**实现数据加密标准DES密码算法**

1. **DES**

DES全称为Data Encryption Standard，即数据加密标准，是一种使用[密钥加密](https://baike.baidu.com/item/%E5%AF%86%E9%92%A5%E5%8A%A0%E5%AF%86)的块算法，1977年被[美国联邦政府](https://baike.baidu.com/item/%E7%BE%8E%E5%9B%BD%E8%81%94%E9%82%A6%E6%94%BF%E5%BA%9C)的国家标准局确定为[联邦资料处理标准](https://baike.baidu.com/item/%E8%81%94%E9%82%A6%E8%B5%84%E6%96%99%E5%A4%84%E7%90%86%E6%A0%87%E5%87%86)（FIPS），并授权在非密级政府通信中使用，随后该算法在国际上广泛流传开来。需要注意的是，在某些文献中，作为算法的DES称为数据加密算法（Data Encryption Algorithm,DEA），已与作为标准的DES区分开来。

1. **DES的基本原则**

DES设计中使用了分组密码设计的两个原则：混淆（confusion）和扩散(diffusion)，其目的是抗击敌手对密码系统的统计分析。混淆是使密文的统计特性与密钥的取值之间的关系尽可能复杂化，以使密钥和明文以及密文之间的依赖性对密码分析者来说是无法利用的。扩散的作用就是将每一位明文的影响尽可能迅速地作用到较多的输出密文位中，以便在大量的密文中消除明文的统计结构，并且使每一位密钥的影响尽可能迅速地扩展到较多的密文位中，以防对密钥进行逐段破译。

1. **算法步骤**

**初始置换**

其功能是把输入的64位数据块按位重新组合，并把输出分为L0、R0两部分，每部分各长32位，其置换规则为将输入的第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:设置换前的输入值为D1D2D3……D64,则经过初始置换后的结果为:L0=D58D50……D8;R0=D57D49……D7

**逆置换**

经过16次迭代运算后,得到L16、R16,将此作为输入，进行逆置换，逆置换正好是初始置换的逆运算，由此即得到密文输出。

1. **模拟实现**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

char str[64];

void Encryption();

//\*\*\*\*\*\*\*明文初始置换\*\*\*\*\*\*\*//

int 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};

//\*\*\*\*\*\*\*将明文右半部分扩充为48位\*\*\*\*\*\*\*//

int 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};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* S盒 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

int s1[4][16]={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};

int s2[4][16]={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};

int s3[4][16]={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};

int s4[4][16]={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};

int s5[4][16]={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};

int s6[4][16]={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};

int s7[4][16]={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};

int s8[4][16]={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};

//\*\*\*\*\*\*\*\*\*\*\*\*\* 对进行F函数后的值进行置换 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

int 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1 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

int PC\_1[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};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 密钥左右两部分移位后进行的置换2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

int PC\_2[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};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 逆初始置换 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

int ip\_[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};

/\*des.cpp\*/

#include "des.h"

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 解密过程函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

void Dnc(int mw[],int my[])

{

FILE \*JieMi;

char c[9]="\0"; //存储二进制字符

char MWen[64];

int i=0,j=0,n,m;

int k;

int MW[64];

int L[2][32],R[2][32]; //密文的左右两部分

int SL[2][28],SR[2][28]; //密钥的左右两部分

int K1[56]; //密钥左右两部分循环后的结果

int K[48]; //密钥左右两部分循环后的结果进行48位的置换

int RE[48]; //密文的右部分扩充为48位

int ss[8][6]; //将48位密钥和48位的右半部分名文异或后的结果分成8组，每组有6个数

int s[8]; //用来存储经过8个S盒之后的值

int M[64]; //存储16轮迭代后的输出结果

int M1[64]; //存储16轮迭代后置换后的输出结果

//\*\*\*\*\*\*\*\*\*\*\*\* 密文的初始置换 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(k=0;k<64;k++)

MW[k]=mw[ip[k]-1];

//\*\*\*\*\*\*\*置换后的密文分为左右两部分\*\*\*\*\*\*\*//

for(k=0;k<32;k++)

L[0][k]=MW[k];

for(k;k<64;k++)

R[0][k-32]=MW[k];

//\*\*\*\*\*\*\*将解密密钥分为左右两部分\*\*\*\*\*\*\*//

for(k=0;k<28;k++)

SL[0][k]=my[k];

for(k;k<56;k++)

SR[0][k-28]=my[k];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*进行16轮密钥逆迭代过程\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(i=16;i>=1;i--)

{

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 因为我们所得的子密钥都是都是该轮移位好的密钥 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\* 比如16轮，它的子密钥是第16轮移位后所得到的结果，即已经是K16 \*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\* 比如15轮，它的子密钥是第16轮的K16进行逆移位所得到的结果，即为K15 \*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*\*\*\*\*\*\*\*\*所以我们要先置换存储和进行解密，完成后再来进行移位 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<28;j++) //先存储移位后的结果

K1[j]=SL[0][j];

for(j;j<56;j++)

K1[j]=SR[0][j-28];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 对移位后的密钥进行置换2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<48;j++) //进行置换

K[j]=K1[PC\_2[j]-1];

if(i==1||i==2||i==9||i==16)//密钥左右两部分循环右移一位

{

for(j=1;j<28;j++)

{

SL[1][j]=SL[0][j-1];

SR[1][j]=SR[0][j-1];

}

SL[1][0]=SL[0][27];

SR[1][0]=SR[0][27];

}

else //密钥左右两部分循环右移两位

{

for(j=2;j<28;j++)

{

SL[1][j]=SL[0][j-2];

SR[1][j]=SR[0][j-2];

}

SL[1][0]=SL[0][26];

SR[1][0]=SR[0][26];

SL[1][1]=SL[0][27];

SR[1][1]=SR[0][27];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 把密文的右半部分扩充为48位 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<48;j++)

RE[j]=R[0][E[j]-1];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将扩充为48位后的密文右半部分与进行置换2后的密钥进行异或 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<48;j++)

RE[j]=(RE[j]+K[j])%2;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的结果分成8组，每组6位。为后面进行S盒操作做准备 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<6;j++)

{

ss[0][j]=RE[j];

ss[1][j]=RE[6+j];

ss[2][j]=RE[12+j];

ss[3][j]=RE[18+j];

ss[4][j]=RE[24+j];

ss[5][j]=RE[30+j];

ss[6][j]=RE[36+j];

ss[7][j]=RE[42+j];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的结果进行S盒操 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

s[0]=s1[ss[0][5]+ss[0][0]\*2][ss[0][4]+ss[0][3]\*2+ss[0][2]\*4+ss[0][1]\*8];

s[1]=s2[ss[1][5]+ss[1][0]\*2][ss[1][4]+ss[1][3]\*2+ss[1][2]\*4+ss[1][1]\*8];

s[2]=s3[ss[2][5]+ss[2][0]\*2][ss[2][4]+ss[2][3]\*2+ss[2][2]\*4+ss[2][1]\*8];

s[3]=s4[ss[3][5]+ss[3][0]\*2][ss[3][4]+ss[3][3]\*2+ss[3][2]\*4+ss[3][1]\*8];

s[4]=s5[ss[4][5]+ss[4][0]\*2][ss[4][4]+ss[4][3]\*2+ss[4][2]\*4+ss[4][1]\*8];

s[5]=s6[ss[5][5]+ss[5][0]\*2][ss[5][4]+ss[5][3]\*2+ss[5][2]\*4+ss[5][1]\*8];

s[6]=s7[ss[6][5]+ss[6][0]\*2][ss[6][4]+ss[6][3]\*2+ss[6][2]\*4+ss[6][1]\*8];

s[7]=s8[ss[7][5]+ss[7][0]\*2][ss[7][4]+ss[7][3]\*2+ss[7][2]\*4+ss[7][1]\*8];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 把上面的值变为二进制 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(n=0;n<8;n++)

{

m=0;

while(m<4)

{

M[m]=s[n]%2;

s[n]=s[n]/2;

m++;

}

for(m=0;m<4;m++)

RE[n\*4+m]=M[3-m];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面变为二进制的F函数的值进行置换 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<32;j++)

K[j]=RE[P[j]-1];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 求变换后的密文结果(左右交换) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<32;j++)

{

R[1][j]=(L[0][j]+K[j])%2;

L[1][j]=R[0][j];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的密文结果进行交换,作为下一轮密文的输入和该轮的输出 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<32;j++)

{

L[0][j]=L[1][j];

R[0][j]=R[1][j];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的密钥结果进行交换,作为下一轮密钥的输入\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<28;j++)

{

SL[0][j]=SL[1][j];

SR[0][j]=SR[1][j];

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 16轮迭代结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*将第16迭代后的输出进行互换存储，产生预输出\*\*\*\*//

for(j=0;j<32;j++)

M[j]=R[0][j];

for(j;j<64;j++)

M[j]=L[0][j-32];

for(j=0;j<64;j++)

M1[j]=M[ip\_[j]-1];

k=7;

i=0;

for(j=0;j<64;j++)

{

c[k]=M1[j]+'0';

if(k==0)

{

if(strtol(c,NULL,2)!=0)

MWen[i++]=(char)strtol(c,NULL,2);

k=7;

memset(c,0,sizeof(c));

}

else k--;

}

JieMi=fopen("C:\\JieMiMW.txt","a");

j=0;

while(j<i)

{

fprintf(JieMi,"%c",MWen[j]);

j++;

}

fclose(JieMi);

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 解密主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

void Decryption()

{

FILE \*miwen;

FILE \*miyao;

int i=0,j=1,k;

int my[64]; //存储密钥

int mw[64]; //存储64位的密文

int \*mwen; //存储读入的密文

int c;

printf("\n//\*\*\*\*\*\*\* 正在解密......................\*\*\*\*\*\*//\n");

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 读入解密密钥 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

i=0;

miyao=fopen("C:\\miyao1.txt","r");

while(i<56)

fscanf(miyao,"%d",&my[i++]);

fclose(miyao);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 读入密文 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

miwen=fopen("C:\\miwen.txt","r");

i=0;

while(!feof(miwen))

{

fscanf(miwen,"%d",&c);

i++;

}

fclose(miwen);

mwen=(int \*)malloc(sizeof(int)\*i);

miwen=fopen("C:\\miwen.txt","r");

i=0;

while(!feof(miwen))

{

fscanf(miwen,"%d",&mwen[i++]);

}

fclose(miwen);

for(j=1;j<=(i/64);j++)

{

for(k=0;k<64;k++)

mw[k]=mwen[k+64\*(j-1)];

Dnc(mw,my);

}

printf("\n\n\n//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 解密完成 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//\n");

printf("\n//\*\*\*\*\*\* 解密后的明文，存储在C:\\JieMiMW.txt中 \*\*\*\*\*\*//\n");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 加密过程函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

void Enc(int mw[],int my[])

{

FILE \*miwen;

FILE \*miyao;

int i=0,j=0,n,m;

int k;

int MW[64],MY[64];

int L[2][32],R[2][32]; //明文的左右两部分

int SL[2][28],SR[2][28]; //密钥的左右两部分

int K1[56]; //密钥左右两部分循环后的结果

int K[48]; //密钥左右两部分循环后的结果进行48位的置换

int RE[48]; //明文的右部分扩充为48位

int ss[8][6]; //将48位密钥和48位的右半部分名文异或后的结果分成8组，每组有6个数

int s[8]; //用来存储经过8个S盒之后的值

int M[64]; //存储16轮迭代后的输出结果

//\*\*\*\*\*\*\*\*\*\*\*\* 明文的初始置换 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(k=0;k<64;k++)

MW[k]=mw[ip[k]-1];

//\*\*\*\*\*\*\*置换后的明文分为左右两部分\*\*\*\*\*\*\*//

for(k=0;k<32;k++)

L[0][k]=MW[k];

for(k;k<64;k++)

R[0][k-32]=MW[k];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 密钥的初始置换 \*\*\*\*\*\*\*\*\*\*\*\*//

for(k=0;k<56;k++)

MY[k]=my[PC\_1[k]-1];

//\*\*\*\*\*\*\*将置换后的密钥分为左右两部分\*\*\*\*\*\*\*//

for(k=0;k<28;k++)

SL[0][k]=MY[k];

for(k;k<56;k++)

SR[0][k-28]=MY[k];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*进行16轮迭代\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(i=1;i<=16;i++)

{

if(i==1||i==2||i==9||i==16)//密钥左右两部分循环左移一位

{

for(j=0;j<27;j++)

{

SL[1][j]=SL[0][j+1];

SR[1][j]=SR[0][j+1];

}

SL[1][27]=SL[0][0];

SR[1][27]=SR[0][0];

}

else //密钥左右两部分循环左移两位

{

for(j=0;j<26;j++)

{

SL[1][j]=SL[0][j+2];

SR[1][j]=SR[0][j+2];

}

SL[1][26]=SL[0][0];

SR[1][26]=SR[0][0];

SL[1][27]=SL[0][1];

SR[1][27]=SR[0][1];

}

//\*\*\*\*\*\*\*\*{\*\*\*\*\* 对移位后的密钥进行置换2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<28;j++) //先存储移位后的结果

K1[j]=SL[1][j];

for(j;j<56;j++)

K1[j]=SR[1][j-28];

for(j=0;j<48;j++) //进行置换

K[j]=K1[PC\_2[j]-1];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 把明文的右半部分扩充为48位 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<48;j++)

RE[j]=R[0][E[j]-1];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将扩充为48位后的明文右半部分与进行置换2后的密钥进行异或 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<48;j++)

RE[j]=(RE[j]+K[j])%2;

//\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的结果分成8组，每组6位。为后面进行S盒操作做准备 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<6;j++)

{

ss[0][j]=RE[j];

ss[1][j]=RE[6+j];

ss[2][j]=RE[12+j];

ss[3][j]=RE[18+j];

ss[4][j]=RE[24+j];

ss[5][j]=RE[30+j];

ss[6][j]=RE[36+j];

ss[7][j]=RE[42+j];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的结果进行S盒操 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

s[0]=s1[ss[0][5]+ss[0][0]\*2][ss[0][4]+ss[0][3]\*2+ss[0][2]\*4+ss[0][1]\*8];

s[1]=s2[ss[1][5]+ss[1][0]\*2][ss[1][4]+ss[1][3]\*2+ss[1][2]\*4+ss[1][1]\*8];

s[2]=s3[ss[2][5]+ss[2][0]\*2][ss[2][4]+ss[2][3]\*2+ss[2][2]\*4+ss[2][1]\*8];

s[3]=s4[ss[3][5]+ss[3][0]\*2][ss[3][4]+ss[3][3]\*2+ss[3][2]\*4+ss[3][1]\*8];

s[4]=s5[ss[4][5]+ss[4][0]\*2][ss[4][4]+ss[4][3]\*2+ss[4][2]\*4+ss[4][1]\*8];

s[5]=s6[ss[5][5]+ss[5][0]\*2][ss[5][4]+ss[5][3]\*2+ss[5][2]\*4+ss[5][1]\*8];

s[6]=s7[ss[6][5]+ss[6][0]\*2][ss[6][4]+ss[6][3]\*2+ss[6][2]\*4+ss[6][1]\*8];

s[7]=s8[ss[7][5]+ss[7][0]\*2][ss[7][4]+ss[7][3]\*2+ss[7][2]\*4+ss[7][1]\*8];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 把上面的值变为二进制 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(n=0;n<8;n++)

{

m=0;

while(m<4)

{

M[m]=s[n]%2;

s[n]=s[n]/2;

m++;

}

for(m=0;m<4;m++)

RE[n\*4+m]=M[3-m];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面变为二进制的F函数的值进行置换 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<32;j++)

K[j]=RE[P[j]-1];

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 求变换后的明文结果(左右交换) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<32;j++)

{

R[1][j]=(L[0][j]+K[j])%2;

L[1][j]=R[0][j];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的明文结果进行交换,作为下一轮明文的输入 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<32;j++)

{

L[0][j]=L[1][j];

R[0][j]=R[1][j];

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 将上面所得的密钥结果进行交换,作为下一轮密钥的输入\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

for(j=0;j<28;j++)

{

SL[0][j]=SL[1][j];

SR[0][j]=SR[1][j];

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 16轮迭代结束 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

//\*\*\*\*将迭代后的结果进行互换，产生预输出\*\*\*\*//

for(j=0;j<32;j++)

M[j]=R[0][j];

for(j;j<64;j++)

M[j]=L[0][j-32];

miwen=fopen("C:\\miwen.txt","a");

for(j=0;j<64;j++)

{

fprintf(miwen," %d",M[ip\_[j]-1]);

}//逆初始置换

fclose(miwen);

miyao=fopen("C:\\miyao1.txt","w");

for(j=0;j<56;j++)

{

if(j<28)

{

fprintf(miyao," %d",SL[1][j]);

}

else

{

fprintf(miyao," %d",SR[1][j-28]);

}

}//逆初始置换

fclose(miwen);

printf("\n//\*\*\*\*\*\* 加密完成 \*\*\*\*\*\*//\n");

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 加密主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

void Encryption()

{

FILE \*mingwen;

FILE \*miyao;

char c;

char \*S;//存储明文字符

int i=0,j=0,m;

int k;

int n; //用来标记字符化为二进制后是否是8位

int length=0;

int my[64]; //存储密钥

int mw[64]; //存储64位的明文

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 统计明文字符的个数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

mingwen=fopen("C:\\mingwen.txt","r");

while(!feof(mingwen))

{

fscanf(mingwen,"%c",&c);

length++;

}

fclose(mingwen);

length--;//明文的个数

S=(char \*)malloc(sizeof(char)\*(length+4));

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 读入明文 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

mingwen=fopen("C:\\mingwen.txt","r");

for(i=0;i<length;i++) //读入明文

fscanf(mingwen,"%c",&S[i]);

S[length]='\0';

fclose(mingwen);

printf("\n读入的明文为: ");

puts(S);

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 读入密钥 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

i=0;

miyao=fopen("C:\\miyao.txt","r");

while(i<64)

fscanf(miyao,"%d",&my[i++]);

fclose(miyao);

printf("\n//\*\*\*\*\*\*\* 读入的64位密钥为\*\*\*\*\*\*\*//\n");

for(i=0;i<64;i++)

{

if(i%8==0)

printf("\n");

printf(" %d",my[i]);

}

printf("\n");

//\*\*\*\*\*\*\*\*\*\*\*计算明文个数是否是8的倍数，若不是加0.因为每8个组成一个64位的二进制\*\*\*\*\*\*\*//

length=strlen(S)/8;

k=1;

for(k=1;k<length;k++)

{

j=0;

for(i=0;i<8;i++)

{

n=0;

m=S[8\*(k-1)+i];

while(m) //把明文字符的ASCII值变为二进制//

{

mw[j++]=m&1;

m=m>>1;

n++;

}

while(n<8) //当ASCII值变为二进制不为8位时补零

{

mw[j++]=0;

n++;

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 进行加密过程 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

if(j==64)

{

for(i=0;i<64;i++)

{

if(i%8==0)

printf("\n");

printf(" %d",mw[i]);

}

Enc(mw,my);

}

}

length=strlen(S)%8;

if(length)

{

k=8\*(k-1);

j=0;

length=strlen(S);

for(k;k<=length-1;k++)

{

n=0;

m=S[k];

while(m) //把明文字符的ASCII值变为二进制//

{

mw[j++]=m&1;

m=m>>1;

n++;

}

while(n<8) //当ASCII值变为二进制不为8位时补零

{

mw[j++]=0;

n++;

}

}

while(j<64)

{

mw[j++]=0;

}

Enc(mw,my);

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 主函数 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*//

void main()

{

int n;

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("\n \*\*\*\*\*\*\*\*\*\*\*\*\*\*　这是一个DES加解密程序　\*\*\*\*\*\*\*\*\*\*\*\*\* \n");

printf("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

while(1)

{

printf("\n请选择:1 加密 2 解密 3 退出\n");

printf("\n请输入: ");

scanf("%d",&n);

if(n==1)

{

printf("\n你选取的是加密\n");

printf("\n请在C盘根目录下建立一个名为mingwen.txt的文件,做为明文的输入\n");

printf("\n是否建立完成(1 是 2 否): ");

scanf("%d",&n);

if(n==1)

{

Encryption();

}

}

else

if(n==2)

{

printf("\n你选取的是解密\n");

printf("\nC盘根目录下是否有名为miwen.txt,和miyao1.txt的文件,做为解密的输入\n");

printf("\n(1 是 2 否): ");

scanf("%d",&n);

if(n==1)

{

Decryption();

}

}

else

if(n==3)

exit(0);

else

printf("输入错误");

}}

1. **运行结果**

