分组密码算法DES

1. 程序设计

本程序中输入的明文、密钥，加密生成的密文，以及解密生成的明文的存储格式都是字符串格式，存储它们的二进制数形式。

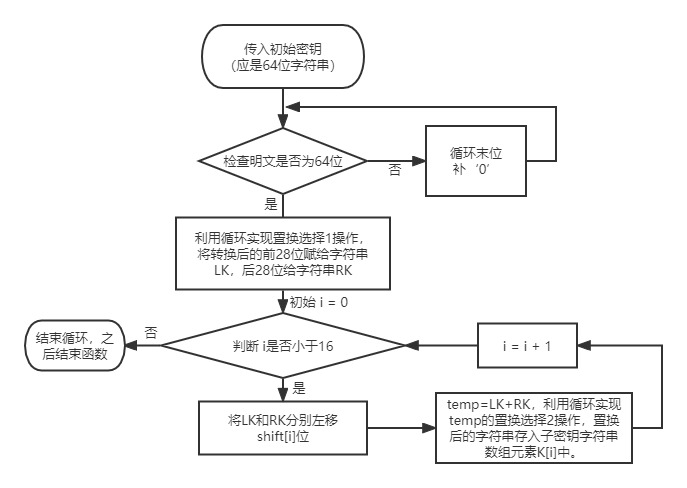
程序中的函数除了main函数外一共有五个，它们的作用分别是密钥生成、DES加密函数、DES解密函数、异或函数、将十进制整数转变为二进制整数的函数。

其中，由于程序中存储明文、密文、密钥的格式都是字符串，因此，实现了异或函数，即按位比对异或操作的两个操作数，相同结果字符串+‘0’字符，不相同结果字符串+‘1’字符。另外，S盒中存储的数都是十进制整数，因此在进行每个S盒代换时，调用了将十进制整数转变为二进制整数的函数。函数的实现就是利用短除法的方式将余数用字符串的形式存储，并在转换完毕后，将字符串倒过来，返回的是4位二进制数的字符串。

下面重点介绍密钥生成、DES加密函数、DES解密函数的实现：

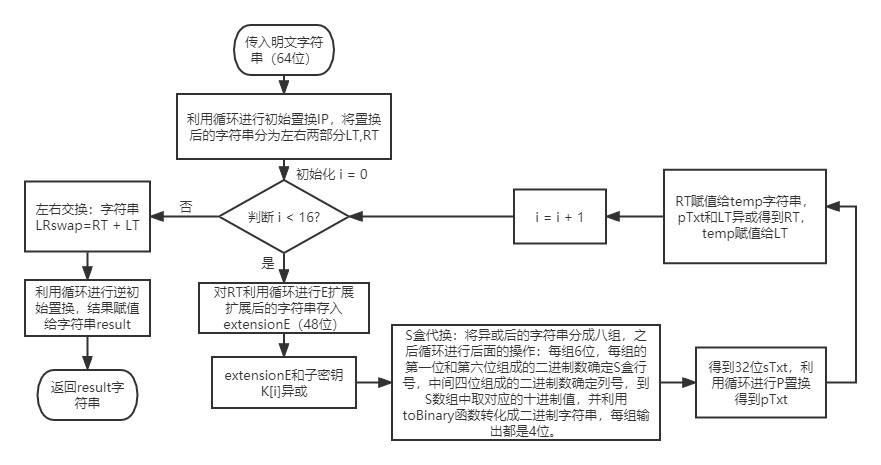
1. 密钥生成

传入初始密钥64位后，经过检验密钥长度是否为64位，不足补‘0’，经过置换选择1变成56位，之后进行16轮的移位和置换选择2操作，将每轮生成的子密钥存入子密钥字符串数组元素K[i]中。



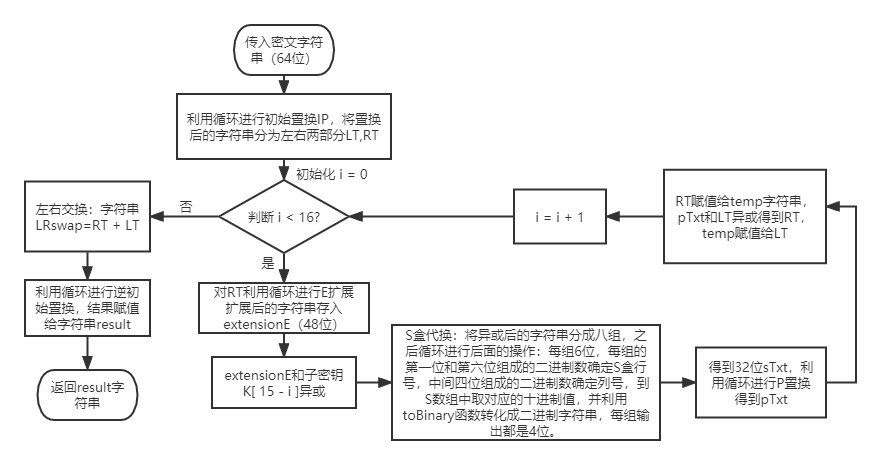
1. DES加密

加密流程：先对明文进行初始置换，再进入16轮加密循环中，每轮先对右半部分进行E扩展，再和子密钥异或，之后进行S盒代换，再进行P置换，最后将P置换后的值和左半部分异或后赋值给右半部分，将原来右半部分的值赋值给左半部分。



1. DES解密

解密过程和加密过程流程基本一样，不同之处在于每轮使用的子密钥是倒着使用的。



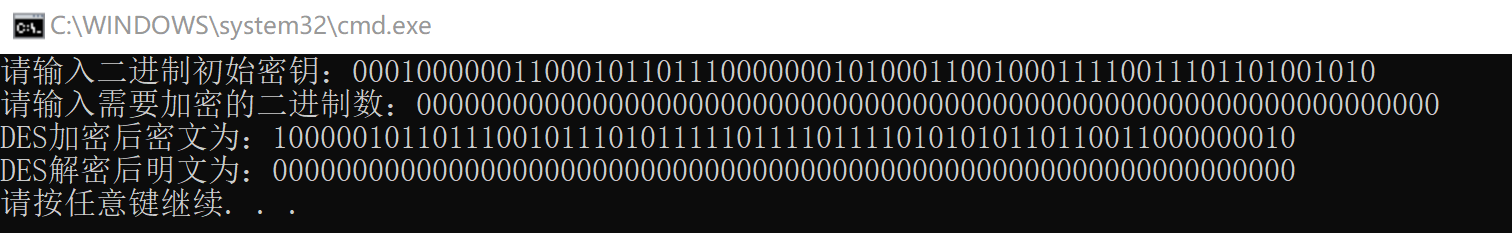
1. 实验结果展示

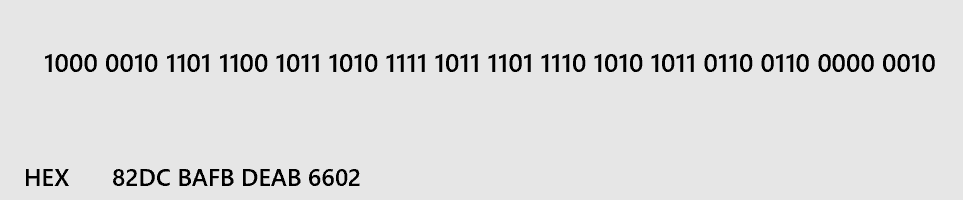
这个程序需要输入二进制初始密钥和需要DES加密的二进制数明文，之后经过计算，会输出加密后的二进制密文，并对二进制密文进行DES解密，得到二进制明文。经过对比我们发现解密后的明文和之前输入的明文都相同。

1、{ 0x10, 0x31, 0x6E, 0x02, 0x8C, 0x8F, 0x3B, 0x4A },

{ 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 },

{ 0x82, 0xDC, 0xBA, 0xFB, 0xDE, 0xAB, 0x66, 0x02 }

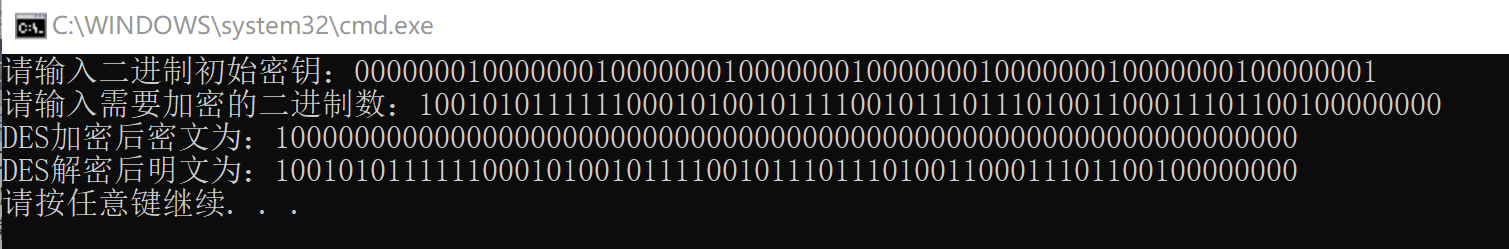




2、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0x95, 0xF8, 0xA5, 0xE5, 0xDD, 0x31, 0xD9, 0x00 },

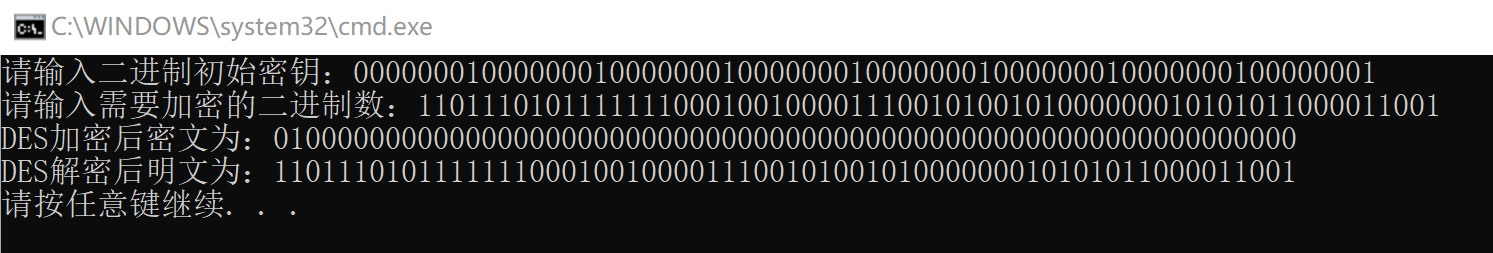
{ 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



3、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0xDD, 0x7F, 0x12, 0x1C, 0xA5, 0x01, 0x56, 0x19 },

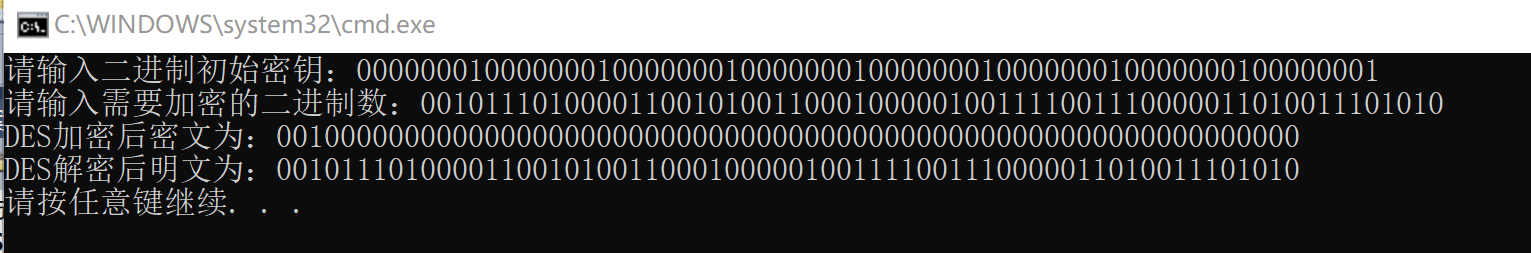
{ 0x40, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



4、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0x2E, 0x86, 0x53, 0x10, 0x4F, 0x38, 0x34, 0xEA },

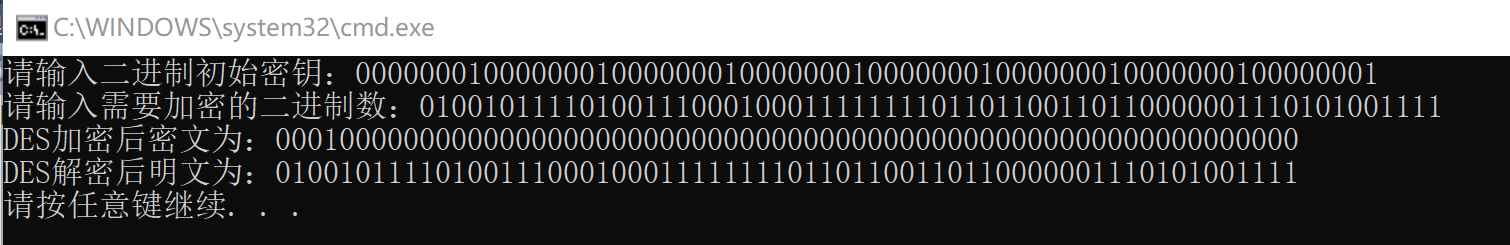
{ 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



5、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0x4B, 0xD3, 0x88, 0xFF, 0x6C, 0xD8, 0x1D, 0x4F },

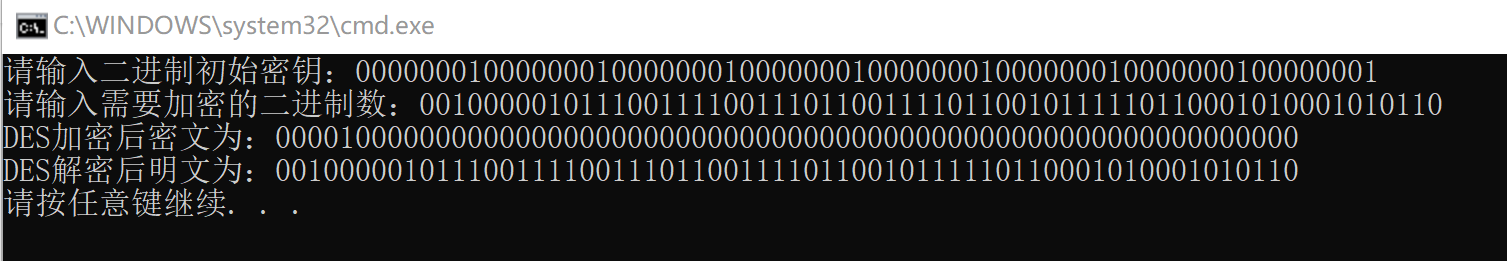
{ 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



6、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0x20, 0xB9, 0xE7, 0x67, 0xB2, 0xFB, 0x14, 0x56 },

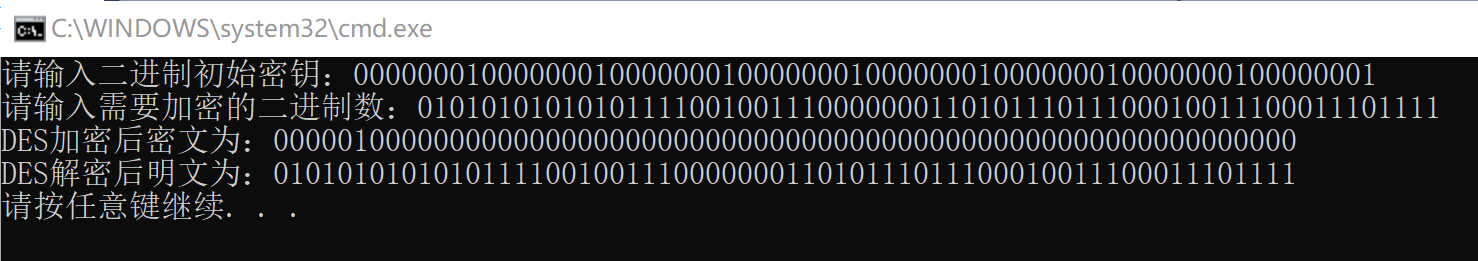
{ 0x08, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



7、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0x55, 0x57, 0x93, 0x80, 0xD7, 0x71, 0x38, 0xEF },

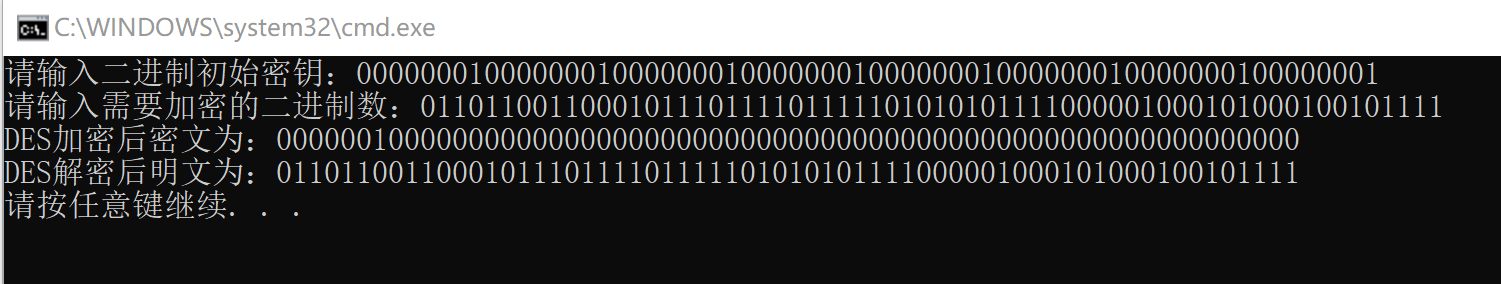
{ 0x04, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



8、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0x6C, 0xC5, 0xDE, 0xFA, 0xAF, 0x04, 0x51, 0x2F },

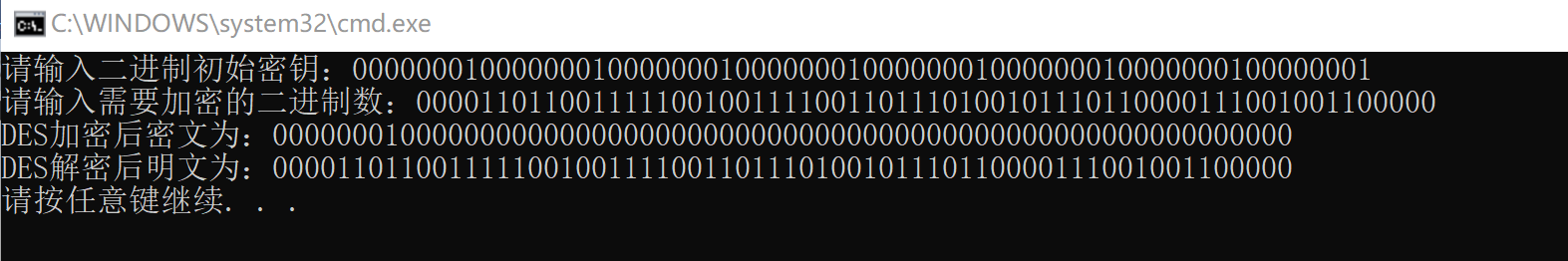
{ 0x02, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



9、 { 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0x0D, 0x9F, 0x27, 0x9B, 0xA5, 0xD8, 0x72, 0x60 },

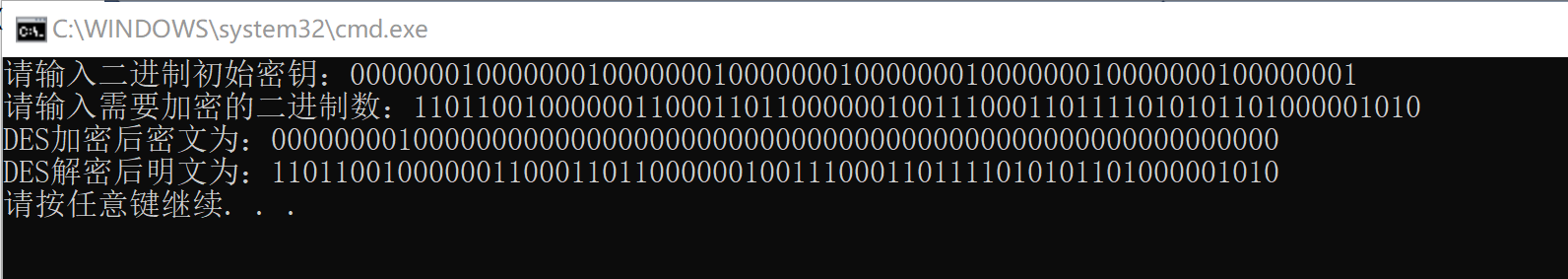
{ 0x01, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



10、{ 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 },

{ 0xD9, 0x03, 0x1B, 0x02, 0x71, 0xBD, 0x5A, 0x0A },

{ 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }



三、雪崩效应检验

（一）固定密钥，任意改变一位明文，计算密文改变的位数。

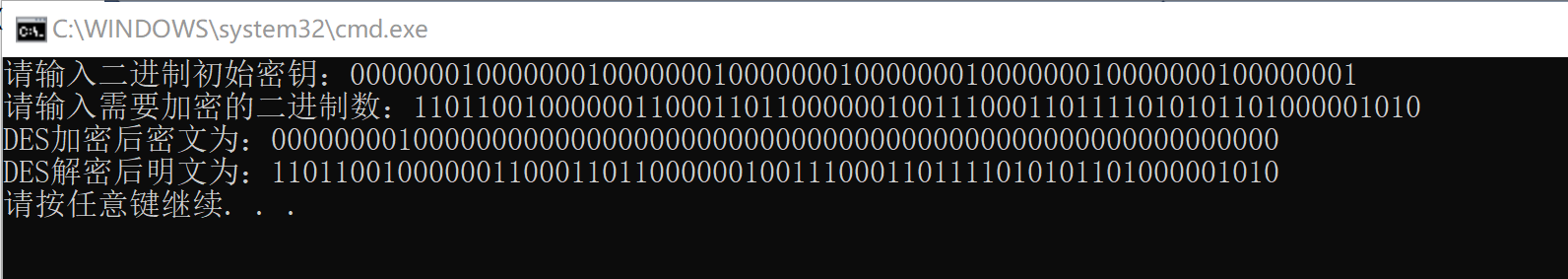
固定密钥为：0000000100000001000000010000000100000001000000010000000100000001

（0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01）

初始明文：1101100100000011000110110000001001110001101111010101101000001010

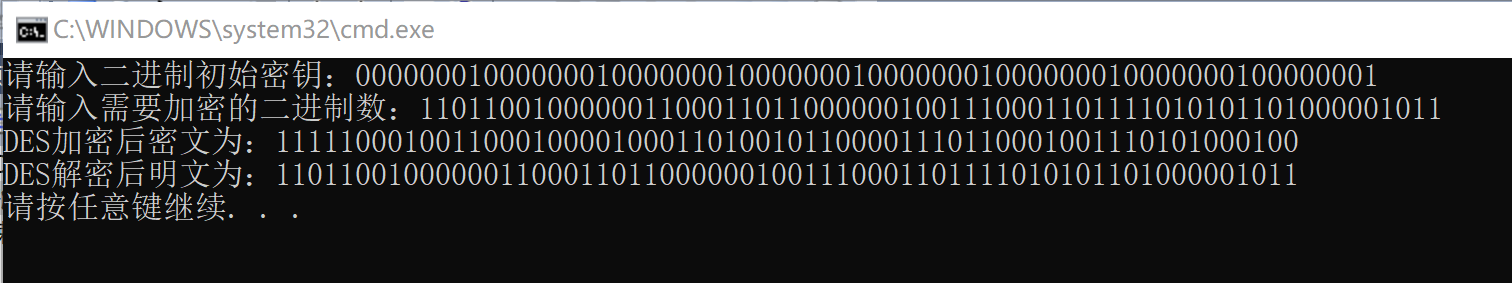
（0xD9, 0x03, 0x1B, 0x02, 0x71, 0xBD, 0x5A, 0x0A）

结果：密文为（0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 ）



1、改变一位明文：1101100100000011000110110000001001110001101111010101101000001011

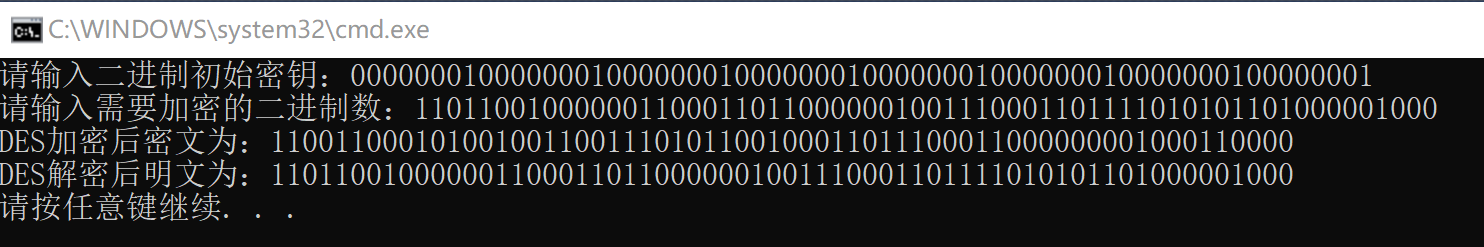
结果：密文如下所示，相比之前改变了27位



1. 改变后明文：

1101100100000011000110110000001001110001101111010101101000001000

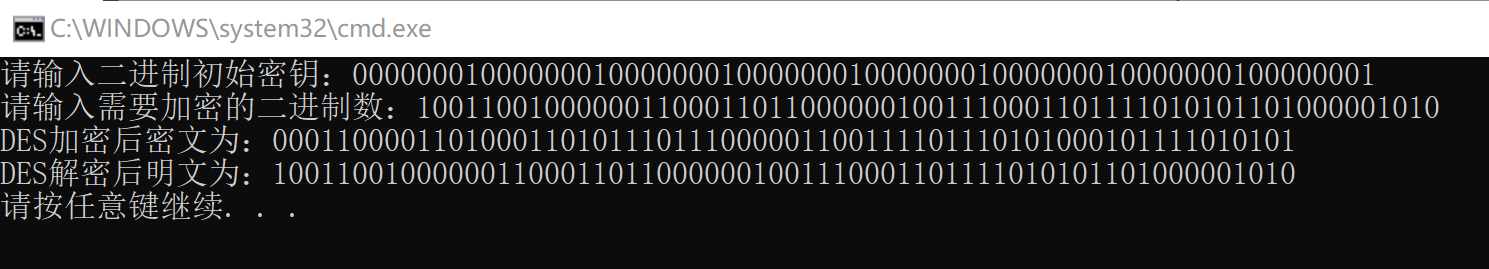
结果：密文如下所示，相比最初改变了27位



1. 改变后明文：

1001100100000011000110110000001001110001101111010101101000001010

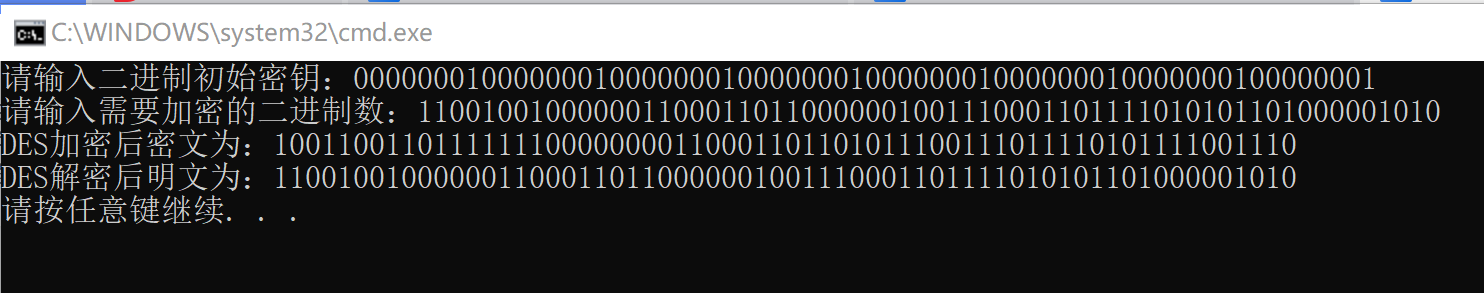
结果：密文如下所示，相比最初改变了34位



1. 改变后明文：

1100100100000011000110110000001001110001101111010101101000001010

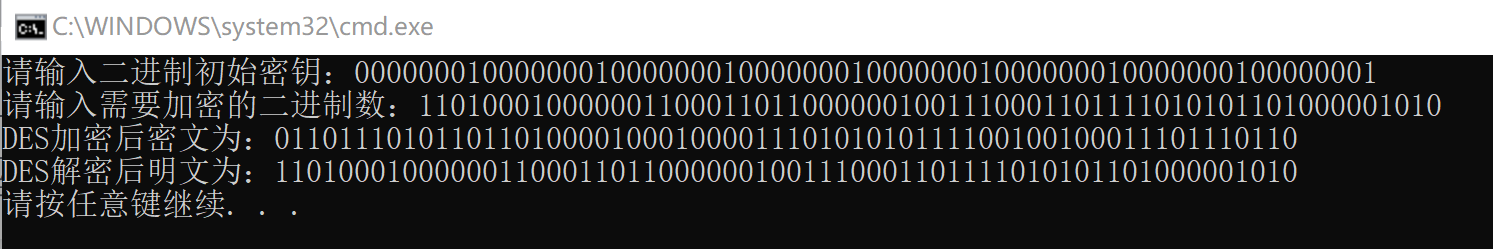
结果：密文如下所示，相比最初改变了36位



1. 改变后明文：

1101000100000011000110110000001001110001101111010101101000001010

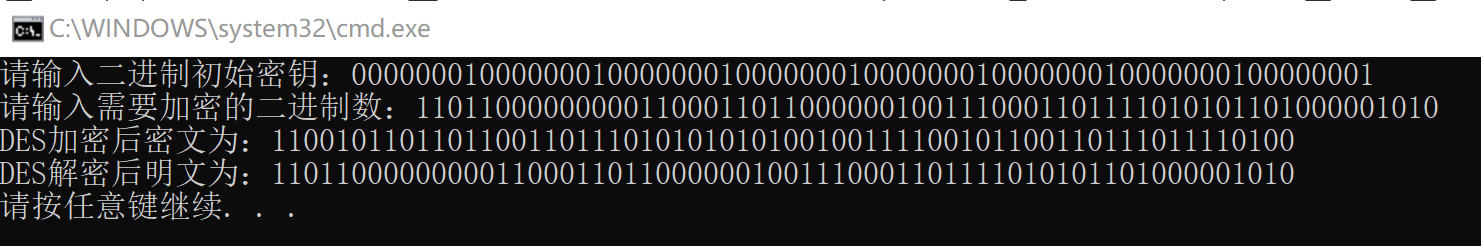
结果：密文如下所示，相比最初改变了32位



1. 改变后明文：

1101100000000011000110110000001001110001101111010101101000001010

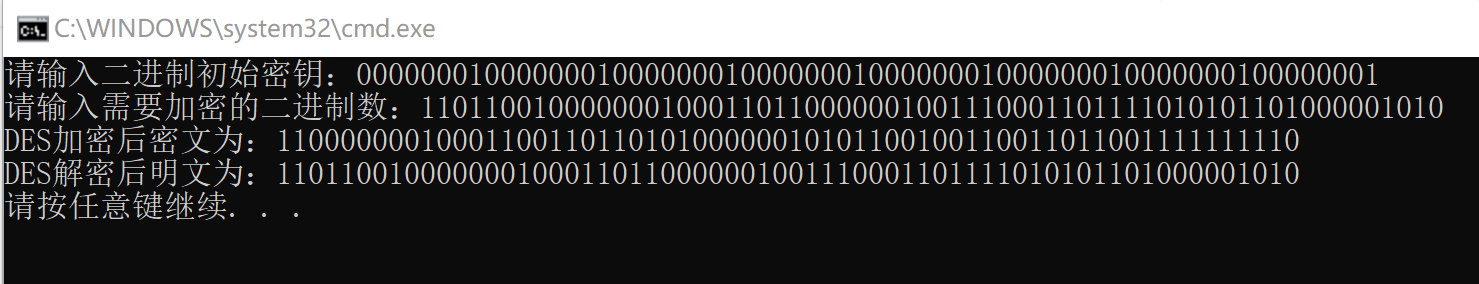
结果：密文如下所示，相比最初改变了38位



1. 改变后明文：

1101100100000001000110110000001001110001101111010101101000001010

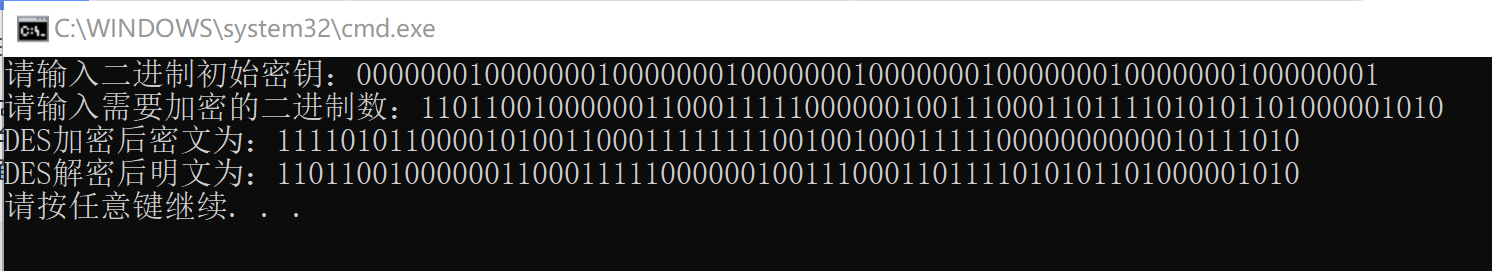
结果：密文如下所示，相比最初改变了32位



1. 改变后明文：

1101100100000011000111110000001001110001101111010101101000001010

结果：密文如下所示，相比最初改变了30位



因此，对于固定密钥，每次改变明文中的一位，最终可以得到密文平均变化位数为：(30+32+38+32+36+34+27+27)/8 = 32 位

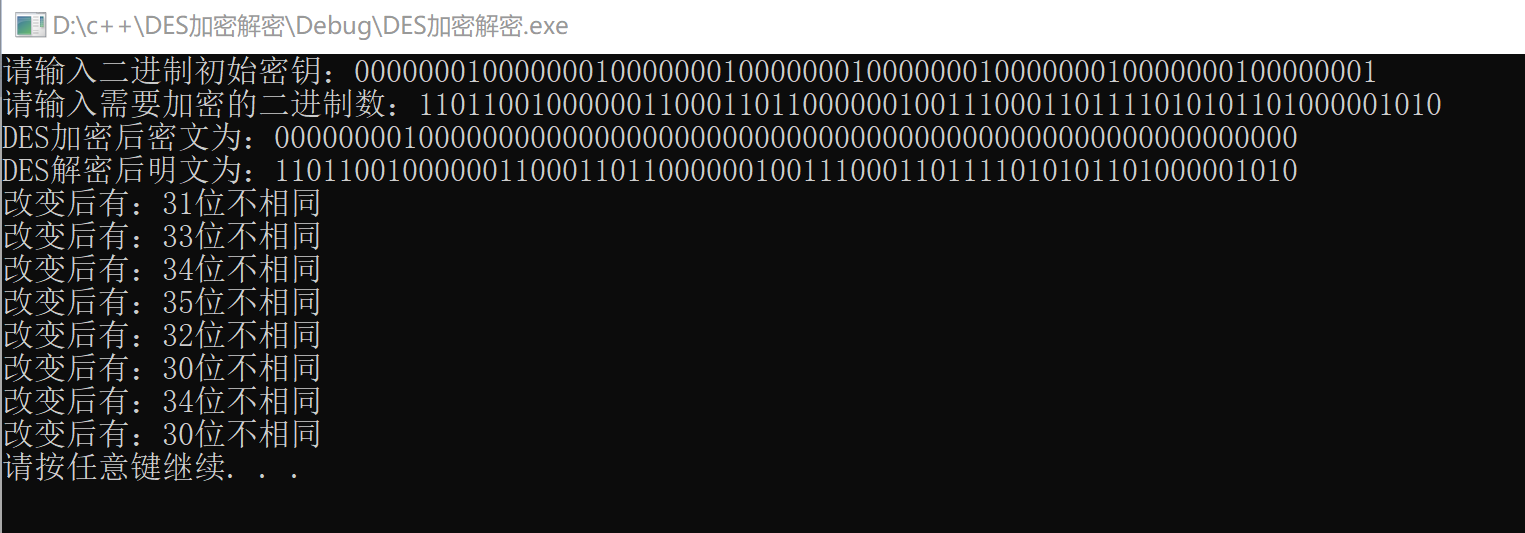
（二）固定密钥，任意改变一位密文，计算明文改变的位数。

固定密钥为：0000000100000001000000010000000100000001000000010000000100000001

初始密文：{ 0x00, 0x80, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 }

编写程序，分别改变第0，8，16，24，32，40，48，56位。

结果展示如下：



因此，对于固定密钥，每次改变一位密文，最终得到解密后的明文平均变化位数为：(31+33+34+35+32+30+34+30)/8 = 32.375 位