

南开大学

## 汇编语言与逆向技术课程实验报告

### 实验八： Reverse Engineering Exercises – Simple



学 院 网络空间安全学院  
专 业 信息安全、法学双学位  
学 号 2212000  
姓 名 宋奕纬  
班 级 1061

## 一、实验目的

- 1、熟悉静态反汇编工具 IDA Freeware;
- 2、熟悉反汇编代码的逆向分析过程;
- 3、掌握反汇编语言中的数学计算、数据结构、条件判断、分支结构的识别和逆向分析

## 二、实验原理

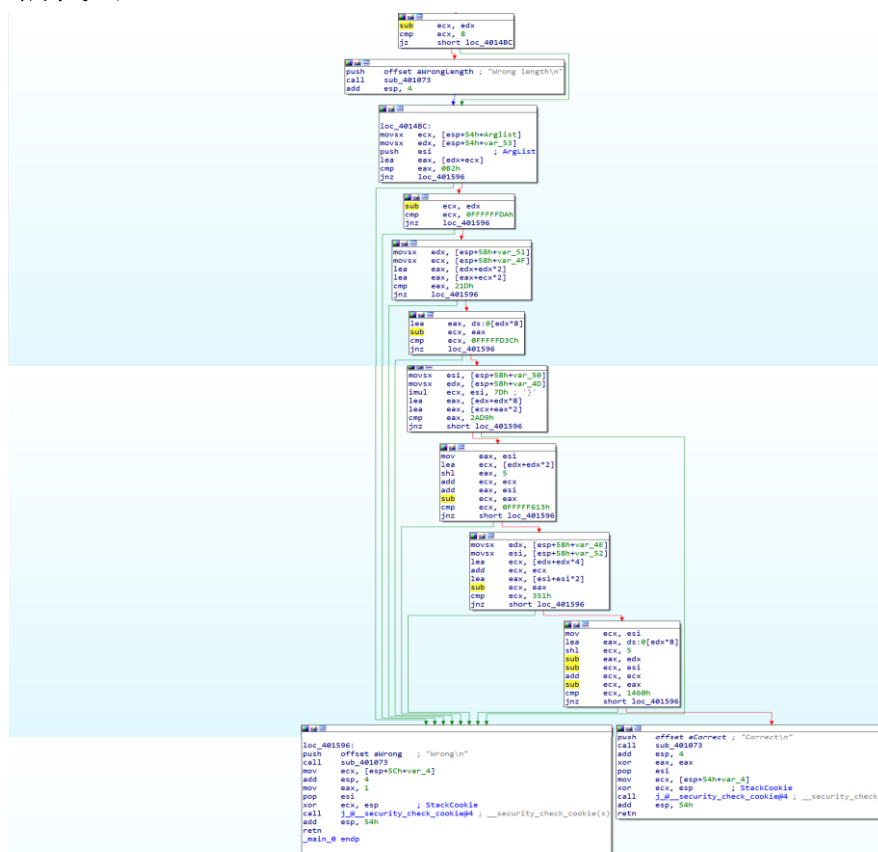
- 1、实验环境：反汇编工具 IDA Freeware
- 2、实验原理：

通过 IDA 得到二进制代码的反汇编代码,利用汇编所学知识对反汇编代码的数学计算、数据结构、条件判断、分支结构进行识别与分析。

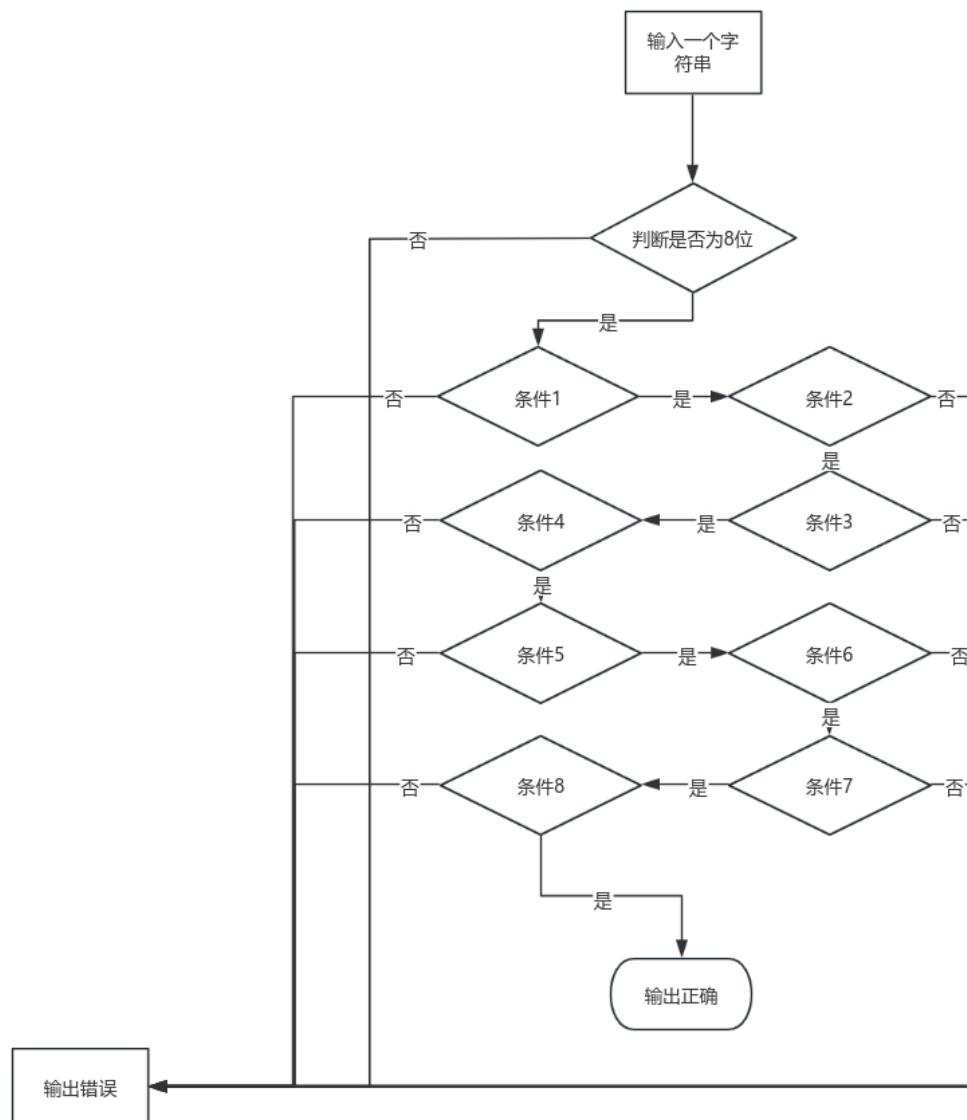
### 三、实验过程

### (一) task1

- 1、使用 IDA Freeware 打开 task1.exe 文件，查看其二进制代码的反汇编代码。



2、对代码进行分析，作出流程图。



总体思路：字符串每一个字符的 ascII 码满足上面的条件即可

3、对代码的详细分析。

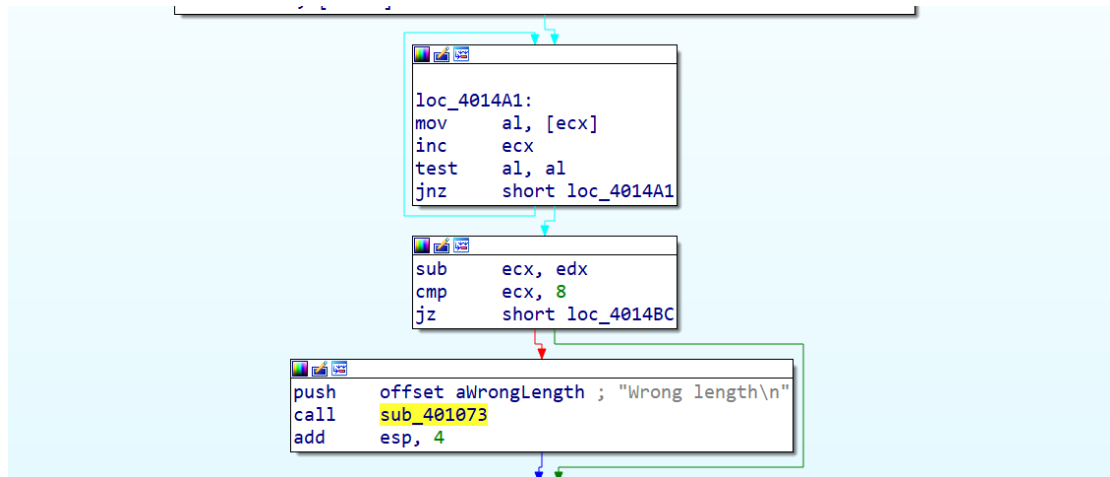
(1) 读取并存储字符串

```

sub     esp, 54h
mov     eax, ___security_cookie
xor     eax, esp
mov     [esp+54h+var_4], eax
push    offset aPleaseInputAS; "Please input a string:\n"
call    sub_401073
lea     eax, [esp+58h+Arglist]
push    eax; Arglist
push    offset Format; "%s"
call    sub_40101E
lea     ecx, [esp+60h+Arglist]
add     esp, 0Ch
lea     edx, [ecx+1]

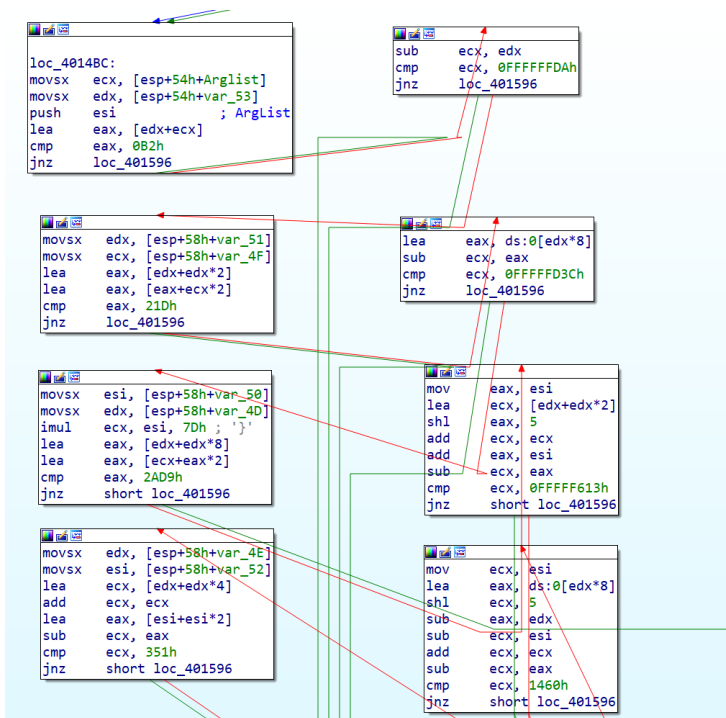
```

## (2) 判断字符串的长度是否为 8



如果为 8，则继续执行，进行下一步判断，如果不是 8，则输出长度错误、错误进而终止程序。

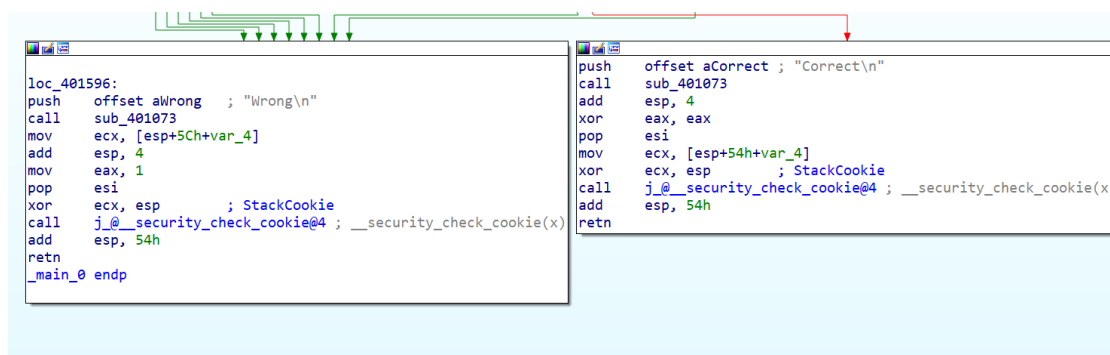
## (3) 判断是否满足下列一系列的方程



第一排:  $s_2 + s_1 = 178$     $s_1 - s_2 = -38$

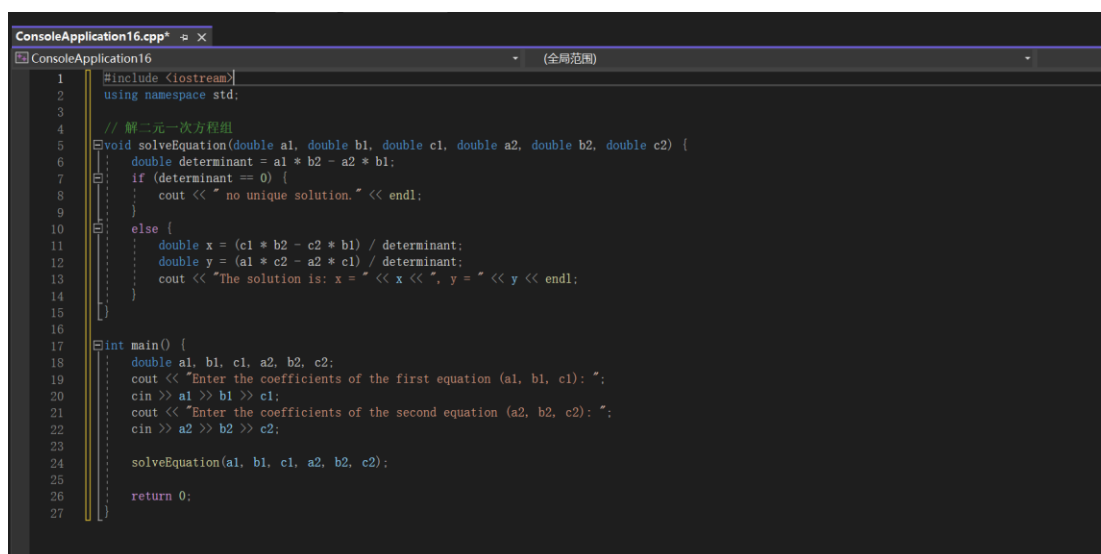
第二排:  $3s_4 + 2s_6 = 541$      $s_6 - 8s_4 = -708$   
 第三排:  $125s_5 + 18s_8 = 10969$      $6s_8 - 33s_5 = -2541$   
 第四排:  $10s_7 - 3s_3 = 849$      $62s_3 - 7s_7 = 5216$   
 具体分析略去

#### (4) 输出正误



#### 4、计算方式（利用 C++ 语言实现）。

主要是方程组的求解，字符串中每一个字符的 ASCII 码都由二元一次方程组确定，故写一个程序用来解二元一次方程组：

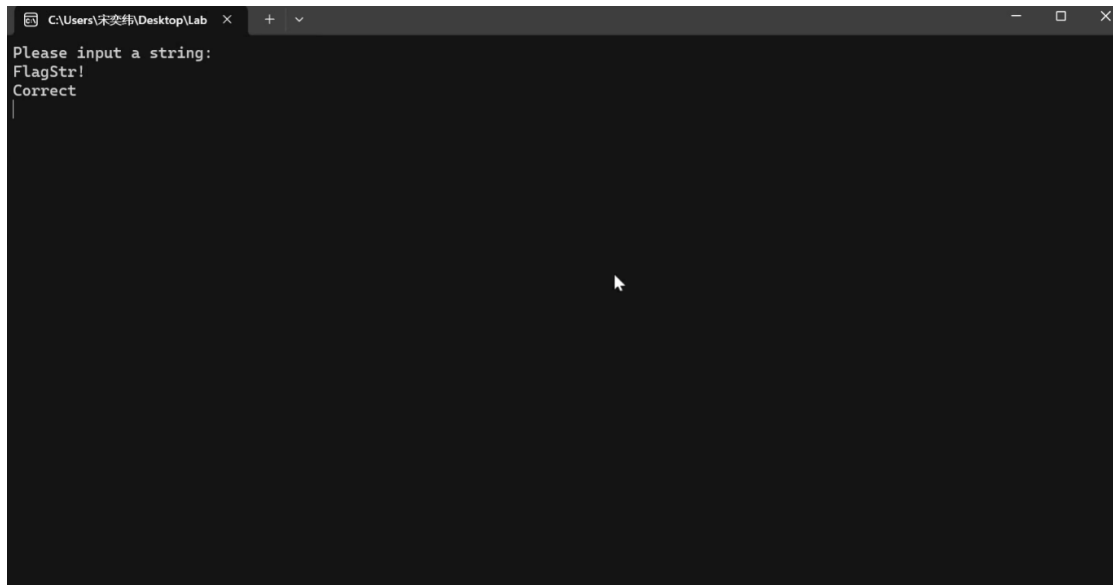


将参数带入，求解出各个字符的 ASCII 码（70 108 97 103 83 116 114 33），并转化为对应的字符：

FlagStr!

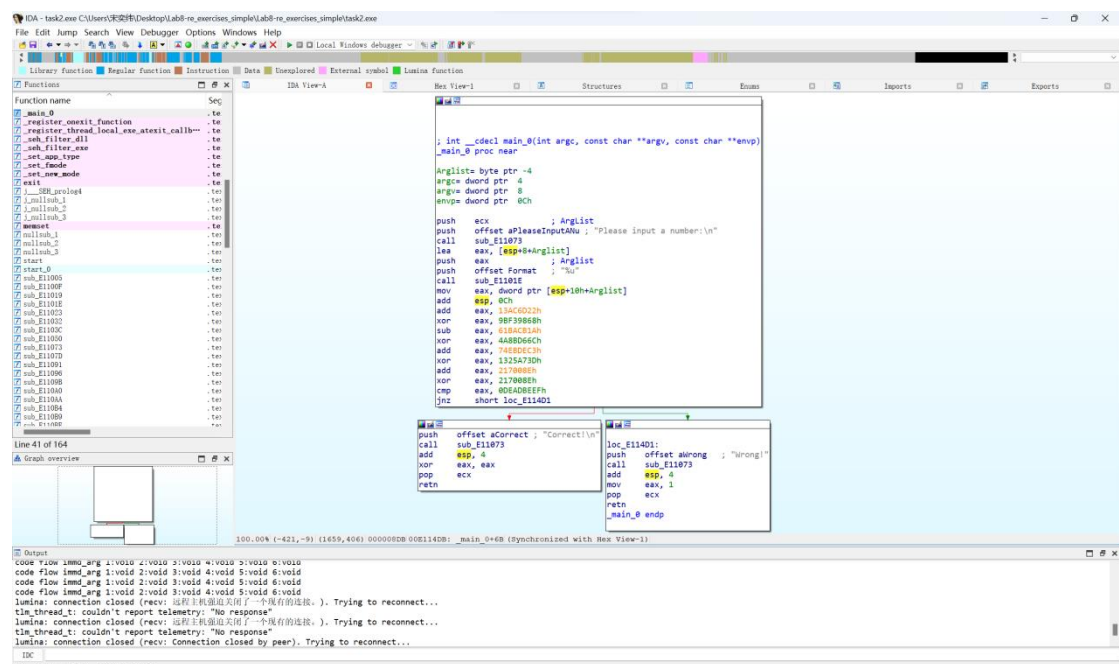
#### 5、测试运行。

将字符串“FlagStr!”输入程序中，得到正确的结果。

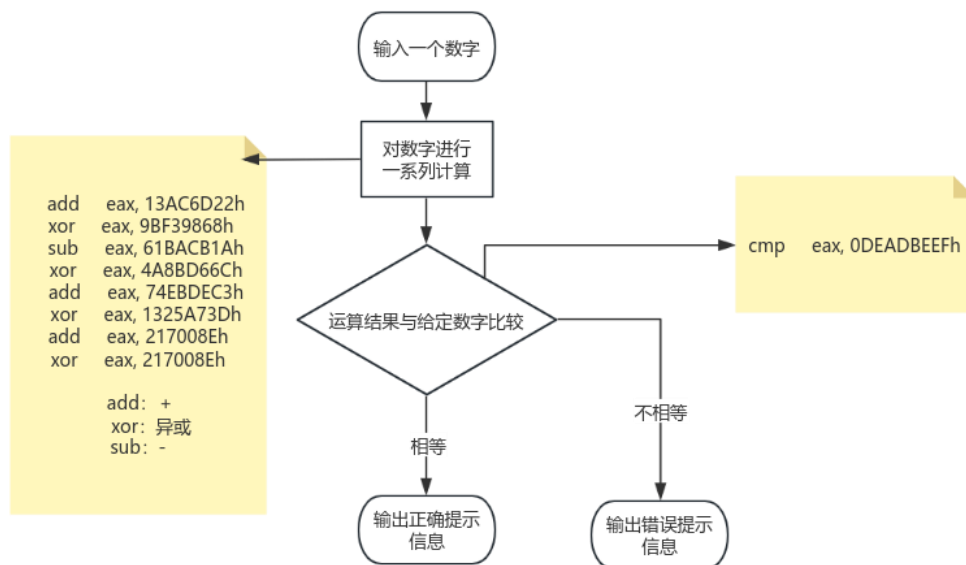


## (二) task2

- 1、使用 IDA Freeware 打开 task1.exe 文件，查看其二进制代码的反汇编代码。



- 2、对代码进行分析，作出流程图。



### 3、对代码的详细分析。

#### (1) 读入一个数字

```

push    ecx                ; ArgList
push    offset aPleaseInputANu ; "Please input a number:\n"
call    sub_E11073
lea     eax, [esp+8+Arglist]
push    eax                ; Arglist
push    offset Format        ; "%u"
call    sub_E1101E
mov     eax, dword ptr [esp+10h+Arglist]
add     esp, 0Ch
  
```

#### (2) 对数字进行运算

```

add     eax, 13AC6D22h
xor     eax, 9BF39868h
sub     eax, 61BACB1Ah
xor     eax, 4A8BD66Ch
add     eax, 74EBDEC3h
xor     eax, 1325A73Dh
add     eax, 217008Eh
xor     eax, 217008Eh
  
```

可以理解为以下的式子：

$$x = (((((((x + 0x13AC6D22) \wedge 0x9BF39868) - 0x61BACB1A) \wedge 0x4A8BD66C) + 0x74EBDEC3) \wedge 0x1325A73D) + 0x217008E) \wedge 0x217008E$$

#### (3) 进行比较：

```

cmp     eax, 0DEADBEEFh
jnz     short loc_E114D1
  
```

与 0DEADBEEFh 比较, 如果不相等跳转到 loc\_E114D1, 输出错误, 相等即继续执行, 输出正确。

(4) 输出正误:

A) 输出错误

```
loc_E114D1:
push     offset aWrong    ; "Wrong!"
call     sub_E11073
add      esp, 4
mov      eax, 1
pop      ecx
retn
_main_0 endp
```

B) 输出正确

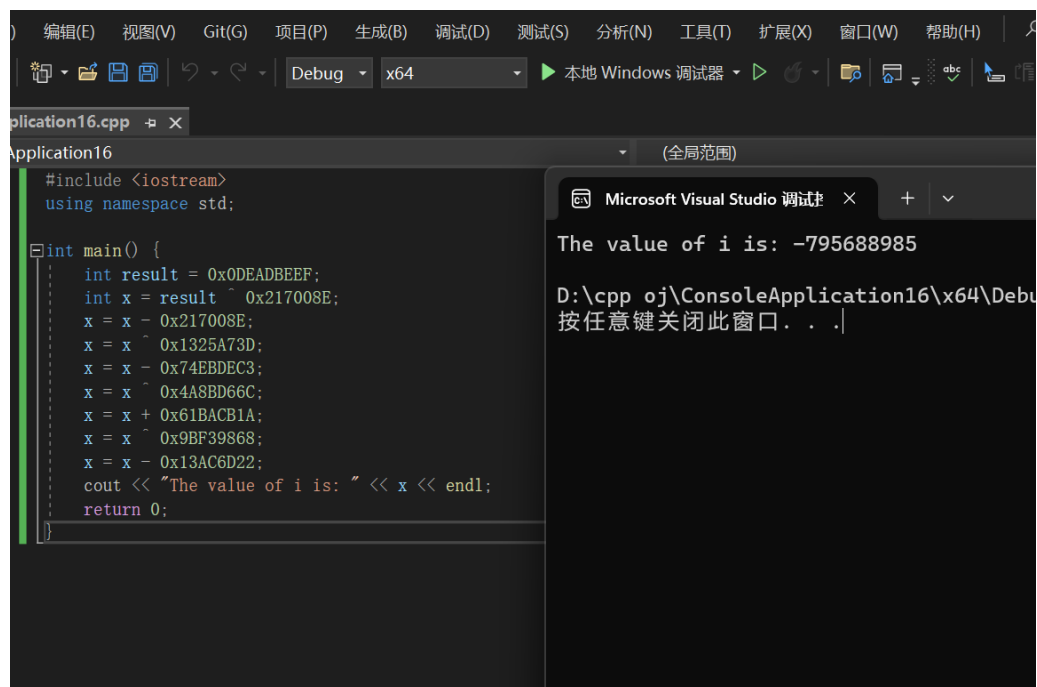
```
push     offset aCorrect ; "Correct!\n"
call     sub_E11073
add      esp, 4
xor      eax, eax
pop      ecx
retn
```

#### 4、计算方式（利用 C++语言实现）。

使用两种方法进行计算:

##### (1) 逆推法

此处利用的原理为  $a \oplus b \oplus b = a$ , 故异或运算的逆运算为异或;  
又有加与减互为逆运算。



The screenshot shows the Microsoft Visual Studio IDE. The left pane displays a C++ source file named 'Application16.cpp'. The code defines a function 'main' that calculates a value 'x' through a series of XOR and subtraction operations. The right pane shows the 'Debug Console' output, which displays the result of the calculation: 'The value of i is: -795688985'.

```
#include <iostream>
using namespace std;

int main() {
    int result = 0xDEADBEEF;
    int x = result ^ 0x217008E;
    x = x - 0x217008E;
    x = x ^ 0x1325A73D;
    x = x - 0x74EBDEC3;
    x = x ^ 0x4A8BD66C;
    x = x + 0x61BACB1A;
    x = x ^ 0x9BF39868;
    x = x - 0x13AC6D22;
    cout << "The value of i is: " << x << endl;
    return 0;
}
```

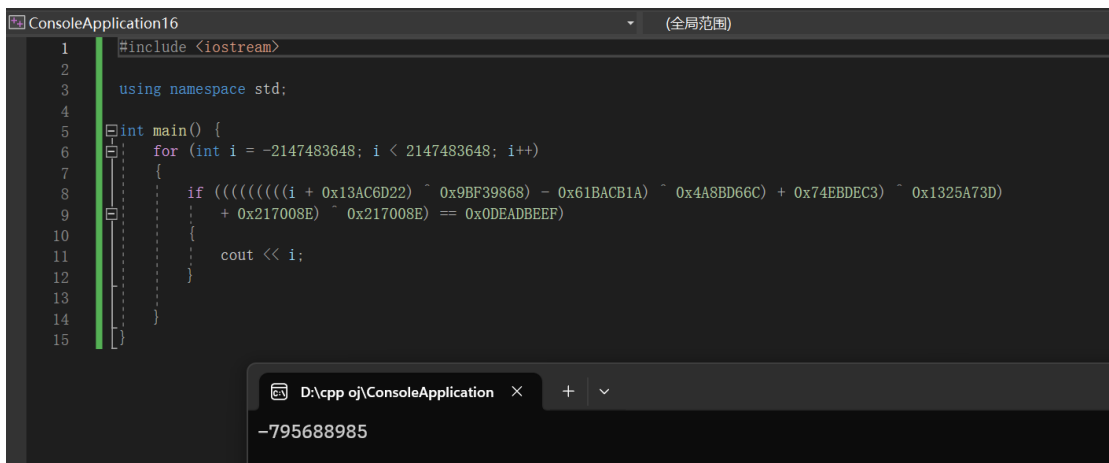
The value of i is: -795688985

D:\cpp oj\ConsoleApplication16\x64\Debug  
按任意键关闭此窗口. . . |



## (2) 遍历法

将原始方程输入，遍历所有的 int 型数，得到解的值。



```
1 #include <iostream>
2
3 using namespace std;
4
5 int main() {
6     for (int i = -2147483648; i < 2147483648; i++)
7     {
8         if (((((((((i + 0x13AC6D22) ^ 0x9BF39868) - 0x61BACB1A) ^ 0x4A8BD66C) + 0x74EBDEC3) ^ 0x1325A73D)
9             + 0x217008E) ^ 0x217008E) == 0xDEADBEEF)
10        {
11            cout << i;
12        }
13    }
14 }
15
```

D:\cpp oj\ConsoleApplication 16

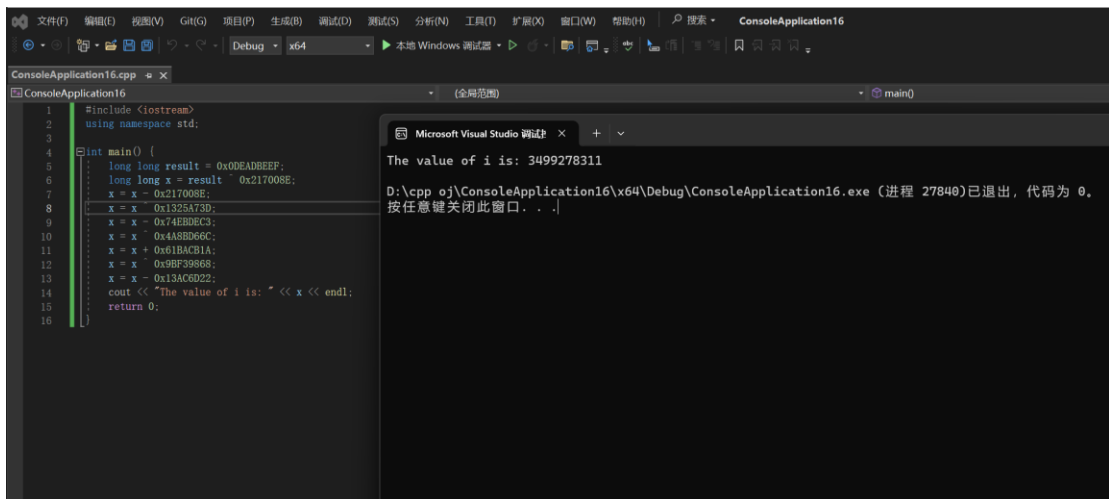
-795688985

故我得到了方程的解为： **-795688985**

但是在整形之外是否有解，发现确实是有的： **3499278311**

(该解超出了 int 的范围，故本人考虑将 int 改成 long long)

(遍历法效率太低，故只采用逆向推理计算，得到 int 外的解)



```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     long long result = 0xDEADBEEF;
6     long long x = result ^ 0x217008E;
7     x = x ^ 0x217008E;
8     x = x ^ 0x1325A73D;
9     x = x ^ 0x74EBDEC3;
10    x = x ^ 0x4A8BD66C;
11    x = x + 0x61BACB1A;
12    x = x ^ 0x9BF39868;
13    x = x ^ 0x13AC6D22;
14    cout << "The value of i is: " << x << endl;
15    return 0;
16 }
```

Microsoft Visual Studio 调试

The value of i is: 3499278311

D:\cpp oj\ConsoleApplication16\x64\Debug\ConsoleApplication16.exe (进程 27840) 已退出，代码为 0。  
按任意键关闭此窗口。 . . |

## 5、测试运行。

将两个解输入，均得到正确答案。

```
C:\Users\宋奕伟\Desktop\Lab x + v - □ x
Please input a number:
-795688985
Correct!
|
```

```
C:\Users\宋奕伟\Desktop\Lab x + v - □ x
Please input a number:
3499278311
Correct!
|
```

## 四、实验结论及心得体会

- 1、熟悉静态反汇编工具 IDA Freeware。
- 2、熟悉反汇编代码的逆向分析过程。
- 3、掌握反汇编语言中的数学计算、数据结构、条件判断、分支结构的识别和逆向分析。
- 4、对汇编语言有了更深入的了解。
- 5、提升了解决问题的能力，学会多种编程语言综合使用。