S-DES算法实验

**第1关：基本测试**

根据S-DES算法编写和调试程序，提供GUI解密支持用户交互。输入可以是8bit的数据和10bit的密钥，输出是8bit的密文。

8bit数加密解密ui



测试明文：10011001 密钥：1000110001

输入8bit明文与10bit密钥后生成密文，再次输入对应10bit密钥后可得到最终的明文，解密成功，第一关通过。

**示例代码**

1. 子密钥生成

// 生成子密钥，包括P10置换vector<bitset<8>> S\_Des::**generateSubKeys**(bitset<10> key) { vector<bitset<8>> subKeys; bitset<10> permutedKey = permute10(key);

bitset<5> leftKey;

bitset<5> rightKey;

for (int i = 0; i < 5; i++) { leftKey[i] = permutedKey[i]; rightKey[i] = permutedKey[i + 5]; }

//生成两把子密钥

for (int i = 0; i < 2; i++) { leftKey = (leftKey << shift[i]) | (leftKey >> (5 - shift[i])); rightKey = (rightKey << shift[i]) | (rightKey >> (5 - shift[i])); bitset<10> roundKey = (leftKey.to\_ulong() << 5) | rightKey.to\_ulong();

// P8 置换，将子密钥缩减为8位 bitset<8> roundKey1 = permute8(roundKey);

// 将 roundKey 添加到 subKeys 向量中 subKeys.push\_back(roundKey1); } /\*cout << "两把子密钥分别为:" << endl; for (const bitset<8>&roundKey : subKeys) { cout << roundKey << endl; }\*/ return subKeys;}

1. 加密解密算法

// S-DES加密bitset<8> S\_Des::**sDesEncrypt**(bitset<8> plaintext, bitset<10> key) { vector<bitset<8>> subKeys = generateSubKeys(key); bitset<8> ipResult = initialPermutation(plaintext); //cout << "加密的初始置换：" << ipResult << endl ;

bitset<4> left = (ipResult >> 4).to\_ulong(); bitset<4> right = (ipResult & bitset<8>(0x0F)).to\_ulong();

bitset<4> rightTemp = right; right = left ^ roundFunction(right, subKeys[0]); left = rightTemp; left = left ^ roundFunction(right, subKeys[1]);

bitset<8> ciphertext = (left.to\_ulong() << 4) | right.to\_ulong(); //cout << "加密最终置换前：" << ciphertext << endl; return inverseInitialPermutation(ciphertext);}

// S-DES解密bitset<8> S\_Des::**sDesDecrypt**(bitset<8> ciphertext, bitset<10> key) { vector<bitset<8>> subKeys = generateSubKeys(key); bitset<8> ipResult = initialPermutation(ciphertext); cout << "解密的初始置换：" << ipResult <<endl;

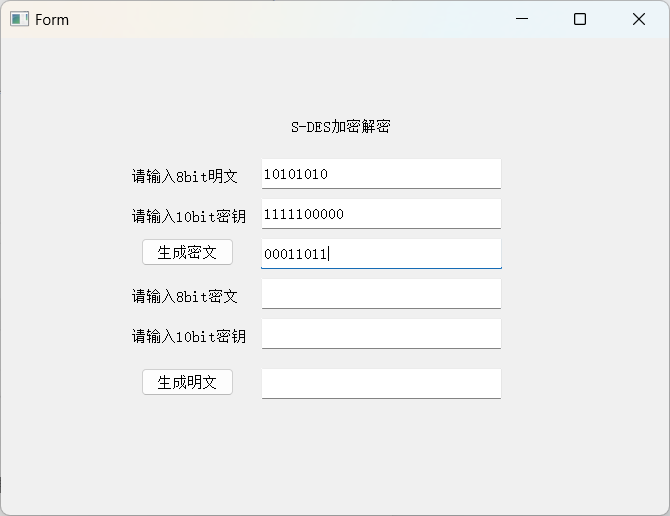
bitset<4> left = (ipResult >> 4).to\_ulong(); bitset<4> right = (ipResult & bitset<8>(0x0F)).to\_ulong();

bitset<4> rightTemp = right; right = left ^ roundFunction(right, subKeys[1]); left = rightTemp; left = left ^ roundFunction(right, subKeys[0]);

bitset<8> plaintext = (left.to\_ulong() << 4) | right.to\_ulong(); cout << "解密最终置换前：" << plaintext << endl; return inverseInitialPermutation(plaintext);}

第2关：交叉测试

考虑到是**算法标准**，所有人在编写程序的时候需要使用相同算法流程和转换单元(P-Box、S-Box等)，以保证算法和程序在异构的系统或平台上都可以正常运行。

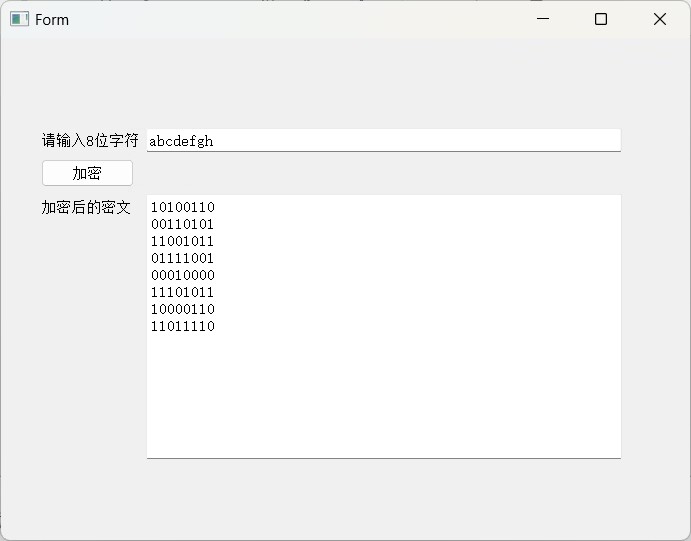




测试结果：与其他组的明文与密钥所生成密文相同，测试通过！

第3关：扩展功能

首先将每一个字符进行ascii编码转换，转换为8位的二进制编码，再对二进制编码分别进行des加密，最后进行密文展示。第三关，拿下！



ascll码加密

关键代码如下：

//把str转换为ascii-bitvoid S\_Des::**convertToASCII**(const char\* str, bitset<8>\* asciiArray, int length) { for (int i = 0; i < length; ++i) { asciiArray[i] = static\_cast<int>(str[i]); }}

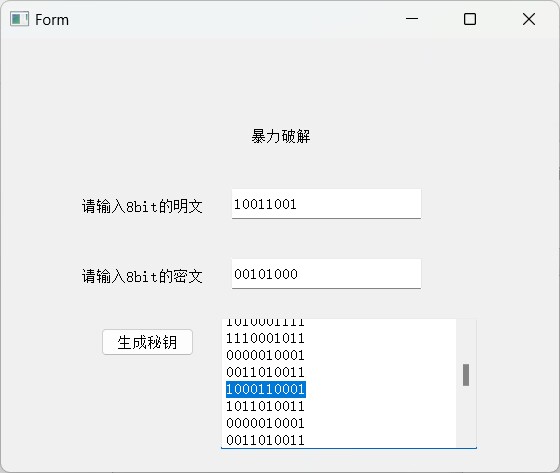
//把ascii-bit进行des加密void S\_Des::**convertToDes**(bitset<8>\* desArray,bitset<8>\* asciiArray, int length){ for (int i=0;i<length;i++){ desArray[i]=sDesEncrypt(asciiArray[i],key); }}

//对des进行解密成ascii-bitvoid S\_Des::**decryptDes**(bitset<8>\* desArray, bitset<8>\* asciiArray,int length){ for (int i=0;i<length;i++){ asciiArray[i]=sDesDecrypt(desArray[i],key); }}

//将ascii-bit再转换成desvoid S\_Des::**convertToChar**(bitset<8>\* asciiArray, char\* charArray, int length) { for (int i = 0; i < length; ++i) { charArray[i] = static\_cast<char>((asciiArray[i].to\_ulong())); }}

第4关：暴力破解

通过for循环对每一个可能的key进行遍历，找到符合要求的key之后添加到数组中。因为一个明密文对会有多个可能的key，所以我们用一个数组保存所有符合要求的key。最后成功破解。第四关，easy！！！



第5关：封闭测试

根据第四关的结果，我们可以知道选择不同的密钥，是有可能加密出相同的密文的。

第五关，通过！

**用户指南**

**开发工具**

Qt5.14.2，C++

**操作方法**

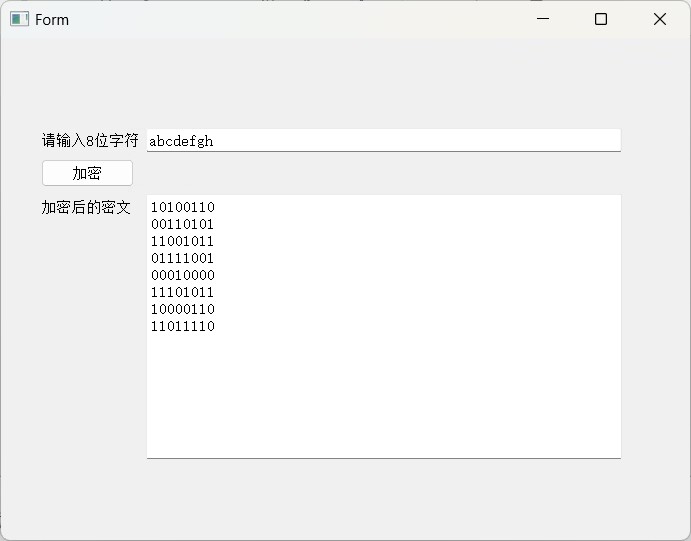


**1.生成密文**

**在这个界面中，请在第一个输入框中输入您想加密的明文，在第二个输入框中输入您的密钥，点击按钮之后将会生成密文。**

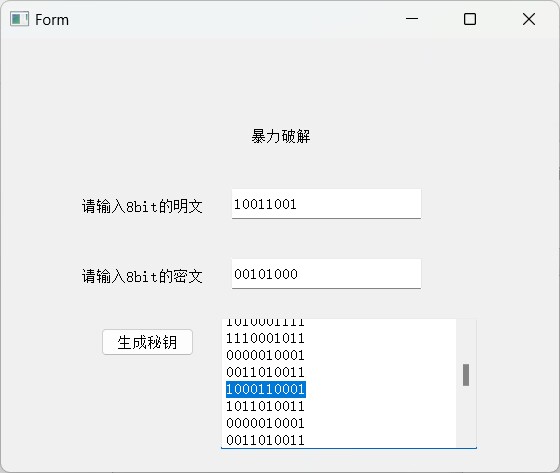
**2.密文解密**

**在第三第四个输入框中分别输入您的密文和密钥之后，点击“生成明文”按钮之后，便会对您的密文进行解密。最后显示出解密结果**



1. **ascii码加密**

**在第一个输入框中输入你想加密的字符，点击加密之后便会生成对应的密文。**



1. **暴力破解**

**在输入框中分别输入您的密文和明文，点击“生成秘钥”之后便会生成可以得到该密文的密钥。**