代码，排除错误；

（4）重复进行测试，观察错误是否修正。

五、 实验数据与实验结果

Vigenere Cipher:

加密：

实验数据：

Plaintext:thisistheplaintext

Key:hold

加密结果：

Ciphertext:avtvpgekldwdpbeheh

实验结果展示(见下页)：

解密：

实验数据：

Ciphertext:viozrzalnxrcgwoirbash

Key:nice

解密结果：

Plaintext:iamveryhappytomeetyou

实验结果展示：

Column Permutation Cipher:

加密：

实验数据：

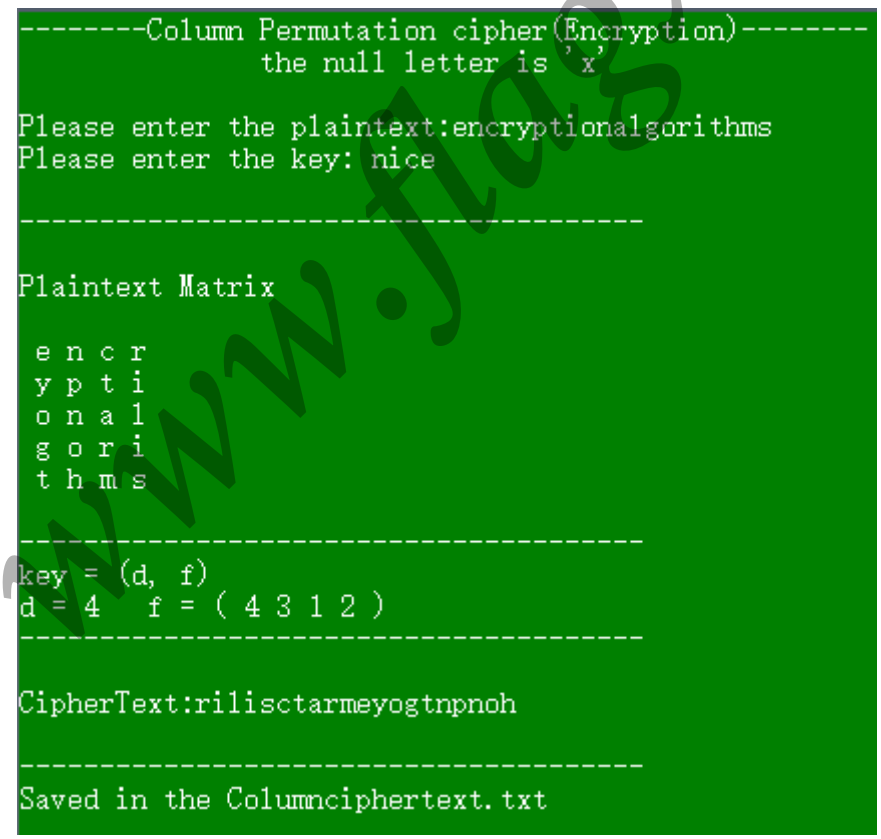
Plaintext:encryptionalgorithms

Key:nice

加密结果：

Ciphertext:rilisctarmeyogtnpnoh

实验结果展示：



解密：

实验数据：

Ciphertext:entntvoeetnshuisiyrrr

Key:machine

解密结果：

Plaintext:northeasternuniversity

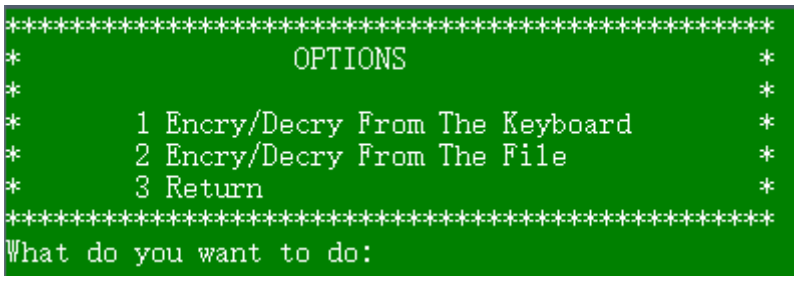
实验结果展示：

```
--------Column Permutation cipher(Decryption)--------

Please enter the ciphertext:entntvoeetnshuisiyrrr
Please enter the key: machine

---------------------------------------
key = (d, f)
d = 7   f = ( 6 1 2 4 5 7 3 )
---------------------------------------

Plaintext Matrix

  n  o  r  t  h  e  s
  t  e  r  n  u  n  i
  v  e  r  s  i  t  y

Plaintext:northesternuniversity

---------------------------------------
Saved in the Columnplaintext.txt
```

附：选择菜单

```
*************************************************
*                   OPTIONS                     *
*                                               *
*                1 Encryption                   *
*                2 Decryption                   *
*                3 Exit                         *
*************************************************
What do you want to do:
```

```
*************************************************
*              OPTIONS                          *
*                                               *
*       1 Vigenere cipher                       *
*       2 Column permutation cipher             *
*       3 Return                                *
*************************************************
What do you want to do:
```



六、 实验用程序清单

```
#include<iostream>
#include<string>
#include<fstream>
#include<sstream>
#include<stdlib.h>
#include<windows.h>

using namespace std;
void menue();
void encryVigenere(string, string, char(*list)[26]);
void decryVigenere(string, string, char(*list)[26]);
void encryVigenereKeyboard();
void decryVigenereKeyboard();
void encryVigenereFile();
void decryVigenereFile();
void encryColumn(string, string);
void decryColumn(string, string);
void encryColumnKeyboard();
void decryColumnKeyboard();
void encryColumnFile();
void decryColumnFile();
string readFileIntoString(char *filename);
char menueOptions();
char options();
char(*vigenereTable(char(&)[26][26]))[26];  //打印vigenere表(类比返回二维数组)
int mainMenue();
int Check(string s);
int *wordKey(string, int[20]);

int mainMenue() {     //主菜单
    char choose;
    string strchoose;
    do {
        system("cls");
        cout << "**********************************************" << endl;
        cout << "*                  OPTIONS                   *" << endl;
        cout << "*                                            *" << endl;
        cout << "*              1 Encryption                  *" << endl;
        cout << "*              2 Decryption                  *" << endl;
```

```
40          cout << "*                    3 Exit                          *" << endl;
41          cout << "********************************************" << endl;
42          cout << "What do you want to do: ";
43          cin >> strchoose;
44          if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3' || strchoose.length() != 1) {
45              cout << "Error!Please enter again!" << endl;
46              Sleep(500);
47          }
48      } while (strchoose.at(0) != '1'&&strchoose.at(0) !=
49      choose = strchoose[0];
50      return choose;
51  }
52
53  char menueOptions() { //选择加密/解密算法
54      string strchoose;
55      char choose;
56      do {
57          system("cls");
58          cout << "********************************************" << endl;
59          cout << "*              OPTIONS                     *" << endl;
60          cout << "*                                          *" << endl;
61          cout << "*       1 Vigenere cipher                  *" << endl;
62          cout << "*       2 Column permutation cipher        *" << endl;
63          cout << "*       3 Return                           *" << endl;
64          cout << "********************************************" << endl;
65          cout << "What do you want to do: ";
66          cin >> strchoose;
67          if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3' || strchoose.length() != 1) {
68              cout << "Error!Please enter again!";
69              Sleep(500);
70          }
71      } while (strchoose.at(0) != '1'&&strchoose.at(0) !=
72      choose = strchoose[0];
73      return choose;
74  }
75
76  char options() {      //选择加密/解密方式
77      string strchoose;
78      char choose;
79      do {
80          system("cls");
81          cout << "********************************************" << endl;
82          cout << "*                OPTIONS                   *" << endl;
83          cout << "*                                          *" << endl;
84          cout << "*       1 Encry/Decry From The Keyboard    *" << endl;
85          cout << "*       2 Encry/Decry From The File        *" << endl;
86          cout << "*       3 Return                           *" << endl;
87          cout << "********************************************" << endl;
88          cout << "What do you want to do: ";
89          cin >> strchoose;
```

```
90              if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3' || strchoose.length() != 1) {
91                  cout << "Error!Please enter again!";
92                  Sleep(500);
93              }
94          } while (strchoose.at(0) != '1'&&strchoose.at(0) !=
95          choose = strchoose[0];
96          return choose;
97      }
98
99      int Check(string s) {      //检查输入的字符串是否为纯英文
100         int i = 0;
101         while (s[i] != '\0') {
102             if (!((s[i] >= 'a'&&s[i] <= 'z') || (s[i] >= 'A'&&s[i] <= 'Z'))) {
103                 cout << "Error!Please enter again!" << endl;
104                 Sleep(1000);
105                 return 1;
106             }
107             i++;
108         }
109         return 0;
110     }
111
112     string readFileIntoString(char * filename) {    //从文件读入到string里
113         ifstream ifile(filename);
114         ostringstream buf;          //将文件读入到ostringstream对象buf中
115         char ch;
116         while (buf&&ifile.get(ch))
117             buf.put(ch);
118         return buf.str();      //返回与流对象buf关联的字符串
119     }
120
121     char(*vigenereTable(char(&list)[26][26]))[26]{//打印VigenereTable并返回二维数组
122         int i, j, k;
123         char ch;
124         for (i = 0; i < 26; i++) {
125             for (j = 0; j < 26; j++) {
126                 k = 'a' + (i + j) % 26;
127                 ch = k;
128                 list[i][j] = ch;
129             }
130         }
131         cout << "  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" << endl;
132         for (i = 0; i < 26; i++) {
133             cout << list[0][i] << " ";
134             for (j = 0; j < 26; j++) {
135
136                 cout << list[i][j] << " ";
137             }
138             cout << endl;
139         }
```

```
140     cout << endl;
141     return list;
142 }
143
144 void encryVigenere(string st, string key, char(*list)[26]) {       //加密
145
146     int m, n;
147     int i = 0;
148     int j = 0;
149     int flag = 0;
150     char ch[200] = { 0 };
151     string rule(50, '\0');
152     ofstream fout;
153
154
155     j = 0;
156     for (i = 0; i < st.length(); i++) {
157         j = j % key.length();
158         rule[i] = key[j];
159         j++;
160     }
161     rule[i] = '\0';
162     cout << endl;
163     cout << "---------------------------------" << endl;
164     cout << "Keyword:    " << rule << endl;
165     cout << "Plaintext:  " << st << endl;
166     cout << "---------------------------------" << endl << endl;
167     cout << "Ciphertext: ";
168     for (i = 0; i < st.length(); i++) {
169         m = rule[i] - 'a';
170         n = st[i] - 'a';
171         cout << list[n][m];
172         ch[i] = list[n][m];
173     }
174     fout.open("Vigenereciphertext.txt");    //Vigenereciphertext.txt文件
175     fout << ch;
176     fout.close();
177     cout << endl << endl;
178     cout << "---------------------------------" << endl;
179     cout << "Saved in the Vigenereciphertext.txt";
180     cout << endl << endl;
181
182     system("PAUSE");
183 }
184
185 void encryVigenereKeyboard() {      //从键盘获取字符串加密
186
187     int flag = 0;
188     char(*list)[26];
189     char list1[26][26] = { 0 };
```

```
190     string key, st;
191
192     do {
193         fflush(stdin);
194         system("cls");
195         cout << "-------------Vigenere cipher(Encryption)------------" << endl;
196         cout << endl << "Vigenere table" << endl << endl;
197         list = vigenereTable(list1);
198         cout << "Please enter the plaintext:" << endl;
199         cout << "Plaintext:";
200         cin >> st;
201         flag = Check(st);
202
203     } while (flag);
204     do {    //输入key
205         fflush(stdin);
206         cout << "Please enter the key: ";
207         cin >> key;
208         flag = Check(key);
209     } while (flag);
210
211     encryVigenere(st, key, list);
212 }
213
214 void encryVigenereFile() {    //从文件读取加密
215     string key;
216     char(*list)[26];
217     char list1[26][26] = { 0 };
218     char filename[30];
219     int flag = 0;
220     fflush(stdin);
221     system("cls");
222     cout << "-------------Vigenere cipher(Encryption)------------" << endl;
223     list = vigenereTable(list1);
224     cout << "Please enter the file name that you want to encryption:" << endl;
225     cout << "File name: ";
226     cin >> filename;
227     strcat(filename, ".txt");
228     fstream openfile(filename);
229     if (!openfile)
230     {
231         cout << "Open failed!" << endl;
232         exit(1);
233     }
234     char * fn = filename;
235     string st;
236     st = readFileIntoString(fn);
237     cout << filename << ": ";
238     cout << st << endl;
239
```

```
240     do {     //输入key
241         fflush(stdin);
242         cout << "Please enter the key: ";
243         cin >> key;
244         flag = Check(key);
245     } while (flag);
246     encryVigenere(st, key, list);
247 }
248
249 void decryVigenere(string st, string key, char(*list)[26]) {       //解密
250     int m, n;
251     int i = 0;
252     int j = 0;
253     int flag = 0;
254     char list1[26][26] = { 0 };
255     char cipher[26][26] = { 0 };
256     char plain[200] = { 0 };
257     string rule(50, '\0');
258     ofstream fout;
259
260     j = 0;
261     for (i = 0; i < st.length(); i++) {
262         j = j % key.length();
263         rule[i] = key[j];
264         j++;
265     }
266     rule[i] = '\0';
267     cout << endl;
268     cout << "-------------------------------------" << endl;
269     cout << "Keyword:     " << rule << endl;
270     cout << "Ciphertext:  " << st << endl;
271     cout << "-------------------------------------" << endl << endl;
272     cout << "Plaintext: ";
273
274     //以下是解密关键代码
275     for (i = 0; i < st.length(); i++) {
276         m = rule[i] - 'a';
277         for (j = 0; j < 26; j++) {
278             if (list[j][m] == st[i])
279                 break;
280         }
281         n = 'a' + j;
282         plain[i] = n;
283         cout << plain[i];
284     }
285     fout.open("Vigenereplaintext.txt");     //打开Vigenereplaintext.txt文件
286     fout << plain;
287     fout.close();
288     cout << endl << endl;
289     cout << "-------------------------------------" << endl;
```

```
290         cout << "Saved in the Vigenereplaintext.txt";
291         cout << endl << endl;
292 
293         system("PAUSE");
294 }
295 
296 void decryVigenereFile() {     //从文件读取解密
297         string key;
298         char(*list)[26];
299         char list1[26][26] = { 0 };
300         char filename[30];
301         int flag = 0;
302         fflush(stdin);
303         system("cls");
304         cout << "-------------Vigenere cipher(Decryption)------------" << endl;
305         list = vigenereTable(list1);
306         cout << "Please enter the file name that you want to decryption:" << endl;
307         cout << "File name: ";
308         cin >> filename;
309         strcat(filename, ".txt");
310         fstream openfile(filename);
311         if (!openfile)
312         {
313             cout << "Open failed!" << endl;
314             exit(1);
315         }
316         char * fn = filename;
317         string st;
318         st = readFileIntoString(fn);
319         cout << filename << ":";
320         cout << st << endl;
321 
322         do {     //输入key
323             fflush(stdin);
324             cout << "Please enter the key: ";
325             cin >> key;
326             flag = Check(key);
327         } while (flag);
328         decryVigenere(st, key, list);
329 }
330 
331 void decryVigenereKeyboard() {     //从键盘获得字符串解密
332 
333         int flag = 0;
334         char(*list)[26];
335         char list1[26][26] = { 0 };
336         string key, st;
337 
338         do {
339             fflush(stdin);
```

```
340             system("cls");
341             cout << "-------------Vigenere cipher(Decryption)------------" << endl;
342             cout << endl << "Vigenere table" << endl << endl;
343             list = vigenereTable(list1);
344             cout << "Please enter the ciphertext:" << endl;
345             cout << "Ciphertext: ";
346             cin >> st;
347             flag = Check(st);
348
349         } while (flag);
350         do {     //输入key
351             fflush(stdin);
352             cout << "Please enter the key: ";
353             cin >> key;
354             flag = Check(key);
355         } while (flag);
356
357         decryVigenere(st, key, list);
358     }
359
360     int *wordKey(string key, int k[20]) {  //Column Permutation Cipher将获得的密钥进行处理
361         int i, j;
362         int *p = k;
363         int flag = 0;
364         char temp1;
365         char tem[20] = { 0 };
366         char keygroup[20] = { 0 };
367
368         for (i = 0; i < key.length(); i++) {        //设置临时数组
369             keygroup[i] = key[i];
370         }
371
372         //冒泡排序
373         for (i = 0; i < key.length(); i++) {
374             flag = 0;
375             for (j = 0; j < key.length() - i - 1; j++) {
376                 if (keygroup[j] > keygroup[j + 1]) {
377                     temp1 = keygroup[j];
378                     keygroup[j] = keygroup[j + 1];
379                     keygroup[j + 1] = temp1;
380                     flag = 1;
381                 }
382             }
383             if (flag == 0)
384                 break;
385         }
386
387         //设置临时数组
388         for (i = 0; i < key.length(); i++) {
389
```

```
390             tem[i] = keygroup[i];
391         }
392
393         //对应的密钥顺序
394         cout << endl << "---------------------------------------" << endl;
395         cout << "key = (d, f)" << endl;
396         cout << "d = " << key.length() << "   ";
397         cout << "f = (";
398         for (i = 0; i < key.length(); i++) {
399             for (j = 0; j < key.length(); j++) {
400                 if (keygroup[j] == key[i] && tem[j] != '#') {
401                     k[i] = j + 1;
402                     tem[j] = '#';     //剥离重复的字母
403                     cout << " " << k[i];
404                     break;
405                 }
406             }
407         }
408         cout << " )" << endl;
409         cout << "---------------------------------------" << endl << endl;
410         return p;
411 }
412
413 void encryColumn(string st, string key) {       //加密
414
415         int flag = 0;
416         int k[20] = { 0 };
417         int t, i, j, m, n, q;
418         char tem[50] = { 0 };
419         char cipher[50][50] = { 0 };
420         char ch[200] = { 0 };
421         int *p;
422         ofstream fout;
423
424         //明文不是key的整数倍补'x'
425         t = st.length();
426         if (t%key.length() != 0)
427             t = ((t / (key.length()) + 1)*key.length());
428         for (i = 0; i < t; i++) {
429             if (i < st.length())
430                 tem[i] = st[i];
431             else
432                 tem[i] = 'x';
433         }
434
435         //打印plain matrix
436         cout << "Plaintext Matrix" << endl << endl;
437         m = 0;
438         n = (strlen(tem)) / (key.length());
439         for (i = 0; i < n; i++) {
```

```
440
441         for (j = 0; j < key.length(); j++) {
442             cipher[i][j] = tem[m];
443             cout << " " << cipher[i][j];
444             m++;
445         }
446         cout << endl;
447     }
448
449     p = wordKey(key, k);
450     //加密
451     cout << "CipherText:";
452     q = 0;
453     for (i = 0; i < key.length(); i++) {
454         for (j = 0; j < n; j++) {
455             cout << cipher[j][(p[i] - 1)];
456             ch[q] = cipher[j][(p[i] - 1)];
457             q++;
458         }
459     }
460     fout.open("Columnciphertext.txt"); //打开Columnciphertext.txt文件
461     fout << ch;
462     fout.close();
463     cout << endl << endl;
464     cout << "---------------------------------------" << endl;
465     cout << "Saved in the Columnciphertext.txt";
466     cout << endl << endl;
467     system("PAUSE");
468 }
469
470 void encryColumnKeyboard() {       //从键盘获取字符串加密
471     int flag = 0;
472     string st, key;
473     do {
474         fflush(stdin);
475         system("cls");
476         cout << "--------Column Permutation cipher(Encryption)--------" << endl;
477         cout << "               the null letter is 'x'               " << endl;
478         cout << endl << "Please enter the plaintext:";
479         cin >> st;
480         flag = Check(st);
481     } while (flag);
482
483     do {     //输入key
484         fflush(stdin);
485         cout << "Please enter the key: ";
486         cin >> key;
487         flag = Check(key);
488     } while (flag);
489     cout << endl;
```

```
490     cout << "----------------------------------------" << endl << endl;
491     encryColumn(st, key);
492 }
493
494 void encryColumnFile() {        //从文件读取加密
495     string key;
496     char filename[30];
497     int flag = 0;
498     fflush(stdin);
499     system("cls");
500     cout << "-------------Column Permutation cipher(Encryption)------------" << endl;
501     cout << endl << "Please enter the file name that you want to encryption:" << endl;
502     cout << "File name: ";
503     cin >> filename;
504     strcat(filename, ".txt");
505     fstream openfile(filename);
506     if (!openfile)
507     {
508         cout << "Open failed!" << endl;
509         exit(1);
510     }
511     char * fn = filename;
512     string st;
513     st = readFileIntoString(fn);
514     cout << filename << ": ";
515     cout << st << endl;
516
517     cout << endl;
518     do {     //输入key
519         fflush(stdin);
520         cout << "Please enter the key: ";
521         cin >> key;
522         flag = Check(key);
523     } while (flag);
524
525     cout << endl;
526     cout << "----------------------------------------" << endl << endl;
527     encryColumn(st, key);
528 }
529
530 void decryColumn(string st, string key) {
531     int k[20] = { 0 };
532     int  i, j, m, n, q;
533     char plain[50][50] = { 0 };
534     char ch[200] = { 0 };
535     int *p;
536     ofstream fout;
537
538
539     //判断是否符合要求
```

```
    if (((st.length()) % (key.length())) != 0) {
        cout << endl << "Error!The length of the ciphertext must be an integral multiple of the length of the key!" << endl << endl;
        exit(-1);
    }

    p = wordKey(key, k);

    //组建明文序列

    m = 0;
    n = (st.length()) / (key.length());
    for (i = 0; i < key.length(); i++) {

        for (j = 0; j < n; j++) {
            plain[j][(p[i] - 1)] = st[m];
            m++;
        }
    }

    //打印plain matrix

    cout << "Plaintext Matrix" << endl << endl;
    for (i = 0; i < n; i++) {
        for (j = 0; j < key.length(); j++) {
            cout << "  " << plain[i][j];
        }
        cout << endl;
    }
    cout << endl;

    //输出明文
    q = 0;
    cout << "Plaintext:";
    for (i = 0; i < n; i++) {
        for (j = 0; j < key.length(); j++) {
            cout << plain[i][j];
            ch[q] = plain[i][j];
            q++;
        }
    }
    fout.open("Columnplaintext.txt");  //打开Columnplaintext.txt文件
    fout << ch;
    fout.close();
    cout << endl << endl;
    cout << "----------------------------------------" << endl;
    cout << "Saved in the Columnplaintext.txt";
    cout << endl << endl;
    system("PAUSE");

}
```

```
590
591 void decryColumnKeyboard() {
592     int flag = 0;
593     string st, key;
594     do {
595         fflush(stdin);
596         system("cls");
597         cout << "--------Column Permutation cipher(Decryption)--------" << endl;
598         cout << endl << "Please enter the ciphertext:";
599         cin >> st;
600         flag = Check(st);
601     } while (flag);
602
603     do {    //输入key
604         fflush(stdin);
605         cout << "Please enter the key: ";
606         cin >> key;
607         flag = Check(key);
608     } while (flag);
609     decryColumn(st, key);
610 }
611
612 void decryColumnFile() {  //从文件读取解密
613     string key;
614     char filename[30];
615     int flag = 0;
616     fflush(stdin);
617     system("cls");
618     cout << "------------Column Permutation cipher(Decryption)------------" << endl;
619     cout << endl << "Please enter the file name that you want to decryption:" << endl;
620     cout << "File name: ";
621     cin >> filename;      //打开想解密的文件名
622     strcat(filename, ".txt");
623     fstream openfile(filename);
624     if (!openfile)
625     {
626         cout << "Open failed!" << endl;
627         exit(1);
628     }
629     char * fn = filename;
630     string st;
631     st = readFileIntoString(fn);
632     cout << filename << ": ";
633     cout << st << endl;
634
635     do {    //输入key
636         fflush(stdin);
637         cout << "Please enter the key: ";
638         cin >> key;
639         flag = Check(key);
```

```
640     } while (flag);
641
642     decryColumn(st, key);
643 }
644
645 void menue() {
646     string strchoose;
647     char getchoose, choose1, choose2;
648     while (1) {
649         getchoose = mainMenue();  //调用主菜单，选择加密解密
650         switch (getchoose) {
651         case '1':    //加密
652             choose1 = menueOptions();
653             switch (choose1) {
654             case '1':    //Vigenere Cipher
655                 choose2 = options();  //  选择加密方式
656                 switch (choose2) {
657                 case '1':
658                     encryVigenereKeyboard();
659                     break;
660                 case '2':
661                     //这里从文件读取加密
662                     encryVigenereFile();
663                     break;
664                 default:
665                     break;
666                 }
667
668                 break;
669             case '2':    //Column permutation cipher
670                 choose2 = options();  //  选择加密方式
671                 switch (choose2) {
672                 case '1':
673                     encryColumnKeyboard();
674                     break;
675                 case '2':
676                     //这里从文件读取加密
677                     encryColumnFile();
678                     break;
679                 default:
680                     break;
681
682                 }
683                 break;
684             default:
685                 break;
686             }
687             break;
688         case '2':        //解密
689             choose1 = menueOptions(); //调用主菜单,选择加密解密
```

```
690                 switch (choose1) {
691                 case '1':    //Vigenere Cipher
692                     choose2 = options();
693                     switch (choose2) {
694                     case '1':
695                         decryVigenereKeyboard();
696                         break;
697                     case '2':
698                         //这里从文件读取解密
699                         decryVigenereFile();
700                         break;
701                     default:
702                         break;
703 
704                     }
705                     break;
706 
707                 case '2':    //Column permutation cipher
708                     choose2 = options();  //  选择解密方式
709                     switch (choose2) {
710                     case '1':
711                         decryColumnKeyboard();
712                         break;
713                     case '2':
714                         //这里从文件读取解密
715                         decryColumnFile();
716                         break;
717                     default:
718                         break;
719 
720                     }
721                     break;
722                 default:
723                     break;
724                 }
725                 break;
726             case '3':    //退出
727                 cout << endl;
728                 return;
729 
730             }
731         }
732 }
733 
734 int main() {
735 
736     system("color 2F");
737     cout << "=============================================" << endl;
738     cout << "----------Encryption/Decryption System----------" << endl;
739     cout << "=============================================" << endl;
```

```
740         menue();
741
742         return 0;
743     }
```

七、 思考题

1. **Vigenere 密码的原理是什么?**

   答：Vigenere 是多表替代，首先构造多个密文字母表，然后用密文字母表中的字母来代替明文字母，明文中的每一个字母都有多种可能的字母来代替。Vigenere 密码的代替规则是用明文字母在 Vigenere 方阵中的列和密钥字母在 Vigenere 方阵中的行的交叉点处的字母来代替该明文字母。

2. **Vigenere 密码的主要缺陷有哪些?**

   答：Vigenere 密码使用循环的密钥进行加密。如果两个相同的明文段加密成两个相同的密文段，它们的位置间距假设为 δ，则 δ=0(mod m)。反过来，如果在密文中观察到两个相同的长度至少为 3 的密文段，则他们对应了相同的明文串，由此根据两个相同密文段间隔的距离可猜测出密钥长度，这给破译者带来了很大的方便。

3. **对 Vigenere 密码的分析方法主要思想?**

   答：Kasiski 测试法：搜索长度至少为 3 的相同密文段，记下其离起始点的那个密文段的距离假如得到如下几个距离δ_1, δ_2,…,那么，可以猜测 m 为这些δ_i的最大公因子的因子。

4. **对 Vigenere 密码的改进方法是什么?**

   答：增加密钥长度，减少密钥的循环使用。

5. **Column permutation cipher 的原理是什么?**

   答：Column Permutation Cipher 属于置换密码的一种，其原理如下：把明文按行填充排列成一个矩阵，其中不足部分用符号Φ填充。然后按预定顺序按列读取形成密文。

6. **给定关键字为“experiment”,加密矩阵将包括几列，以及列置换的次序是什么?**

   答：加密矩阵将包括 10 列，列置换次序为：1-10-7-2-8-4-5-3-6-9.

7. **Column permutation cipher 的安全性增强方法是什么?**

   答：进行双重换位加密。先用列换位法将明文加密，然后再次利用列换位法将第一次换位加密的密文加密。经过两次换位后，明文字母的位置完全被打乱了，从而可以增强安全性。

八、 结束语

通过实践课程，我深入了解了 Polyalphabetic Cipher 和 Transposition Cipher 的区别，加深了我对古典密码体制的理解，在实践过程中不仅验证了理论知识，还加强了实验手段和实践技能，培养了我自主分析问题、解决问题、应用知识的能力和创新精神，获益匪浅。我认为实践课程对深入理解密码学的内涵有很大帮助，理论与实践相结合，对我的综合素质有很大的提升。同时，通过实践方式来学习古典密码，为以后学习现代密码奠定了很好的基础。

九、 参考文献

1.Douglas R.Stinson:《Cryptography Theory and Practice》(Third Edition)，电子工业出版社，2008

2.张焕国，唐明著:《密码学引论》(第三版)，武汉大学出版社，2015

3.郑东 李祥学等著:《密码学——密码算法与协议》(第二版),电子工业出版社，2013

4.Michael Welschenbach:《Cryptography in C and C++》(Second Edition),机械工业出版社，2015

实验成绩

考查内容	分数	得分
做好实验内容的预习，写出预习报告	10	
了解实验题目的调试方法	10	
按实验要求预先设计好程序	10	
认真记录实验数据并分析实验结果	10	
实验后按要求书写实验报告，记录实验用数据及运行结果	30	
创新能力强，在实验中设计的程序有一定的通用性，算法优化	20	
实验过程中，具有严谨的学习态度，认真、踏实、一丝不苟的科学作风	10	

密码学基础实验报告

东北大学

实验名称	DES 的编程实现
班　级	软信-1503
学　号	20155362
姓　名	薛旗
日　期	2017.04.20
成　绩	
评 阅 人	

软件学院

一、 实验目的与意义

实验目的：

(1)通过编写程序实现 DES 密码，并加密一个文件；

(2)理解分组密码的基本理论；

(3)掌握 DES 的基本原理，了解算法的详细步骤。

实验意义：

(1)通过实践，亲自编程体验置换、替代、代数等多种密码技术的综合运用情况，体味 Shannon 所阐述的设计密码的思想；

(2)通过实践，将理论知识与实践相结合，培养分析问题、解决问题、应用问题的能力，培养创新精神，提高综合素质。

二、 实验环境

操作系统：Windows 10 专业版

调试软件：Microsoft Visual Studio Community 2017

版本号：4.7.02046

上机地点：信息学馆 B405

机器台号：随机

三、 实验的预习内容

预习内容：

(1)DES 简介：DES 是 Data Encryption Standard 的缩写，是美国国家标准局于 1977 年公布的由 IBM 公司研制的加密算法，并被作为非机要部门使用的数据加密标准。DES 算法是应用最广泛的一种分组密码算法，对密码理论的发展和应用起了重大作用。DES 综合运用了置换、代替、代数等多种密码技术，是面向二进制的密码算法，因而能够加解密任何形式的计算机数据。

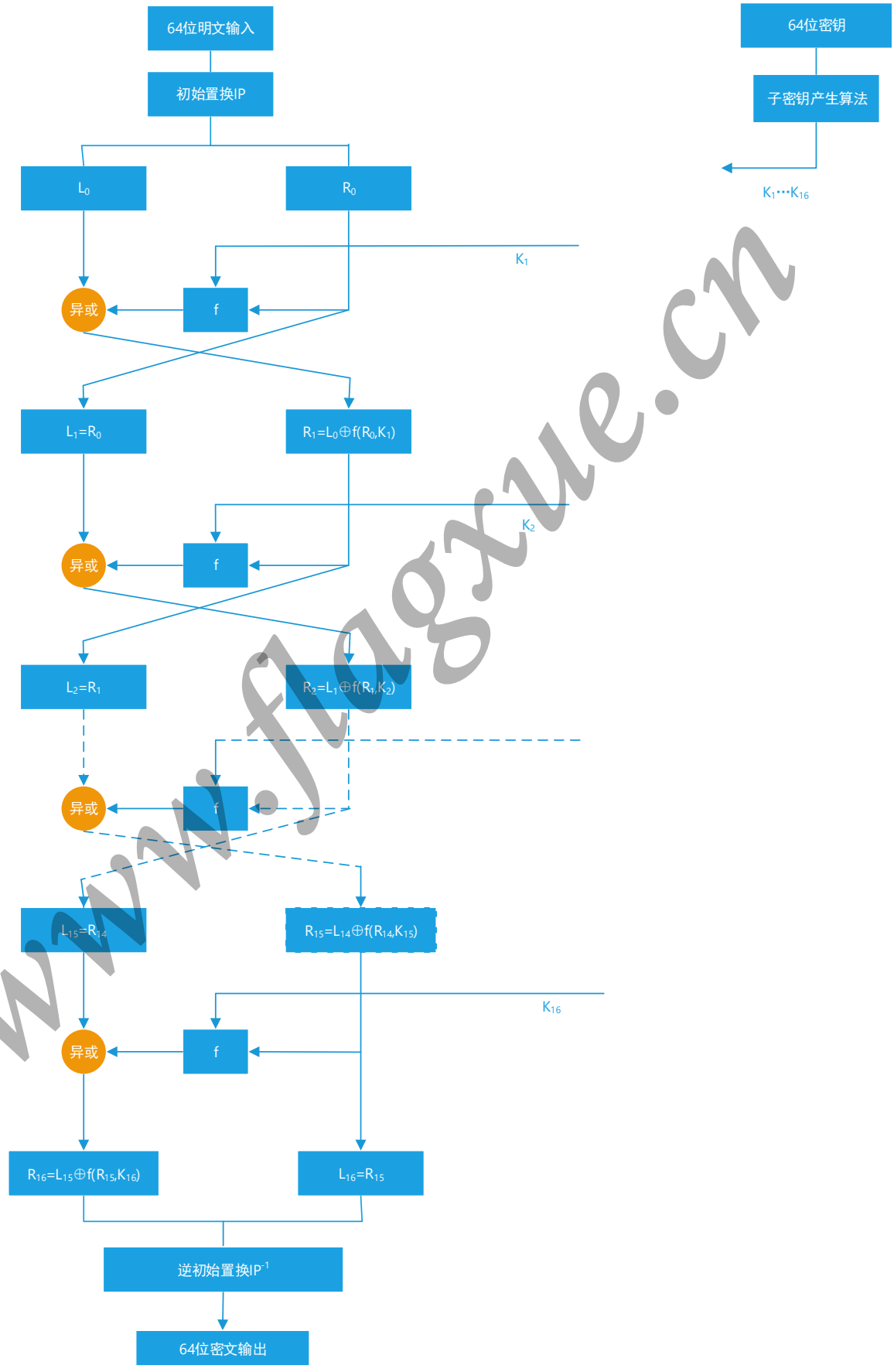
(2)DES 加密算法：DES 算法中数据以 64 比特分组进行加密，有效密钥长度为 56 位。它的加密算法与解密算法相同，只是解密子密钥与加密子密钥的使用顺序刚好相反。

实验思路：

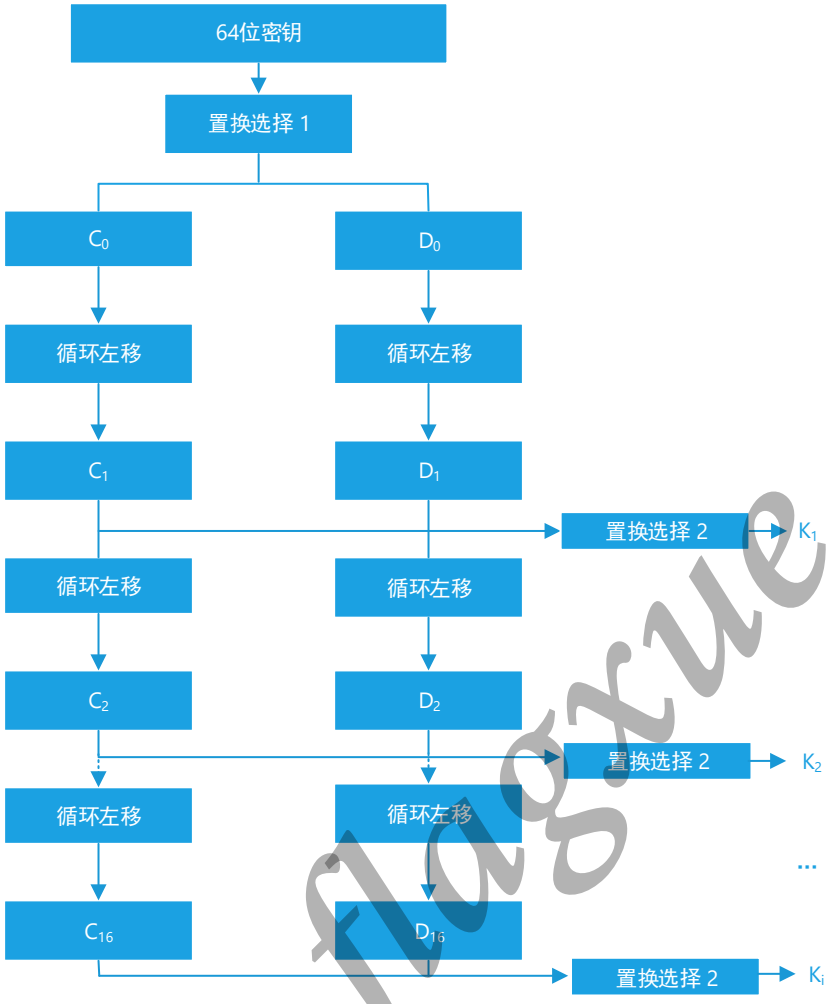
实验最关键的地方在于对明文的分块处理、二进制转化、盒子操作、以及子密钥的产生和十六次加密迭代使用。因此，在进行实验时目标明确，先处理密钥，生成16个48位的子密钥，再处理明文，接着实现盒子操作与加密函数f，然后迭代16次，产生64位数据组，之后对数据组进行处理后，经过逆初始置换，得到64位明文，至此一组加密过程全部结束。在本实验中，定义以下规则：(1)将初始明文/密文分组并依次转化为二进制比特流之后，若分组内比特流长度不足64位，则补0。(2)对密钥的操作处理：输入7位英文字符，得到其对应的ASCII码，将ASCII码用二进制表示，最终得到56位二进制比特流，将此作为DES的56位密钥。之后在第8，16，24，32，40，48，56，64位处插入奇偶校验位，合成64位密钥。

程序框图：（见下页）

DES 的整体结构：



子密钥的产生:



四、 实验的步骤与调试方法

实验步骤:

(1)64 位密钥经过子密钥产生算法产生 16 个子密钥：K_1，K_2，…，K_{16}，分别供第一次，第二次，…，第十六次加密迭代使用。

(2)64 位明文首先经过初始置换 IP(Initial permutation),将数据打乱重新排列并分成左右两半。左边 32 位构成L_0，左边 32 位构成R_0.

(3)由加密函数 f 实现子密钥K_1对R_0的加密，结果为 32 位的数据组 $f(R_0, K_1)$。$f(R_0, K_1)$再与L_0模 2 相加，又得到一个 32 位的数据组

$L_0 \oplus f(R_0, K_1)$。以$L_0 \oplus f(R_0, K_1)$作为第二次加密迭代的R_1，以R_0作为第二次加密迭代的L_1.至此，第一次加密迭代结束。

(4)第二次加密迭代至第十六次加密迭代分别用于子密钥K_2，…K_{16}进行，其过程与第一次加密迭代相同。

(5)第十六次加密迭代结束后，产生一个 64 位的数组。以其左边 32 位作为R_{16}，以其右边 32 位作为L_{16}，两者合并再经过逆初始置换IP^{-1}，将数据重新排列，便得到 64 位密文。至此加密过程全部结束。

步骤	说明	调用函数
1	加密/解密主函数	main()
2	检查输入的密钥是否合法	Check(string)
3	从文件读取加密	readFileIntoString(char *)
4	实现明/密文块和二进制转化	plainTrans(string, int, int(&)[][64], int)
5	密钥二进制转换	keyTrans(string, int[64])
6	构造子密钥	makeKey(int *, int(&)[16][48], int)
7	DES 初始置换	initial(int(&)[][64], int(&)[][64], string)
8	扩展置换(EBox)	funEBox(int [48], int [32])
9	代替函数组(SBox)	funSBox(int[][48], int[48], int[32], int)
10	置换运算(PBox)	funPBox(int[32], int[32])
11	逆初始置换IP^{-1}	funFinalTP(int[32],int[32],int[64],int,int(&)[][64])
12	输出加/解密结果并储存到文件	outputchar(int(&)[800][64], int, int)

调试方法：

(1)从文件读取解密时，发现不能将加密后的密文全部读取，只能读取少部分密文。通过分析，可能是加密的文件中存在因为加密而产生的文件终止符(EOF),因此必须想办法将其转化为可读取的二进制形式。在编程过程中，通过二进制读取文件的方式读取文件可解决此问题。读取格式如下：ifstream ifile(filename, _IOSbinary)，其中_IOSbinary 指以二进制方式打开文件。

(2)程序编写完成后，在运行程序时发现，当输入的明文过多时(大于 500 字符)会出现程序异常退出的情况。通过分析，发现是变量空间分配不足导致的。将变量空间扩大即可解决此问题。

五、 实验数据与实验结果

加密：

实验数据：

Plaintext:NortheasternUniversity

Key:program

File:DESplaintext.txt

加密结果：

Stream Ciphertext:

00010011010101001011111100101000101000001101110101100001100010010

00101011000010010010011001101001010110110000101110001101110010011

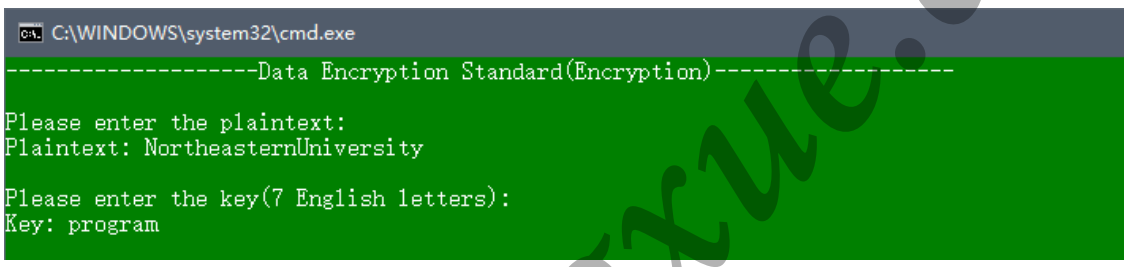
00111111000010000010010101110101010001111000010100110101111111001

Character Ciphertext:

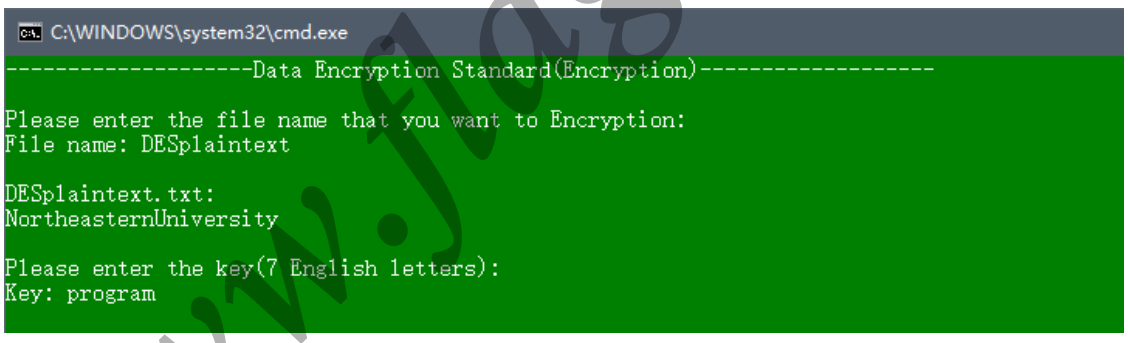
T? 犦 a?　　&i[

嵥 %uG?

实验结果展示：

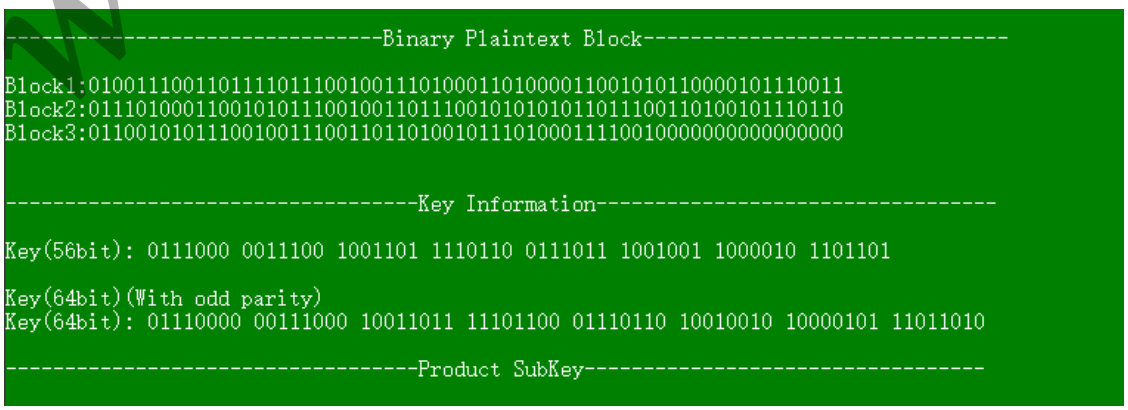


(OR)



.

.



.

.

```
------------------------------SubKey Information-----------------------------

SubKey1: 001111011000111111100110100110111001111111001000
SubKey2: 101010110011110110011000010011110000101100100011
SubKey3: 010111000010111011101101110111101100000111101100
SubKey4: 110100111111110000011000000000001101111111001001
SubKey5: 010011001010111111100110110110101011010000110001
SubKey6: 111100101111110000001111111010110100111100101000
SubKey7: 011010011010011101100010000110000111101100011010
SubKey8: 111000001101110010111111111110101010100000010100
SubKey9: 100011001101011011100010100010111100101011010100
SubKey10: 111100100101101101111111001010001111001111010001
SubKey11: 101011001111001101000001101110110000010000001101
SubKey12: 000000110101111101111111111001010011100111000010
SubKey13: 111011010111000111010001001101000110001110101101
SubKey14: 000101111100111111101001111110010000110001100011
SubKey15: 110110110111000110010011111000110101000110011011
SubKey16: 000111110110101000101111101001011100010110001011
```

.

.

```
------------------------------Final Result ------------------------------

Stream Ciphertext:

0001001101010100101111110010100010100000110111010110000110001001
0010101100001001001001100110100101011011000010111000110111001011
0011111100001000001001010111010101000111100001010011010111111001


Character Ciphertext:

□T? 犦 a?       &i[□嵥 %uG?


--------------------------------
Saved in the DESciphertext.txt
```

(子密钥产生过程和明文分组加密过程在此处省略，未展示)

解密：

实验数据：

Ciphertext:

T? 犦 a?　　&i[

嵥 %uG?

File：DESciphertext.txt

Key:program

解密结果：

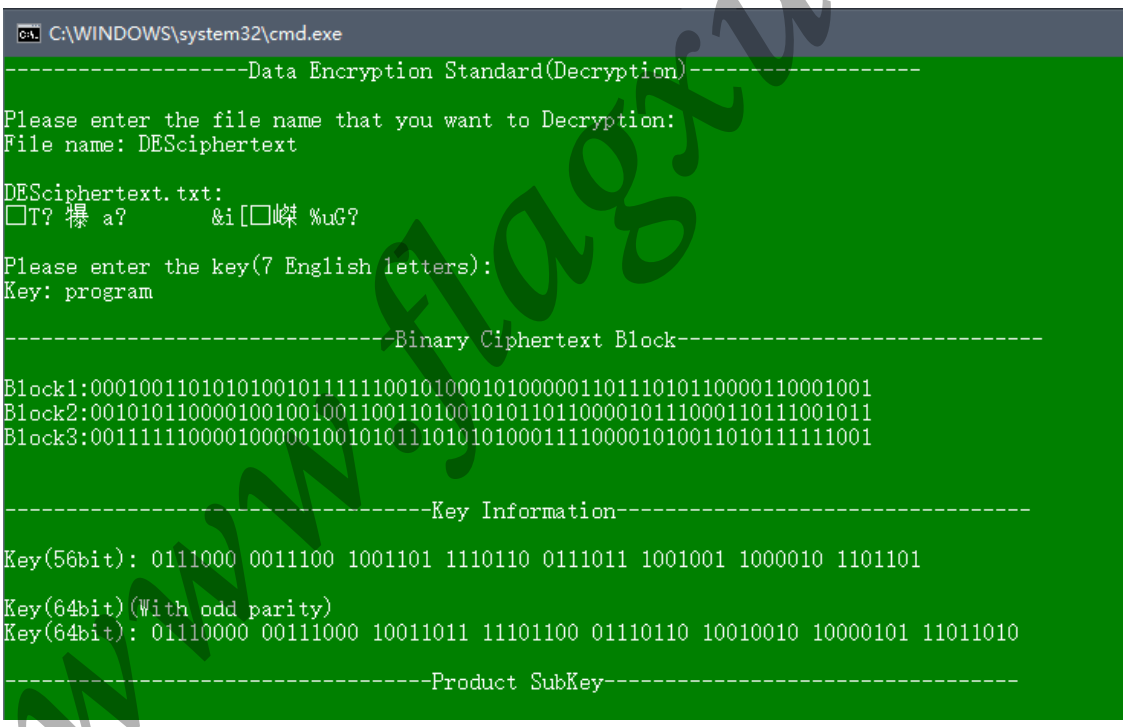
Stream Ciphertext:

01001110011011110111001001110100011010000110010101100001011100110111010001100101011100100111011001010101011011100110100101110110011001010111001001110011011010010111010001111001000000000000000000

Character Ciphertext:

NortheasternUniversity

实验结果展示：



.

.

```
------------------------------SubKey Information-----------------------------

SubKey1: 000111110110101000101111101001011100010110001011
SubKey2: 110110110111000110010011110001101010001100111011
SubKey3: 000101111100111111101001111100100001100011000011
SubKey4: 111011010111000111010001001101000110001110101101
SubKey5: 000000110101111101111111110010100111001110000110
SubKey6: 101011001111001101000001101110110000010000001101
SubKey7: 111100100101101101111110010100011110011110010001
SubKey8: 100011001101011011100010100010111100101011010100
SubKey9: 111000001101110010111111111101010101000000110100
SubKey10: 011010011010011101100010000110000111101100011010
SubKey11: 111100101111110000001111111010110100111100101000
SubKey12: 010011001010111111100110110110101011010000110001
SubKey13: 110100111111110000011000000000001101111111001001
SubKey14: 010111000010111011101101110111011000001111101100
SubKey15: 101010110011110110011000010011100001011001000111
SubKey16: 001111011000111111001101001101110011111101001000
```

.

.

```
------------------------------Final Result ------------------------------

Stream Ciphertext:

010011100110111101110010011101000110100001100101011000010111 0011
011101000110010101110010011011100101010101101110011010010111 0110
011001010111001001110011011010010111010001111001000000000000 0000


Character Ciphertext:

NortheasternUniversity


--------------------------------
Saved in the DESplaintext.txt
```

(子密钥产生过程和密文分组解密过程在此处省略，未展示)

附：选择菜单

六、 实验用程序清单

```
#include<iostream>
#include<string>
#include<stdlib.h>
#include<windows.h>
#include<math.h>
#include<fstream>
#include<sstream>

using namespace std;
int Check(string);
int *keyTrans(string, int[64]);
void plainTrans(string, int, int(&)[800][64], int); //将明文分块，并转化为二进制存储形式
void initial(int(&)[800][64], int(&)[800][64], string);
void makeKey(int *, int(&)[16][48], int);
void funEBox(int extrightPlain[48], int tempright[32]);
void funSBox(int[][48], int[48], int[32], int);
void funPBox(int[32], int[32]);
void funFinalTP(int[32], int[32], int[64], int, int(&)[800][64]);
void outputchar(int(&)[800][64], int, int); //由ASCII码输出字符
string readFileIntoString(char * filename);


const static int IP[64] = {         //初始置换IP(针对明文)
    58, 50, 42, 34, 26, 18, 10, 2,
    60, 52, 44, 36, 28, 20, 12, 4,
    62, 54, 46, 38, 30, 22, 14, 6,
    64, 56, 48, 40, 32, 24, 16, 8,
    57, 49, 41, 33, 25, 17,  9, 1,
    59, 51, 43, 35, 27, 19, 11, 3,
    61, 53, 45, 37, 29, 21, 13, 5,
```

```
    63, 55, 47, 39, 31, 23, 15, 7
};

const static int PC_1[56] = {       //密钥置换，将64位密钥置换为56位
    57, 49, 41, 33, 25, 17, 9,
    1, 58, 50, 42, 34, 26, 18,
    10, 2, 59, 51, 43, 35, 27,
    19, 11, 3, 60, 52, 44, 36,
    63, 55, 47, 39, 31, 23, 15,
    7, 62, 54, 46, 38, 30, 22,
    14, 6, 61, 53, 45, 37, 29,
    21, 13, 5, 28, 20, 12, 4
};

const static int PC_2[48] = {       //密钥压缩置换，将56位密钥置换为48位
    14, 17, 11, 24, 1, 5,
    3, 28, 15, 6, 21, 10,
    23, 19, 12, 4, 26, 8,
    16, 7, 27, 20, 13, 2,
    41, 52, 31, 37, 47, 55,
    30, 40, 51, 45, 33, 48,
    44, 49, 39, 56, 34, 53,
    46, 42, 50, 36, 29, 32
};

const static int EBox[48] = {       //扩展置换，将32位扩展为48位
    32, 1, 2, 3, 4, 5,
    4, 5, 6, 7, 8, 9,
    8, 9, 10, 11, 12, 13,
    12, 13, 14, 15, 16, 17,
    16, 17, 18, 19, 20, 21,
    20, 21, 22, 23, 24, 25,
    24, 25, 26, 27, 28, 29,
    28, 29, 30, 31, 32, 1
};

const static int SBox[8][4][16] = {
    {   //Sbox_1
        { 14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7 },
        { 0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8 },
        { 4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0 },
        { 15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13 }
    },
    {   //Sbox_2
        { 15,1,8,14,6,11,3,4,9,7,2,13,12,0,5,10 },
        { 3,13,4,7,15,2,8,14,12,0,1,10,6,9,11,5 },
        { 0,14,7,11,10,4,13,1,5,8,12,6,9,3,2,15 },
        { 13,8,10,1,3,15,4,2,11,6,7,12,0,5,14,9 }
    },
    {   //Sbox_3
```

```
81              { 10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8 },
82              { 13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1 },
83              { 13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7 },
84              { 1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12 }
85      },
86      {   //Sbox_4
87              { 7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15 },
88              { 13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9 },
89              { 10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4 },
90              { 3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14 }
91      },
92      {   //Sbox_5
93              { 2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9 },
94              { 14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6 },
95              { 4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14 },
96              { 11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3 }
97      },
98      {   //Sbox_6
99              { 12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11 },
100             { 10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8 },
101             { 9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6 },
102             { 4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13 }
103     },
104     {   //Sbox_7
105             { 4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1 },
106             { 13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6 },
107             { 1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2 },
108             { 6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12 }
109     },
110     {   //Sbox_8
111             { 13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7 },
112             { 1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2 },
113             { 7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8 },
114             { 2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11 }
115     }
116 };
117
118 const static int PBox[32] = {
119     16,  7,  20,  21,  29,  12,  28,  17,
120     1,  15,  23,  26,  5,  18,  31,  10,
121     2,  8,  24,  14,  32,  27,  3,  9,
122     19,  13,  30,  6,  22,  11,  4,  25
123 };
124
125 const static int  FT[64] = {
126     40,  8,  48,  16,  56,  24,  64,  32,
127     39,  7,  47,  15,  55,  23,  63,  31,
128     38,  6,  46,  14,  54,  22,  62,  30,
129     37,  5,  45,  13,  53,  21,  61,  29,
130     36,  4,  44,  12,  52,  20,  60,  28,
```

```
131      35, 3, 43, 11, 51, 19, 59, 27,
132      34, 2, 42, 10, 50, 18, 58, 26,
133      33, 1, 41,  9, 49, 17, 57, 25
134  };
135
136  int Check(string s) {      //检查输入的字符串是否为纯英文(检查密钥)
137      int i = 0;
138      while (s[i] != '\0') {
139          if (!((s[i] >= 'a'&&s[i] <= 'z') || (s[i] >= 'A'&&s[i] <= 'Z'))) {
140              cout << "Error!Please enter again!" << endl;
141              Sleep(1000);
142              return 1;
143          }
144          i++;
145      }
146      return 0;
147  }
148
149  int *keyTrans(string key, int keytemp[64]) {//密钥二进制转换并输出
150      int i, k, t, m, n, q;
151      int block;
152      int flag = 0;
153      int temp[56] = { 0 };
154      int *p = keytemp;
155      m = 0;
156      for (k = 0; k < 7; k++) {
157          t = key[6 - k];          //二进制转换
158          for (i = 0; i < 8; i++) {
159
160              temp[56 - 1 - m] = t % 2;
161              t /= 2;
162              m++;
163          }
164      }
165      cout << endl;
166
167      cout << "-----------------Key Information-----------------" << endl << endl;
168      cout << "Key(56bit): ";        //56位密钥
169      for (i = 0; i < 56; i++) {
170          cout << temp[i];
171          if ((i + 1) % 7 == 0)
172              cout << " ";
173      }
174      cout << endl << endl;
175
176      n = 0;
177      m = 0;
178      q = 0;
179      cout << "Key(64bit)(With odd parity)" << endl; //显示有奇偶校验位的64位密钥
180      cout << "Key(64bit): ";
```

```
    for (i = 0; i < 56; i++) {
        cout << temp[i];
        keytemp[q] = temp[i];
        q++;
        if (temp[i] == 0) {
            n++;
        }

        if (((m + 1) % 7 == 0) && (n % 2 != 0)) {
            cout << "1 ";
            keytemp[q] = 1;   //将64位密钥存入数组
            q++;
            m++;
            n = 0;
            continue;
        }
        else if (((m + 1) % 7 == 0) && (n % 2 == 0)) {
            cout << "0 ";
            keytemp[q] = 0;
            q++;
            m++;
            n = 0;
            continue;
        }
        m++;
    }
    cout << endl << endl;
    return p;
}

void plainTrans(string st, int block, int(&plaintext)[800][64], int way) {//实现明文/密文分块和二进制转化
    unsigned int i, k, t, m, n, j;
    int flag = 0;
    int temp[56] = { 0 };

    //将明文分块，存入二维数组
    n = 0;
    k = st.length() % 8;
    if (k == 0)
        k = 8;
    m = block;
    for (i = 0; i < st.length(); i++) {
        t = (unsigned int)st[(st.length() - 1 - i)];
        if (t > 256) {
            t = t - 4294967040;        //若字符的ASCII超出范围，则减去过余码
        }
        for (j = 0; j < 8; j++) {
            if (n == 64) {
                n = 0;
                m--;
```

```
231                 }
232                 if (m != block) {
233                     plaintext[m][(64 - 1 - n)] = t % 2;
234                 }
235                 else {
236                     plaintext[m][(8 * k - 1 - n)] = t % 2;
237                     if (n == (8 * k - 1)) {    //明文最后一个分块，不足64位补0
238                         n = -1;
239                         m--;
240                     }
241                 }
242                 t /= 2;
243                 n++;
244             }
245         }
246         cout << endl;
247 
248         //输出分块明文比特流
249         if (way == 0) {
250             cout << "--------------Binary Plaintext Block------------" << endl << endl;
251         }
252         else {
253             cout << "--------------Binary Ciphertext Block------------" << endl << endl;
254         }
255 
256         for (i = 0; i < block + 1; i++) {
257             cout << "Block" << i + 1 << ":";
258             for (j = 0; j < 64; j++) {
259                 cout << plaintext[i][j];
260             }
261             cout << endl;
262         }
263         cout << endl;
264 }
265 
266 void initial(int(&plainTemp)[800][64], int(&initialPlain)[800][64], string st) {    //DES初始置换
267         int block;
268         int i, j;
269         block = (st.length() - 1) / 8;
270         for (i = 0; i < block + 1; i++) {
271             for (j = 0; j < 64; j++) {
272                 initialPlain[i][j] = plainTemp[i][(IP[j] - 1)];//查表，初始化明文块
273             }
274         }
275 }
276 
277 void makeKey(int *key, int(&subKey)[16][48], int way) {       //构造子密钥
278         int i, j, l, r, k, flag, p, q;
279         int temp0[56] = { 0 };
280         int templeft[28] = { 0 };
```

```
    int tempright[28] = { 0 };
    int combine[56] = { 0 };
    cout << "------------------Product SubKey------------------" << endl << endl;
    for (i = 0; i < 56; i++) {//  PC-1
        temp0[i] = key[(PC_1[i] - 1)];
        if (i < 28) {
            templeft[i] = key[(PC_1[i] - 1)];
        }
        else {
            tempright[i - 28] = key[(PC_1[i] - 1)];
        }
        //PC-1完成，输出结果用于检验
    }

    //输出PC-1
    cout << endl << "PC-1: ";
    for (i = 0; i < 56; i++) {
        cout << temp0[i];
    }
    cout << endl;

    //输出L0和R0
    cout << "L(0): ";
    for (i = 0; i < 28; i++) {
        cout << templeft[i];
    }

    cout << endl << "R(0): ";
    for (i = 0; i < 28; i++) {
        cout << tempright[i];
    }
    cout << endl << endl;


    for (i = 0; i < 16; i++) {//循环左移
        if (i == 0 || i == 1 || i == 8 || i == 15) {
            l = templeft[0];
            r = tempright[0];
            for (j = 0; j < 27; j++) {
                templeft[j] = templeft[j + 1];
                tempright[j] = tempright[j + 1];
            }
            templeft[j] = l;
            tempright[j] = r; //移完一次 储存

                                //输出L和R
            cout << "L(" << i + 1 << "): ";
            for (p = 0; p < 28; p++) {
                cout << templeft[p];
            }
```

```
331                 cout << endl << "R(" << i + 1 << "): ";
332                 for (p = 0; p < 28; p++) {
333                     cout << tempright[p];
334                 }
335 
336                 //组合L和R
337                 for (flag = 0; flag < 56; flag++) {
338                     if (flag < 28) {
339                         combine[flag] = templeft[flag];
340                     }
341                     else {
342                         combine[flag] = tempright[flag - 28];
343                     }
344                 }
345                 //PC-2
346                 cout << endl << "SubKey" << i + 1 << ": ";
347                 if (way == 0) {  //加密
348                     for (flag = 0; flag < 48; flag++) {
349                         subKey[i][flag] = combine[(PC_2[flag] - 1)];
350                         cout << subKey[i][flag];
351                     }
352                 }
353                 else {  //  解密子密钥顺序相反
354                     for (flag = 0; flag < 48; flag++) {
355                         subKey[15 - i][flag] = combine[(PC_2[flag] - 1)];
356                         cout << subKey[15 - i][flag];
357                     }
358                 }
359                 cout << endl << endl;
360             }
361             else {
362                 for (k = 0; k < 2; k++) {
363                     l = templeft[0];
364                     r = tempright[0];
365                     for (j = 0; j < 27; j++) {
366                         templeft[j] = templeft[j + 1];
367                         tempright[j] = tempright[j + 1];
368                     }
369                     templeft[j] = l;
370                     tempright[j] = r; //移完两次 储存
371                 }
372 
373                 //输出L和R
374                 cout << "L(" << i + 1 << "): ";
375                 for (p = 0; p < 28; p++) {
376                     cout << templeft[p];
377                 }
378                 cout << endl << "R(" << i + 1 << "): ";
379                 for (p = 0; p < 28; p++) {
380                     cout << tempright[p];
```

```
381                 }
382
383
384                 for (flag = 0; flag < 56; flag++) {
385                     if (flag < 28) {
386                         combine[flag] = templeft[flag];
387                     }
388                     else {
389                         combine[flag] = tempright[flag - 28];
390                     }
391                 }
392                 //PC-2
393
394                 cout << endl << "SubKey" << i + 1 << ": ";
395                 if (way == 0) {  //加密
396                     for (flag = 0; flag < 48; flag++) {
397                         subKey[i][flag] = combine[(PC_2[flag] - 1)];
398                         cout << subKey[i][flag];
399                     }
400                 }
401                 else {  //解密密钥顺序相反
402                     for (flag = 0; flag < 48; flag++) {
403                         subKey[15 - i][flag] = combine[(PC_2[flag] - 1)];
404                         cout << subKey[15 - i][flag];
405                     }
406                 }
407                 cout << endl << endl;
408             }
409         }
410         cout << "-------------SubKey Information-------------" << endl << endl;
411
412         for (i = 0; i < 16; i++) {
413             cout << "SubKey" << i + 1 << ": ";
414             for (flag = 0; flag < 48; flag++) {
415                 cout << subKey[i][flag];
416             }
417             cout << endl;
418         }
419         cout << endl;
420
421 }
422
423 void funEBox(int extrightPlain[48], int tempright[32]) {      //扩展置换(EBox)
424     int i;
425     for (i = 0; i < 48; i++) {
426         extrightPlain[i] = tempright[(EBox[i] - 1)];
427         cout << extrightPlain[i];
428     }
429     cout << endl;
430 }
```

```
431
432 void funSBox(int key[][48], int extrightPlain[48], int out[32], int j) {//代替函数组(SBox)
433     int m, n, p, s, t, k, q;
434     int temp[8][6] = { 0 };
435     int temp2[8] = { 0 };
436
437     cout << endl << "   SubKey(" << j + 1 << "): ";
438     for (m = 0; m < 48; m++) {
439         cout << key[j][m];
440     }
441     cout << endl << "   Key XOR RP: ";
442     p = 0;
443     for (m = 0; m < 8; m++) {
444         for (n = 0; n < 6; n++) {
445             temp[m][n] = (key[j][p]) ^ (extrightPlain[p]);
446             cout << temp[m][n];
447             p++;
448         }
449     }
450
451     for (m = 0; m < 8; m++) {
452         s = (temp[m][0]) * 2 + temp[m][5];
453         t = (temp[m][1] * 8 + temp[m][2] * 4 + temp[m][3] * 2 + temp[m][4]);
454         temp2[m] = SBox[m][s][t];
455
456     }
457
458     //二进制转换
459     m = 0;
460     for (k = 0; k < 8; k++) {
461         t = temp2[7 - k];     //二进制转换
462         for (q = 0; q < 4; q++) {
463
464             out[32 - 1 - m] = t % 2;
465             t /= 2;
466             m++;
467         }
468
469     }
470
471     cout << endl << "   OutPut: ";
472     for (m = 0; m < 32; m++) {
473         cout << out[m];
474     }
475     cout << endl;
476
477 }
478
479 void funPBox(int outSBox[32], int outPBox[32]) {     //置换运算(PBox)
480     int i;
```

```
481     for (i = 0; i < 32; i++) {
482         outPBox[i] = outSBox[(PBox[i] - 1)];
483         cout << outPBox[i];
484     }
485     cout << endl << endl;
486 }
487
488 void funFinalTP(int left[32], int right[32], int finalTrans[64], int i,
489     int j;
490     int tempfinalTrans[64] = { 0 };
491
492     for (j = 0; j < 64; j++) {
493         if (j < 32) {
494             tempfinalTrans[j] = left[j];
495         }
496         else {
497             tempfinalTrans[j] = right[j - 32];
498         }
499     }
500     cout << endl << "Block" << i + 1 << " Ciphertext" << ": ";
501     for (j = 0; j < 64; j++) {
502         finalTrans[j] = tempfinalTrans[(FT[j] - 1)];
503         finalresult[i][j] = finalTrans[j];
504         cout << finalTrans[j];
505     }
506     cout << endl;
507 }
508
509 void outputchar(int(&finalresult)[800][64], int i, int way) {
510     ofstream fout;
511     int temp[8000] = { 0 };
512     int t, j, q, sum, m, n;
513     char ch[10000] = { '\0' };
514     sum = 0;
515     q = 0;
516     for (t = 0; t < i; t++) {
517         for (j = 0; j < 64; j++) {
518             m = 7 - (j % 8);
519             n = pow(2, m);
520             sum = sum + (finalresult[t][j])*n;
521             if (m == 0) {
522                 temp[q] = sum;
523                 q++;
524                 sum = 0;
525             }
526         }
527     }
528
529     //输出ASCII对应的字符
530     cout << endl << endl << "Character Ciphertext: " << endl << endl;
```

```
531         for (m = 0; m < q; m++) {
532             ch[m] = temp[m];
533             cout << ch[m];
534         }
535         if (way == 0) {
536             fout.open("DESciphertext.txt");     //打开DESciphertext.txt文件
537             fout << ch;
538             fout.close();
539             cout << endl << endl << endl;
540             cout << "-------------------------------" << endl;
541             cout << "Saved in the DESciphertext.txt";
542         }
543         else {
544             fout.open("DESplaintext.txt"); //打开DESciphertext.txt文件
545             fout << ch;
546             fout.close();
547             cout << endl << endl << endl;
548             cout << "-------------------------------" << endl;
549             cout << "Saved in the DESplaintext.txt";
550         }
551 
552         cout << endl << endl;
553 
554 }
555 
556 string readFileIntoString(char * filename) {    //从文件读入到string里
557         cin >> filename;
558         strcat(filename, ".txt");
559         fstream openencryfile(filename);
560         if (!openencryfile)
561         {
562             cout << "Open failed!" << endl << endl;
563             exit(1);
564         }
565         ifstream ifile(filename, _IOSbinary);   //文件以二进制形式打开
566         ostringstream buf;          //将文件读入到ostringstream对象buf中
567         char ch;
568         while (buf && ifile.get(ch))
569             buf.put(ch);
570         return buf.str();      //返回与流对象buf关联的字符串
571 }
572 
573 int main() {
574         string key;
575         string st;
576         string strchoose;
577         fstream opendecryfile;
578         fstream openencryfile;
579         char choose;
580         char filename[30];
```

```
581     char * fn = filename;
582     int i, j, block, m, n, flag, way;
583     int *temp;        //指向二进制密钥的指针
584     int keytemp[64] = { 0 };
585     int pltext[800][64] = { 0 };
586     int leftPlain[800][32] = { 0 };
587     int rightPlain[800][32] = { 0 };
588     int extrightPlain[48] = { 0 };
589     int iniPlain[800][64] = { 0 };
590     int subKey[16][48] = { 0 };
591     int outSBox[32] = { 0 };
592     int outPBox[32] = { 0 };
593     int templeft[32] = { 0 };
594     int tempright[32] = { 0 };
595     int temptrans[32] = { 0 };
596     int finalTrans[64] = { 0 };
597     int finalresult[800][64] = { 0 };  //将各block密文组合到一块储存起来
598
599     while (1) {
600         fflush(stdin);
601         system("cls");
602         way = 0;
603         flag = 0;
604         do {
605             fflush(stdin);
606             system("cls");
607             system("color 2F");
608             cout << "============================================" << endl;
609             cout << "-Data Encryption Standard(Encryption/Decryption)-" << endl;
610             cout << "============================================" << endl;
611             cout << endl << endl;
612             cout << "      ************************************      " << endl;
613             cout << "      *             OPTIONS              *      " << endl;
614             cout << "      *                                  *      " << endl;
615             cout << "      *    1 Encryption from the Keyboard*      " << endl;
616             cout << "      *    2 Encryption from the File    *      " << endl;
617             cout << "      *    3 Decryption from the File    *      " << endl;
618             cout << "      *    4 Exit                        *      " << endl
619             cout << "      ************************************      " << endl;
620             cout << endl << "   What do you want to do: ";
621             cin >> strchoose;
622             if (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) != '4' || strchoose.length() != 1) {
623                 cout << "Error!Please enter again!" << endl;
624                 Sleep(500);
625             }
626         } while (strchoose.at(0) != '1'&&strchoose.at(0) != '2'&&strchoose.at(0) != '3'&&strchoose.at(0) != '4' || strchoose.length() != 1);
627         choose = strchoose[0];
628
629         switch (choose) {
630         case '1':
```

```
631                 fflush(stdin);
632                 system("cls");
633                 cout << "-------Data Encryption Standard(Encryption)-------" << endl;
634                 cout << endl << "Please enter the plaintext:" << endl;
635                 cout << "Plaintext: ";
636                 cin >> st;
637                 break;
638             case '2':
639                 fflush(stdin);
640                 system("cls");
641                 cout << "-------Data Encryption Standard(Encryption)-------" << endl;
642                 cout << endl << "Please enter the file name that you want to Encryption:" << endl;
643                 cout << "File name: ";
644                 st = readFileIntoString(fn);
645                 cout << endl << filename << ": " << endl;
646                 cout << st;
647                 cout << endl;
648                 break;
649             case '3':
650
651                 fflush(stdin);
652                 system("cls");
653                 cout << "-------Data Encryption Standard(Decryption)-------" << endl;
654                 cout << endl << "Please enter the file name that you want to Decryption:" << endl;
655                 cout << "File name: ";
656                 st = readFileIntoString(fn);
657                 cout << endl << filename << ": " << endl;
658                 cout << st;
659                 cout << endl;
660                 way = 1; //设置标记，标记为1为解密，标记为0为加密
661                 break;
662             default:
663                 cout << endl << "            ";
664                 return 0;
665             }
666
667             do {     //输入key
668                 fflush(stdin);
669                 cout << endl << "Please enter the key(7 English letters): " << endl;//通过输入字符，转化为ASCII作为密钥
670                 cout << "Key: ";
671                 cin >> key;
672                 if (key.length() != 7) {
673                     cout << "Error!The key must be 56bit(7 English letters)!" << endl << endl;
674                     exit(-1);
675                 }
676                 flag = Check(key);
677             } while (flag);
678
679             block = (st.length() - 1) / 8; //  明文分组，共block+1块
680             plainTrans(st, block, pltext, way);
```

```
681         temp = keyTrans(key, keytemp); //密钥二进制转换
682
683         makeKey(temp, subKey, way);         //得到子密钥
684
685         initial(pltext, iniPlain, st); //DES初始置换
686
687
688         for (i = 0; i < block + 1; i++) {//将初始化了的明文分离存入数组，得到LP和
689
690             for (j = 0; j < 64; j++) {
691                 if (j < 32) {
692                     leftPlain[i][j] = iniPlain[i][j];
693                 }
694                 else {
695                     rightPlain[i][j - 32] = iniPlain[i][j];
696                 }
697             }
698         }
699         for (i = 0; i < block + 1; i++) {  //处理的分块
700
701             for (flag = 0; flag < 32; flag++) {    //明文初始块的LP和RP
702
703                 templeft[flag] = leftPlain[i][flag];
704                 tempright[flag] = rightPlain[i][flag];
705             }
706             cout << endl << "-----Block " << i + 1 << " --------" << endl;
707             cout << endl << "Block" << i + 1 << ":";
708             for (m = 0; m < 64; m++) {
709                 cout << pltext[i][m];
710             }
711             cout << endl;
712
713             cout << "IT: ";
714             for (m = 0; m < 64; m++) {
715                 cout << iniPlain[i][m];
716             }
717
718             cout << endl << endl << "LP(0): ";
719             for (flag = 0; flag < 32; flag++) {
720                 cout << templeft[flag];
721             }
722             cout << endl;
723
724             cout << "RP(0): ";
725             for (flag = 0; flag < 32; flag++) {
726                 cout << tempright[flag];
727             }
728             cout << endl;
729
730             for (j = 0; j < 16; j++) {//每次加密需要循环16次
```

```
731                     cout << "EBox" << j << ": ";   //Ebox扩展置换
732                     funEBox(extrightPlain, tempright);
733                     cout << "SBox" << j << ": ";
734                     funSBox(subKey, extrightPlain, outSBox, j);
735                     cout << "PBox" << j << ": ";
736                     funPBox(outSBox, outPBox);
737                     for (m = 0; m < 32; m++) {
738                         temptrans[m] = tempright[m];
739                         tempright[m] = (templeft[m]) ^ (outPBox[m]);
740                         templeft[m] = temptrans[m];
741                     }
742
743                     if (j == 15) {
744                         for (m = 0; m < 32; m++) {
745                             temptrans[m] = tempright[m];
746                             tempright[m] = templeft[m];
747                             templeft[m] = temptrans[m];
748                         }
749                     }
750                     cout << endl << "LP(" << j + 1 << "): ";
751                     for (flag = 0; flag < 32; flag++) {
752                         cout << templeft[flag];
753                     }
754
755                     cout << endl << "RP(" << j + 1 << "): ";
756                     for (flag = 0; flag < 32; flag++) {
757                         cout << tempright[flag];
758                     }
759                     cout << endl;
760                 }
761                 funFinalTP(templeft, tempright, finalTrans, i, finalresult);
762             }
763             cout << endl << "------------Final Result --------------" << endl;
764             cout << endl << "Stream Ciphertext: " << endl << endl;
765             for (i = 0; i < block + 1; i++) {
766                 for (j = 0; j < 64; j++) {
767                     cout << finalresult[i][j];
768                 }
769                 cout << endl;
770             }
771
772             //由ASCII码输出字符
773             outputchar(finalresult, i, way);
774             system("PAUSE");
775         }
776
777 }
778
```

七、 思考题

1. DES 的原理是什么?

   答：DES 是分组加密算法， 明文按 64 位进行分组，有效密钥长度为 56 位。通过将分组后的明文组和 56 位的密钥按位替代或置换的方法形成密文组。

2. DES 使用多少位密钥?

   答：DES 使用 56 位密钥。

3. DES 对明文分块的单位是多少?

   答：DES 算法中明文以 64 位进行分组。

4. DES 对每一个数据块加密的轮次是多少?

   答：DES 对每一个数据块加密的轮次为 16 轮。

5. 简单描述 EBox 的操作过程。

   答：Ebox 是一种扩展运算，对 32 位的数据组的各位按照一定的顺序进行选择和排列，产生一个 48 位的结果。

6. 简单描述 SBox 的操作过程。

   答：Sbox 将 48 比特向量通过非线性映射变为 32 比特向量。首先，48 比特的向量被分为 8 个 6 比特分组，8 个分组在 8 个不同的非线性 Sbox 的作用下被转变为 8 个 4 比特分组，其中每个 Sbox 都将 6 比特输入映射为 4 比特输出。

7. 简单描述 PBox 的操作过程。

   答：Pbox 是一种置换运算，把 Sbox 输出的 32 位数据打乱重排，得到 32 位的加密函数输出。用 P 置换来提供扩散，把 S 盒的混淆作用扩散开来。

八、 结束语

通过实践课程，我深入了解了数据加密标准(DES)的加密原理及算法步骤，加深了我对分组密码的理解，在实践过程中不仅验证了理论知识，还加强了实验手段和实践技能，培养了自主分析问题、解决问题、应用知识的能力和创新精神，获益匪浅。我认为实践课程对深入理解密码学的内涵有很大帮助，理论与实践相结合，对我的综合素质有很大的提升。

九、 参考文献

1.Douglas R.Stinson:《Cryptography Theory and Practice》(Third Edition)，电子工业出版社，2008

2.张焕国，唐明著:《密码学引论》(第三版)，武汉大学出版社，2015

3.郑东 李祥学等著:《密码学——密码算法与协议》(第二版),电子工业出版社，2013

4.Michael Welschenbach:《Cryptography in C and C++》(Second Edition),机械工业出版社，2015

实验成绩

考查内容	分数	得分
做好实验内容的预习，写出预习报告	10	
了解实验题目的调试方法	10	
按实验要求预先设计好程序	10	
认真记录实验数据并分析实验结果	10	
实验后按要求书写实验报告，记录实验用数据及运行结果	30	
创新能力强，在实验中设计的程序有一定的通用性，算法优化	20	
实验过程中，具有严谨的学习态度，认真、踏实、一丝不苟的科学作风	10	