Information Security HW2

- ## 分工

四資工三 B10615033 王璽禎 Encrypt
四資工三 B10615035 陳靜萱 Decrypt

- ## 開發環境

Visual studio 2017 C++

- ## 範例

1. DES Encrypt：

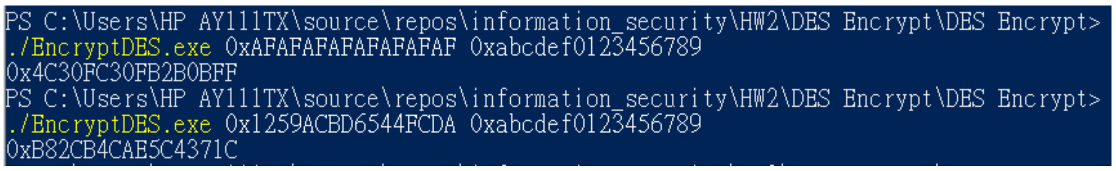
Input:

Key: 0xAFAFAFAFAFAFAFAF

Plaintext: 0xabcdef0123456789

Output:

ciphertext: **0x4C30FC30FB2B0BFF**



```
PS C:\Users\HP AY111TX\source\repos\information_security\HW2\DES Encrypt\DES Encrypt>
./EncryptDES.exe 0xAFAFAFAFAFAFAFAF 0xabcdef0123456789
0x4C30FC30FB2B0BFF
PS C:\Users\HP AY111TX\source\repos\information_security\HW2\DES Encrypt\DES Encrypt>
./EncryptDES.exe 0x1259ACBD6544FCDA 0xabcdef0123456789
0xB82CB4CAE5C4371C
```

2. DES Decrypt：

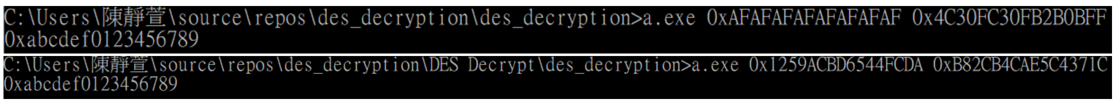
Input:

Key: 0xAFAFAFAFAFAFAFAF

ciphertext: 0x4C30FC30FB2B0BFF

Output:

plaintext: **0xabcdef0123456789**



```
C:\Users\陳靜萱\source\repos\des_decryption\des_decryption>a.exe 0xAFAFAFAFAFAFAFAF 0x4C30FC30FB2B0BFF
0xabcdef0123456789
C:\Users\陳靜萱\source\repos\des_decryption\DES Decrypt\des_decryption>a.exe 0x1259ACBD6544FCDA 0xB82CB4CAE5C4371C
0xabcdef0123456789
```

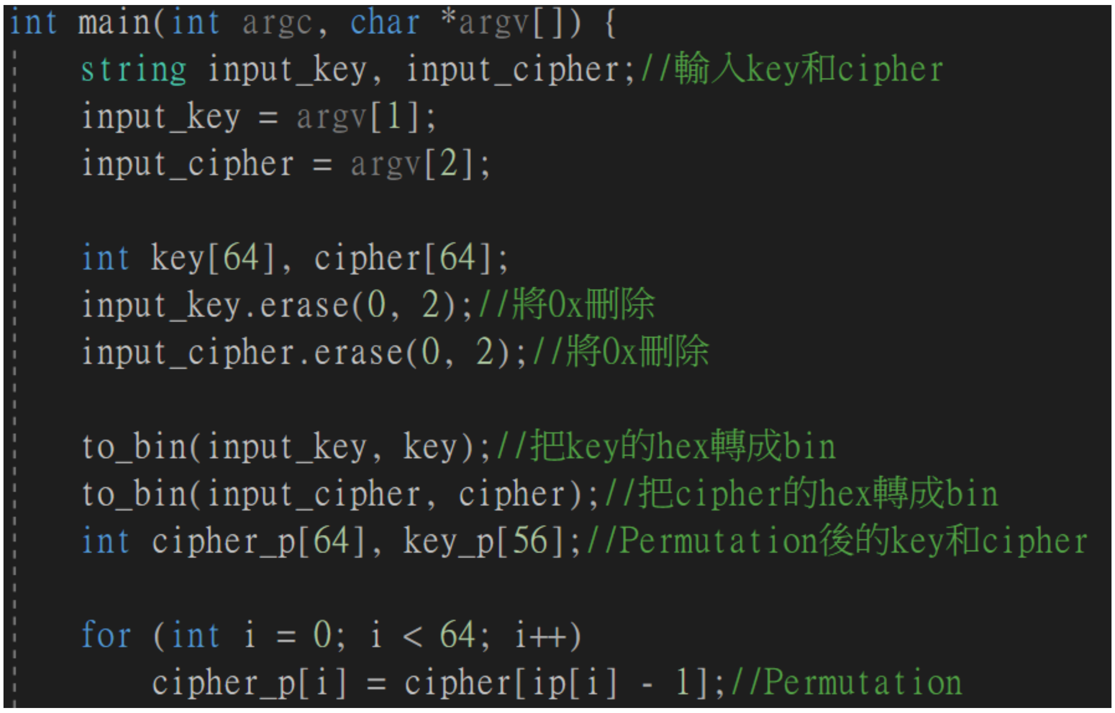
- ## 心得與遇到的困難

在撰寫 DES 的過程中，整體而言不算太難，只是步驟複雜了點，當程式 compile 後沒有任何 error，但 output 卻不是正解，要 debug 真的有點麻煩，但還好 partner 之間互相合作，馬上就找出不同點在哪，

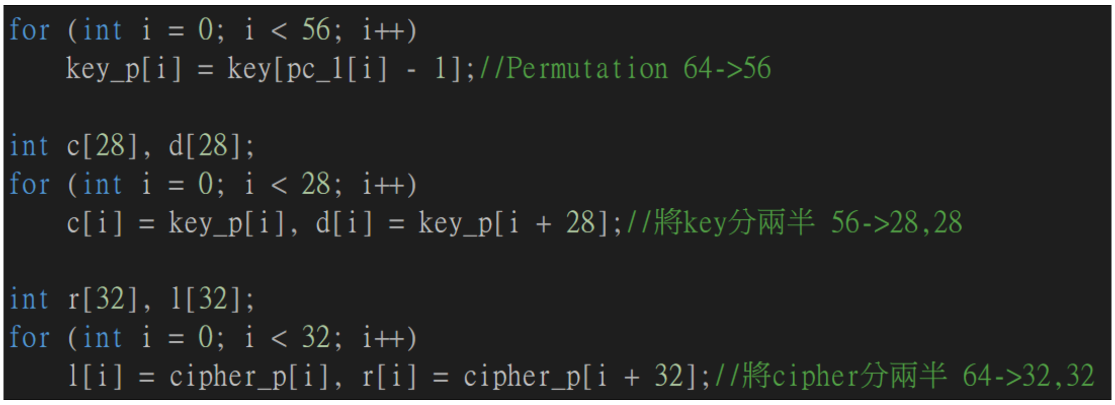
順利完成 DES 加解密！

程式碼及註解：

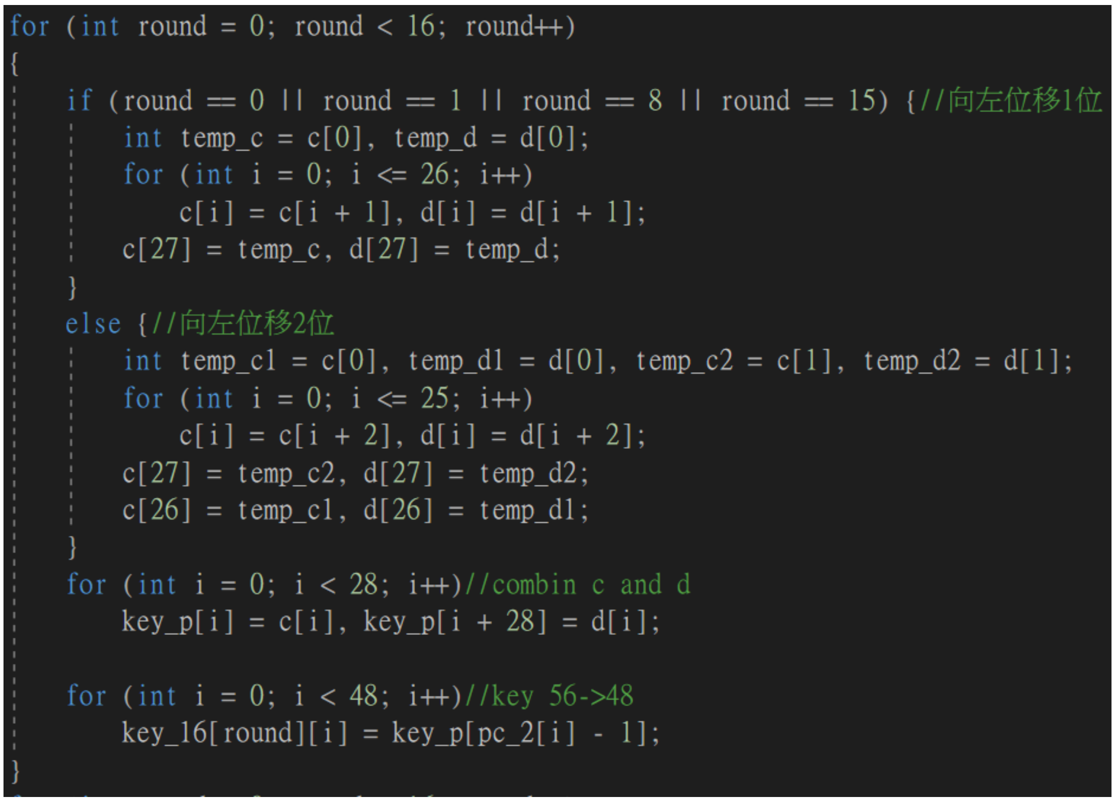
解密只有 key 需要作變化，其他都跟加密相似。



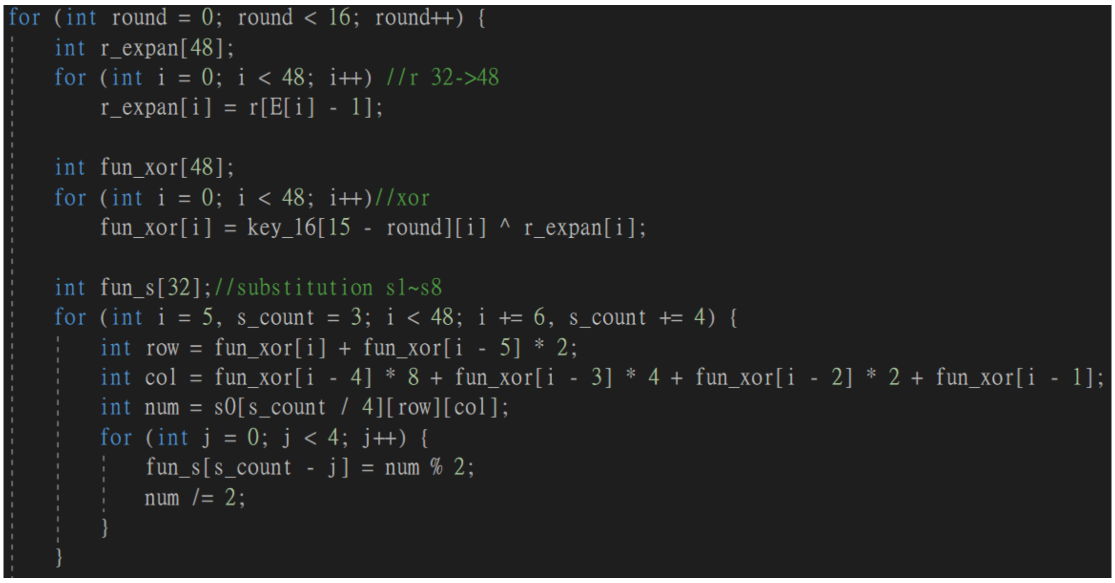
```
int main(int argc, char *argv[]) {
    string input_key, input_cipher;//輸入key和cipher
    input_key = argv[1];
    input_cipher = argv[2];

    int key[64], cipher[64];
    input_key.erase(0, 2);//將0x刪除
    input_cipher.erase(0, 2);//將0x刪除

    to_bin(input_key, key);//把key的hex轉成bin
    to_bin(input_cipher, cipher);//把cipher的hex轉成bin
    int cipher_p[64], key_p[56];//Permutation後的key和cipher

    for (int i = 0; i < 64; i++)
        cipher_p[i] = cipher[ip[i] - 1];//Permutation
```



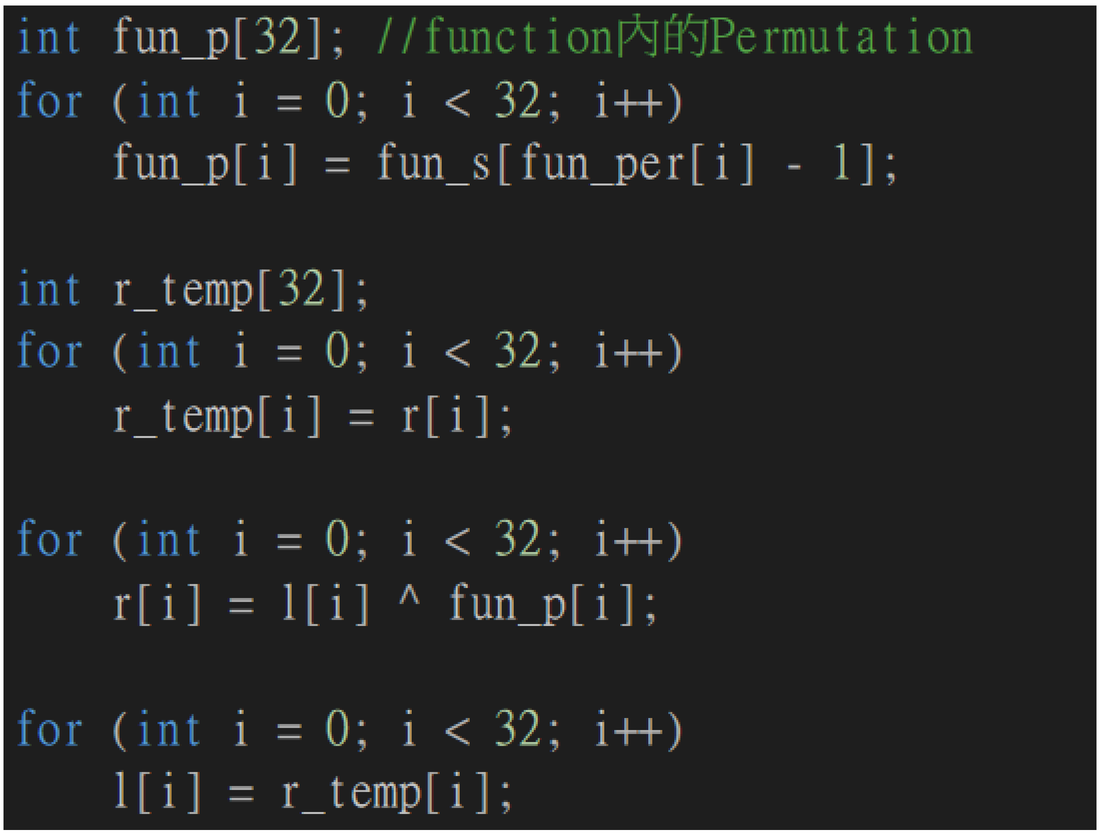
```
for (int i = 0; i < 56; i++)
    key_p[i] = key[pc_1[i] - 1];//Permutation 64->56

int c[28], d[28];
for (int i = 0; i < 28; i++)
    c[i] = key_p[i], d[i] = key_p[i + 28];//將key分兩半 56->28,28

int r[32], l[32];
for (int i = 0; i < 32; i++)
    l[i] = cipher_p[i], r[i] = cipher_p[i + 32];//將cipher分兩半 64->32,32
```



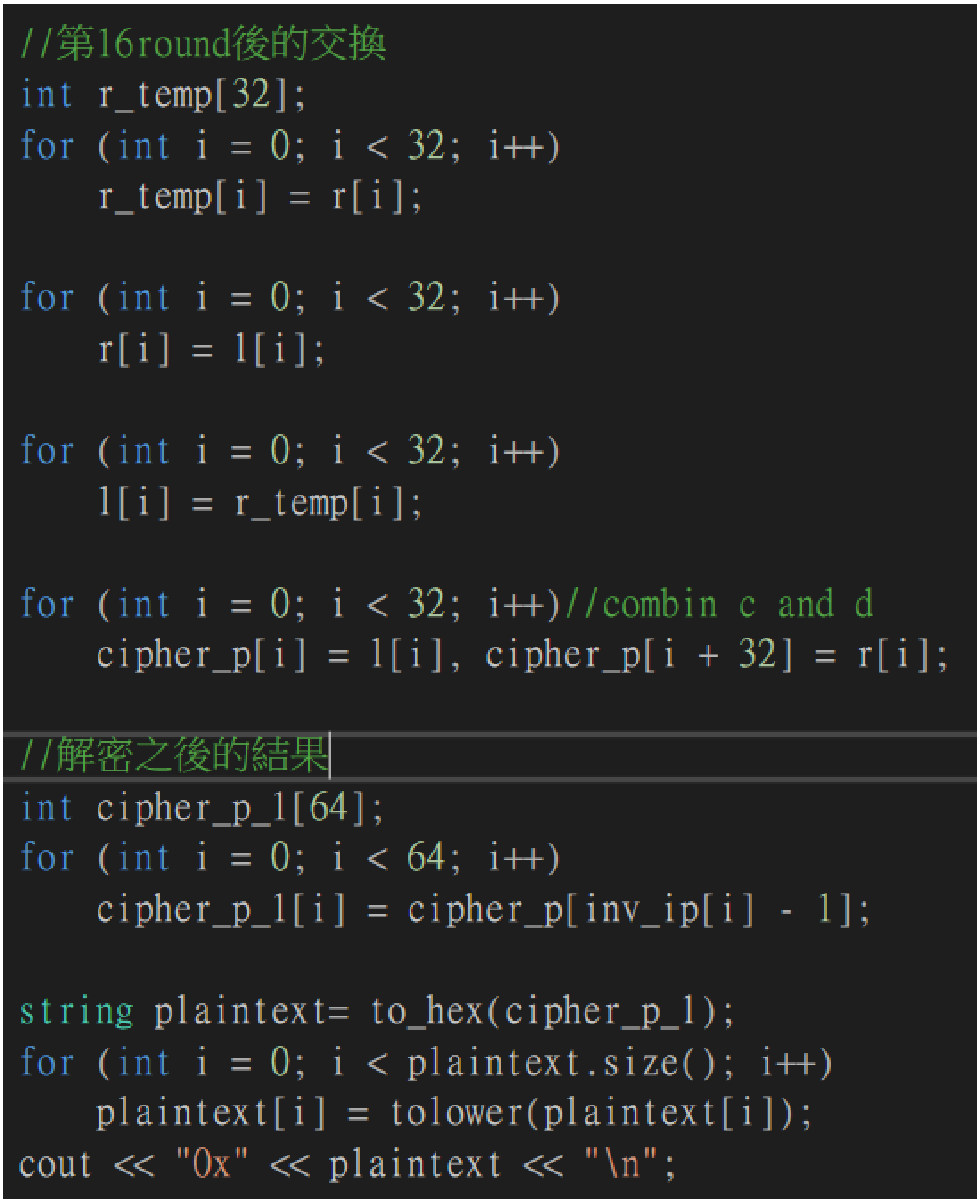
```
for (int round = 0; round < 16; round++)
{
    if (round == 0 || round == 1 || round == 8 || round == 15) {//向左位移1位
        int temp_c = c[0], temp_d = d[0];
        for (int i = 0; i <= 26; i++)
            c[i] = c[i + 1], d[i] = d[i + 1];
        c[27] = temp_c, d[27] = temp_d;
    }
    else {//向左位移2位
        int temp_c1 = c[0], temp_d1 = d[0], temp_c2 = c[1], temp_d2 = d[1];
        for (int i = 0; i <= 25; i++)
            c[i] = c[i + 2], d[i] = d[i + 2];
        c[27] = temp_c2, d[27] = temp_d2;
        c[26] = temp_c1, d[26] = temp_d1;
    }
    for (int i = 0; i < 28; i++)//combin c and d
        key_p[i] = c[i], key_p[i + 28] = d[i];

    for (int i = 0; i < 48; i++)//key 56->48
        key_16[round][i] = key_p[pc_2[i] - 1];
}
```



```
for (int round = 0; round < 16; round++) {
    int r_expan[48];
    for (int i = 0; i < 48; i++) //r 32->48
        r_expan[i] = r[E[i] - 1];

    int fun_xor[48];
    for (int i = 0; i < 48; i++)//xor
        fun_xor[i] = key_16[15 - round][i] ^ r_expan[i];

    int fun_s[32];//substitution s1~s8
    for (int i = 5, s_count = 3; i < 48; i += 6, s_count += 4) {
        int row = fun_xor[i] + fun_xor[i - 5] * 2;
        int col = fun_xor[i - 4] * 8 + fun_xor[i - 3] * 4 + fun_xor[i - 2] * 2 + fun_xor[i - 1];
        int num = s0[s_count / 4][row][col];
        for (int j = 0; j < 4; j++) {
            fun_s[s_count - j] = num % 2;
            num /= 2;
        }
    }
```



```
int fun_p[32]; //function內的Permutation
for (int i = 0; i < 32; i++)
    fun_p[i] = fun_s[fun_per[i] - 1];

int r_temp[32];
for (int i = 0; i < 32; i++)
    r_temp[i] = r[i];

for (int i = 0; i < 32; i++)
    r[i] = l[i] ^ fun_p[i];

for (int i = 0; i < 32; i++)
    l[i] = r_temp[i];
```



```
//第16round後的交換
int r_temp[32];
for (int i = 0; i < 32; i++)
    r_temp[i] = r[i];

for (int i = 0; i < 32; i++)
    r[i] = l[i];

for (int i = 0; i < 32; i++)
    l[i] = r_temp[i];

for (int i = 0; i < 32; i++)//combin c and d
    cipher_p[i] = l[i], cipher_p[i + 32] = r[i];

//解密之後的結果
int cipher_p_1[64];
for (int i = 0; i < 64; i++)
    cipher_p_1[i] = cipher_p[inv_ip[i] - 1];

string plaintext= to_hex(cipher_p_1);
for (int i = 0; i < plaintext.size(); i++)
    plaintext[i] = tolower(plaintext[i]);
cout << "0x" << plaintext << "\n";
```