**本 科 实 验 报 告**

实验名称： 通信保密与信息安全课程设计

学 员： 曹务绅 学 号： 200604015014

培养类型： 技术类 年 级： 大三

专 业： 通信工程 所属学院： 四院

指导教员： 张权 职 称： 副教授

实 验 室： 实验日期： 2009-6-20

国防科学技术大学训练部制

**通信保密与信息安全课程设计**

曹务绅 200604015014 通信工程

## 任务需求和规格说明

DES加密算法是由 IBM 研究在1977年提出的。使用64位(bit)的数据进行加密和解密的，所用的密钥也是64位。并被美国国家标准局宣布为数据加密标准DES，主要用于非国家保密机关。



DES主要的加解密过程如右图所示：

本实验就是在了解掌握DES加解密原理的基础上，

利用掌握的软件对DES过程进行仿真，要求

1. 能对键盘输入的任何数据（字符串）进行加密
2. DES加密，
3. 显示加密结果。
4. 实现DES的解密

## 问题描述：

对于DES加密过程，可以由以下算式表示：

**L0R0🡨IP(<64bit输入码>)**

**Li🡨Ri-1 i=1,2,…,16**

**Ri🡨Li-1⊕f(Ri-1,ki) i=1,2,…,16**

**<64bit密文>🡨IP-1(R16L16)**

所以对于软件仿真，需要解决的问题主哟啊有3个：初始装换IP以及初始逆转换IP-1的实现，子密钥的生成，加密函数f的实现。

1、初始装换IP以及初始逆转换IP-1

DES算法把64位的明文输入块变為64位的密文输出块，它所使用的密钥也是64位,其功能是把输入的64位数据块按位重新组合，并把输出分為L0、R0两部分，每部分各长32位，其置换规则见下表：

58,50,12,34,26,18,10,2,60,52,44,36,28,20,12,4,  
　　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,

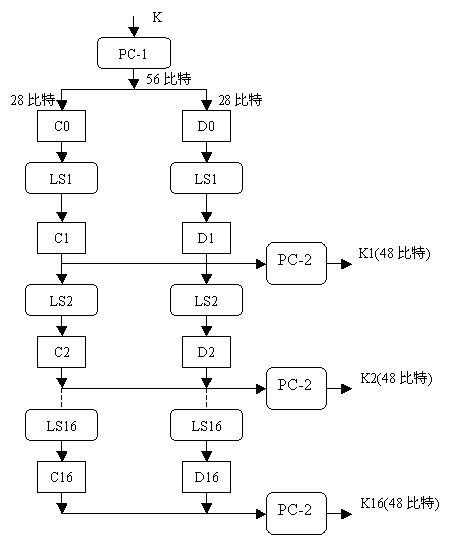
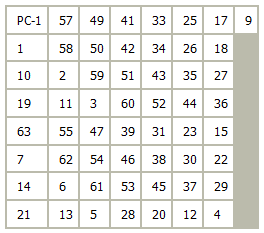
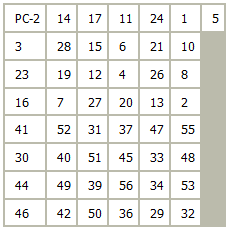
　　即将输入的第58位换到第一位，第50位换到第2位，...，依此类推，最后一位是原来的第7位。L0、R0则是换位输出后的两部分，L0是输出的左32位，R0 是右32位，例：设置换前的输入值為D1D2D3......D64，则经过初始置换后的结果為：L0=D550...D8；R0=D57D49...D7。

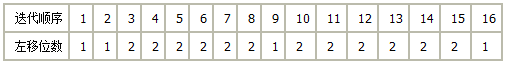
　　经过26次迭代运算后。得到L16、R16，将此作為输入，进行逆置换，即得到密文输出。逆置换正好是初始置的逆运算，例如，第1位经过初始置换后，处於第40位，而通过逆置换，又将第40位换回到第1位，其逆置换规则如下表所示：

　　40,8,48,16,56,24,64,32,39,7,47,15,55,23,63,31,  
　　38,6,46,14,54,22,62,30,37,5,45,13,53,21,61,29,  
　　36,4,44,12,52,20,60,28,35,3,43,11,51,19,59,27,  
　　34,2,42,10,50,18,58 26,33,1,41, 9,49,17,57,25,

2、子密钥的生成

子密钥生成的过程主要包括置换选择，循环转移，再置换三个过程。

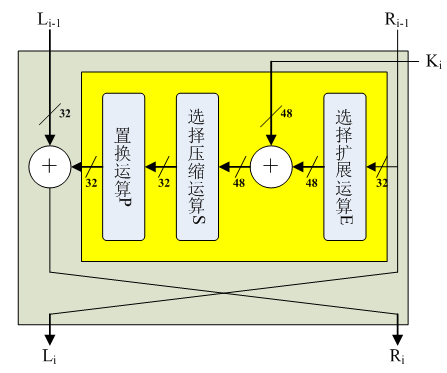


3、加密函数f

加密函数是DES算法的核心，是将32位比特串转换为另一个32位比特串的运算。  
其中包含多个操作：

将32位的输入扩展为48位；

与轮密钥按位异或；

把得到的48位分为八组，每组六位，分别通过8个S盒S1，S2，…，S8输出，每组只输出四位，组合成32位；

将32位最后通过置换后输出；

其中：

S-盒是DES算法的关键所在，DES中其它算

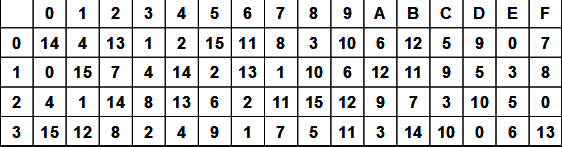
法都是线性的，S-盒运算是非线性的S-盒不

易于分析，它提供了很好的安全性8个S盒

S1,S2,…,S8, Si:{0,1}6→{0,1}46bits映射为

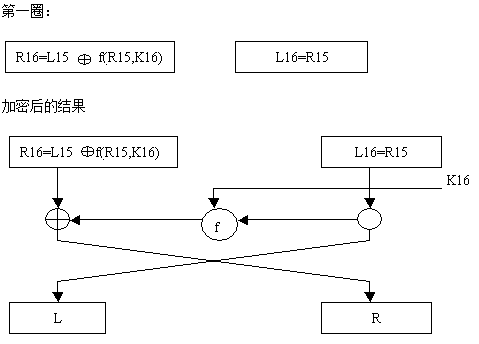
4bits,用4\*16的矩阵来描述，元素为0,1 ,

…,15。



4、DES解密

DES的解密过程和DES的加密过程完全类似，只不过将16圈的子密钥序列K1，K2……K16的顺序倒过来。即第一圈用第16个子密钥K16，第二圈用K15，其余类推。



## 方案设计

### 3.1 设计思路：主要算法思想

根据以上的分析，DES过程的仿真中可以大致分为4个模块来进行，分别对应上述的初始装换IP以及初始逆转换IP-1的实现，子密钥的生成，加密函数f，解密模块。由于解密模块可以看做DES加密过程的逆运算，故在编程的时候，宜将各个移位、置换、S盒运算等等都以子函数的形式编程。此外，由于是对字符串做处理，所以至始至终都会存在字符串分组，移位，异或，比较，字节转换等操作，为了提高程序效率，也将这些函数以子函数的形式编辑在程序中，方便调用。

// 通用置换函数

void Transform(bool \*Out, bool \*In, const char \*Table, int len)

{

static bool Tmp[256];

for(int i=0; i<len; i++)

Tmp[i] = In[ Table[i]-1 ];

memcpy(Out, Tmp, len);

}

// 异或运算

void Xor(bool \*InA, const bool \*InB, int len)

{

for(int i=0; i<len; i++)

InA[i] ^= InB[i];

}

// 循环左移

void RotateL(bool \*In, int len, int loop)

{

static bool Tmp[256];

memcpy(Tmp, In, loop);//loop 为位移的个数

memcpy(In, In+loop, len-loop);

memcpy(In+len-loop, Tmp, loop);

}

// 字节转换成位

void ByteToBit(bool \*Out, const char \*In, int bits)

{

for(int i=0; i<bits; i++)

Out[i] = (In[i/8]>>(i%8)) & 1;

}

// 位转换成字节

void BitToByte(char \*Out, const bool \*In, int bits)

{

memset(Out, 0, (bits+7)/8);

for(int i=0; i<bits; i++)

Out[i/8] |= In[i]<<(i%8);

}

### 3.2 设计表示：

相关函数及注释如下所示：

void Des\_Run(char Out[8], char In[8],int i)

{

static bool M[64], Tmp[32], \*Li = &M[0], \*Ri = &M[32];

ByteToBit(M, In, 64);// 字节组转换成位组

Transform(M, M, IP, 64);// 初始变换

if( i == 0 ){ //加密

for(int i=0; i<16; i++) {

memcpy(Tmp, Ri, 32);

F\_func(Ri, SubKey[i]);

Xor(Ri, Li, 32);

memcpy(Li, Tmp, 32);

}

}else{ //解密

for(int i=15; i>=0; i--) {

memcpy(Tmp, Li, 32);

F\_func(Li, SubKey[i]);

Xor(Li, Ri, 32);

memcpy(Ri, Tmp, 32);

}

}

Transform(M, M, IPR, 64);

BitToByte(Out, M, 64);

}

void Des\_SetKey(const char Key[8]) //设置密钥

{

static bool K[64], \*KL = &K[0], \*KR = &K[28];

ByteToBit(K, Key, 64);// 字节组转换成位组

Transform(K, K, PC1, 56);// 密钥变换 PC-1

for(int i=0; i<16; i++) {

RotateL(KL, 28, LOOP[i]);

RotateL(KR, 28, LOOP[i]);

Transform(SubKey[i], K, PC2, 48);// 密钥压缩变换 PC-2

}

}

// F 函数

void F\_func(bool In[32], const bool Ki[48])

{

static bool MR[48];

Transform(MR, In, E, 48);

Xor(MR, Ki, 48);

S\_func(In, MR);

Transform(In, In, P, 32);

}

// S 盒置换

void S\_func(bool Out[32], const bool In[48])

{

for(char i=0,j,k; i<8; i++,In+=6,Out+=4) {

j = (In[0]<<1) + In[5];

k = (In[1]<<3) + (In[2]<<2) + (In[3]<<1) + In[4];

ByteToBit(Out, &S\_Box[i][j][k], 4);

}

}

### 3.3 详细设计：主要算法的伪代码

主函数：（各子函数请参照3.1和3.2）

int main()

{

char key[8]={1,0,5,7,9,4,7},message[8];

string mm,m1;

cout<<"Please input The string before encrypting:\n";

cin>>mm;

int len = mm.length(),i;

if (len % 8!=0)

{

for (i = 0; i < 8 - len % 8; i++) mm+=' ';

len = mm.length();

}

puts("Before encrypting");

puts(mm.c\_str());

Des\_SetKey(key);

m1 = mm;

for (i = 0; i < len; i ++)

{

if (i % 8 == 0 && i > 0)

{

Des\_Run(message,message,0);

for (int j = 0; j < 8; j ++)

m1[i - 8 + j] = message[j];

}

message[i % 8] = mm[i];

}

Des\_Run(message,message,0);

for (int j = 0; j < 8; j ++)

m1[i - 8 + j] = message[j];

puts("After encrypting");

puts(m1.c\_str());

puts("After decrypting");

for (i = 0; i < len; i ++)

{

if (i % 8 == 0 && i > 0)

{

Des\_Run(message,message,1);

for (int j = 0; j < 8; j ++)

mm[i - 8 + j] = message[j];

}

message[i % 8] = m1[i];

}

Des\_Run(message,message,1);

for (int j = 0; j < 8; j ++)

mm[i - 8 + j] = message[j];

puts(mm.c\_str());

system("Pause");

}

## 测试结果分析

### 4.1 调试过程中遇到的主要问题，是如何解决的

在试验中，主函数是比较好编辑的，按照DES加解密的每一个环节，调用需要的子函数即可，但是在写S盒运算和循环移位函数的时候遇到比较大的困难，在这方面确实查阅了不少资料，对如何用C语言完成这两种运算有了比较深的了解。

此外，原来写的程序只能对8的倍数比特的信息进行加密，考虑到这种情况的局限性，在源程序的基础上添加了补空格的操作，即对输入不是8比特的倍数的信息，首先对信息进行补空格，然后在进行分组加密。

### 4.2 改进设想

由于现在所做的程序只是实验和仿真，距离真正的应用还有很大的差距。在查阅资料的时候已经发现现在网上已经出现好多集成化很高的DES加密软件，这种软件不仅可以对输入的信息进行加密，还可以读取文本文档、甚至是歌曲、视频文件，对此进行加密，功能拓展了好多。因此若是对本程序加以改进，一个可以读取TXT文档并且对其进行加解密的功能是很必要的。

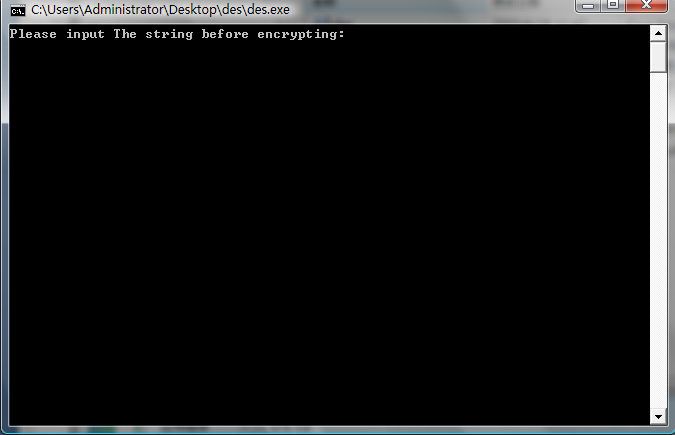
除此之外，还可以给程序附上一个方便与用户交互的操作界面，

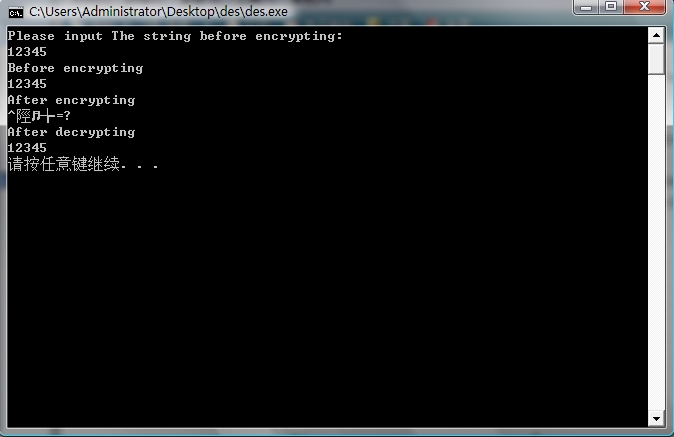
### 4.3 经验和体会

1、实验的目的都是为了对所学的知识进一步深化和掌握，在实验中，我们可以清楚的认识到自己所学的东西在现实生活中到底是怎么应用的，就像本次实验，若不是自己亲自动手进行仿真，也许现在我对DES的了解还是只停留在课件上，书本上，这样的知识是不牢固的，也是没有用处的。

2、在这次实验中，由于对字符的操作上C++有很大的便利性，所以时隔1年多，我又重新捡起C++这种工具，现在专业课上用的最多的是MATLAB，对C++的利用是少之又少，所以在这次仿真中确实遇到不少麻烦，好多基本的C++的语法都不是很熟悉了，给编程带来很大的困难，托的时间也比较久，导致做出来的程序比较粗糙，功能很简单，更没有达到预想中的做出界面的效果。

## 附录：





所有代码：

#include <cstdio>

#include <iostream>

#include <memory>

#include <string>

using namespace std;

static void F\_func(bool In[32], const bool Ki[48]);// F 函数

static void S\_func(bool Out[32], const bool In[48]);// S 盒代替

static void Transform(bool \*Out, bool \*In, const char \*Table, int len);

static void Xor(bool \*InA, const bool \*InB, int len);// 异或

static void RotateL(bool \*In, int len, int loop);// 循环左移

static void ByteToBit(bool \*Out, const char \*In, int bits);

static void BitToByte(char \*Out, const bool \*In, int bits);

// 初始变换

const static char IP[64] = {

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

};

// 末置换

const static char IPR[64] = {

40, 8, 48, 16, 56, 24, 64, 32, 39, 7, 47, 15, 55, 23, 63, 31,

38, 6, 46, 14, 54, 22, 62, 30, 37, 5, 45, 13, 53, 21, 61, 29,

36, 4, 44, 12, 52, 20, 60, 28, 35, 3, 43, 11, 51, 19, 59, 27,

34, 2, 42, 10, 50, 18, 58, 26, 33, 1, 41, 9, 49, 17, 57, 25

};

// 扩展置换

static const char E[48] = {

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

};

// 作用在S盒的32位输出结果上的P变换

const static char P[32] = {

16, 7, 20, 21, 29, 12, 28, 17, 1, 15, 23, 26, 5, 18, 31, 10,

2, 8, 24, 14, 32, 27, 3, 9, 19, 13, 30, 6, 22, 11, 4, 25

};

// 密钥初始置换

const static char PC1[56] = {

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

};

// 密钥压缩变换 PC-2

const static char PC2[48] = {

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

};

// 每圈移位次数

const static char LOOP[16] = {

1,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1

};

// S盒

const static char S\_Box[8][4][16] = {

// S1

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,

// S2

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,

// S3

10, 0, 9, 14, 6, 3, 15, 5, 1, 13, 12, 7, 11, 4, 2, 8,

13, 7, 0, 9, 3, 4, 6, 10, 2, 8, 5, 14, 12, 11, 15, 1,

13, 6, 4, 9, 8, 15, 3, 0, 11, 1, 2, 12, 5, 10, 14, 7,

1, 10, 13, 0, 6, 9, 8, 7, 4, 15, 14, 3, 11, 5, 2, 12,

// S4

7, 13, 14, 3, 0, 6, 9, 10, 1, 2, 8, 5, 11, 12, 4, 15,

13, 8, 11, 5, 6, 15, 0, 3, 4, 7, 2, 12, 1, 10, 14, 9,

10, 6, 9, 0, 12, 11, 7, 13, 15, 1, 3, 14, 5, 2, 8, 4,

3, 15, 0, 6, 10, 1, 13, 8, 9, 4, 5, 11, 12, 7, 2, 14,

// S5

2, 12, 4, 1, 7, 10, 11, 6, 8, 5, 3, 15, 13, 0, 14, 9,

14, 11, 2, 12, 4, 7, 13, 1, 5, 0, 15, 10, 3, 9, 8, 6,

4, 2, 1, 11, 10, 13, 7, 8, 15, 9, 12, 5, 6, 3, 0, 14,

11, 8, 12, 7, 1, 14, 2, 13, 6, 15, 0, 9, 10, 4, 5, 3,

// S6

12, 1, 10, 15, 9, 2, 6, 8, 0, 13, 3, 4, 14, 7, 5, 11,

10, 15, 4, 2, 7, 12, 9, 5, 6, 1, 13, 14, 0, 11, 3, 8,

9, 14, 15, 5, 2, 8, 12, 3, 7, 0, 4, 10, 1, 13, 11, 6,

4, 3, 2, 12, 9, 5, 15, 10, 11, 14, 1, 7, 6, 0, 8, 13,

// S7

4, 11, 2, 14, 15, 0, 8, 13, 3, 12, 9, 7, 5, 10, 6, 1,

13, 0, 11, 7, 4, 9, 1, 10, 14, 3, 5, 12, 2, 15, 8, 6,

1, 4, 11, 13, 12, 3, 7, 14, 10, 15, 6, 8, 0, 5, 9, 2,

6, 11, 13, 8, 1, 4, 10, 7, 9, 5, 0, 15, 14, 2, 3, 12,

// S8

13, 2, 8, 4, 6, 15, 11, 1, 10, 9, 3, 14, 5, 0, 12, 7,

1, 15, 13, 8, 10, 3, 7, 4, 12, 5, 6, 11, 0, 14, 9, 2,

7, 11, 4, 1, 9, 12, 14, 2, 0, 6, 10, 13, 15, 3, 5, 8,

2, 1, 14, 7, 4, 10, 8, 13, 15, 12, 9, 0, 3, 5, 6, 11

};

static bool SubKey[16][48];// 16圈子密钥

void Des\_Run(char Out[8], char In[8],int i)

{

static bool M[64], Tmp[32], \*Li = &M[0], \*Ri = &M[32];

ByteToBit(M, In, 64);// 字节组转换成位组

Transform(M, M, IP, 64);// 初始变换

if( i == 0 ){ //加密

for(int i=0; i<16; i++) {

memcpy(Tmp, Ri, 32);

F\_func(Ri, SubKey[i]);

Xor(Ri, Li, 32);

memcpy(Li, Tmp, 32);

}

}else{ //解密

for(int i=15; i>=0; i--) {

memcpy(Tmp, Li, 32);

F\_func(Li, SubKey[i]);

Xor(Li, Ri, 32);

memcpy(Ri, Tmp, 32);

}

}

Transform(M, M, IPR, 64);

BitToByte(Out, M, 64);

}

void Des\_SetKey(const char Key[8]) //设置密钥

{

static bool K[64], \*KL = &K[0], \*KR = &K[28];

ByteToBit(K, Key, 64);// 字节组转换成位组

Transform(K, K, PC1, 56);// 密钥变换 PC-1

for(int i=0; i<16; i++) {

RotateL(KL, 28, LOOP[i]);

RotateL(KR, 28, LOOP[i]);

Transform(SubKey[i], K, PC2, 48);// 密钥压缩变换 PC-2

}

}

// F 函数

void F\_func(bool In[32], const bool Ki[48])

{

static bool MR[48];

Transform(MR, In, E, 48);

Xor(MR, Ki, 48);

S\_func(In, MR);

Transform(In, In, P, 32);

}

// S 盒置换

void S\_func(bool Out[32], const bool In[48])

{

for(char i=0,j,k; i<8; i++,In+=6,Out+=4) {

j = (In[0]<<1) + In[5];

k = (In[1]<<3) + (In[2]<<2) + (In[3]<<1) + In[4];

ByteToBit(Out, &S\_Box[i][j][k], 4);

}

}

// 通用置换函数

void Transform(bool \*Out, bool \*In, const char \*Table, int len)

{

static bool Tmp[256];

for(int i=0; i<len; i++)

Tmp[i] = In[ Table[i]-1 ];

memcpy(Out, Tmp, len);

}

// 异或运算

void Xor(bool \*InA, const bool \*InB, int len)

{

for(int i=0; i<len; i++)

InA[i] ^= InB[i];

}

// 循环左移

void RotateL(bool \*In, int len, int loop)

{

static bool Tmp[256];

memcpy(Tmp, In, loop);//loop 为位移的个数

memcpy(In, In+loop, len-loop);

memcpy(In+len-loop, Tmp, loop);

}

// 字节转换成位

void ByteToBit(bool \*Out, const char \*In, int bits)

{

for(int i=0; i<bits; i++)

Out[i] = (In[i/8]>>(i%8)) & 1;

}

// 位转换成字节

void BitToByte(char \*Out, const bool \*In, int bits)

{

memset(Out, 0, (bits+7)/8);

for(int i=0; i<bits; i++)

Out[i/8] |= In[i]<<(i%8);

}

int main()

{

char key[8]={1,0,5,7,9,4,7},message[8];

string mm,m1;

cout<<"Please input The string before encrypting:\n";

cin>>mm;

int len = mm.length(),i;

if (len % 8!=0)

{

for (i = 0; i < 8 - len % 8; i++) mm+=' ';

len = mm.length();

}

puts("Before encrypting");

puts(mm.c\_str());

Des\_SetKey(key);

m1 = mm;

for (i = 0; i < len; i ++)

{

if (i % 8 == 0 && i > 0)

{

Des\_Run(message,message,0);

for (int j = 0; j < 8; j ++)

m1[i - 8 + j] = message[j];

}

message[i % 8] = mm[i];

}

Des\_Run(message,message,0);

for (int j = 0; j < 8; j ++)

m1[i - 8 + j] = message[j];

puts("After encrypting");

puts(m1.c\_str());

puts("After decrypting");

for (i = 0; i < len; i ++)

{

if (i % 8 == 0 && i > 0)

{

Des\_Run(message,message,1);

for (int j = 0; j < 8; j ++)

mm[i - 8 + j] = message[j];

}

message[i % 8] = m1[i];

}

Des\_Run(message,message,1);

for (int j = 0; j < 8; j ++)

mm[i - 8 + j] = message[j];

puts(mm.c\_str());

system("Pause");

}