

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
学 号 2211044
姓 名 陆皓喆
班 级 信息安全

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

(一) task1

1.通过 IDA Freeware 得到 task1.exe 的反汇编代码，如图 1 和图 2 所示。

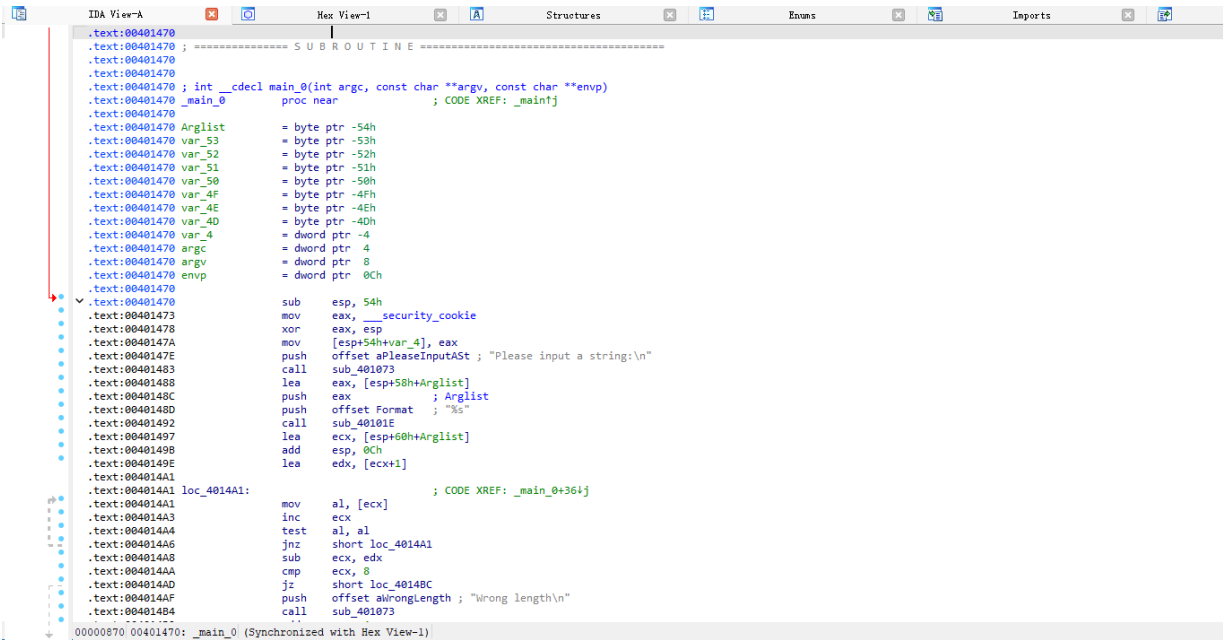


图 1 task1.exe 的反汇编代码

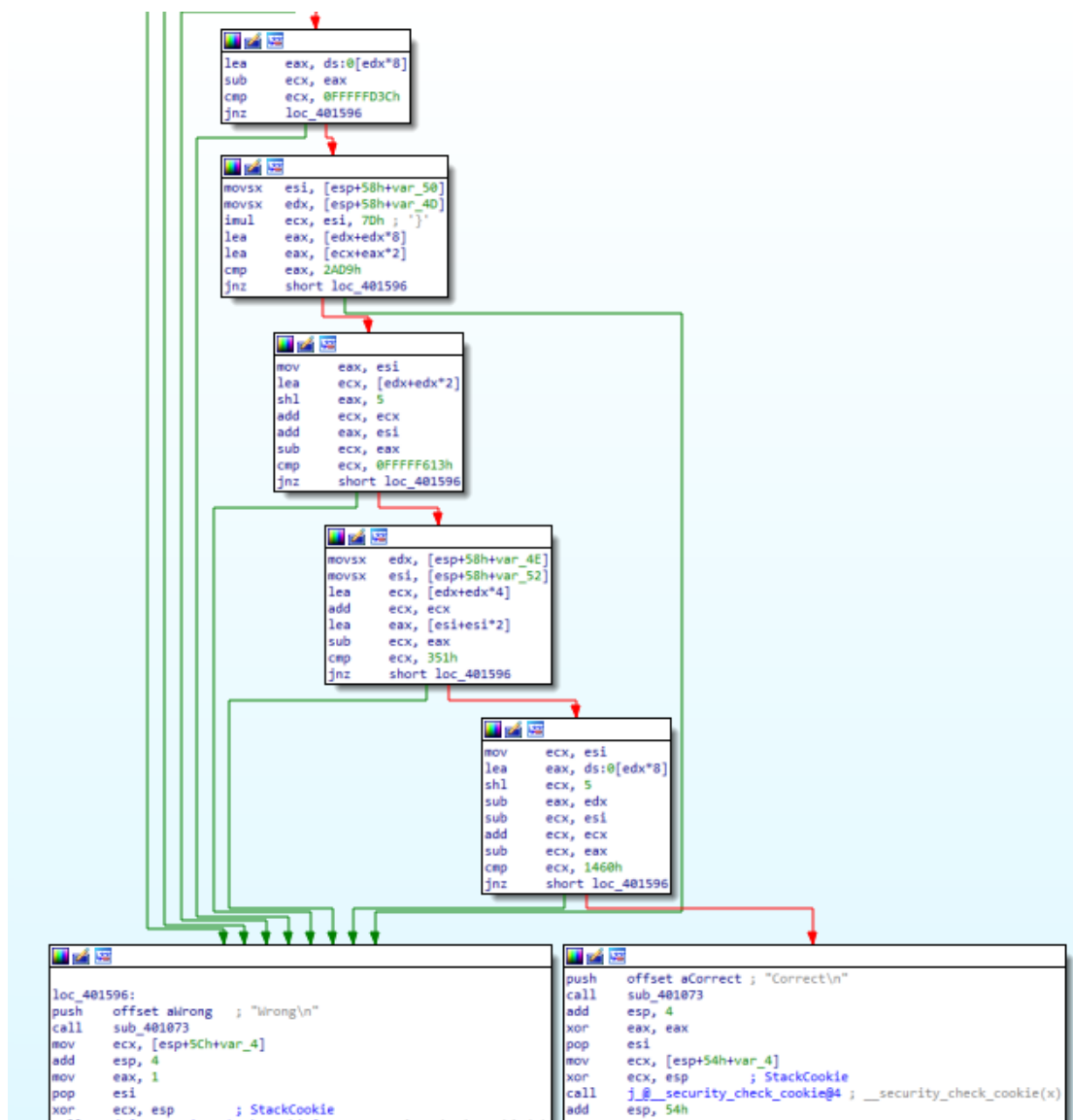


图 2 task1.exe 反汇编代码的图形化显示

2. 对反汇编代码和计算过程、条件判断、分支结构等信息进行分析，逆向推出程序的正确输入数据，完成逆向分析挑战。

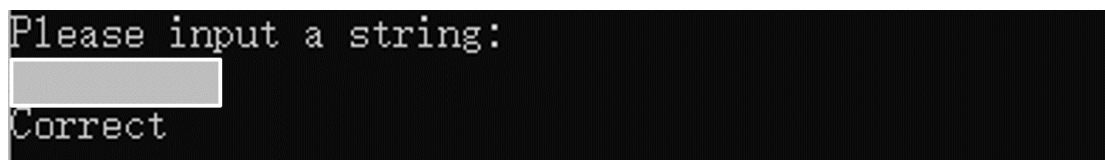


图 3 逆向分析，完成 task1 练习

(二) task2

1. 通过 IDA Freeware 得到 task2.exe 的反汇编代码，如图 4 和图 5 所示。

```

.text:00401470 ; ===== SUBROUTINE =====
.text:00401470
.text:00401470
.text:00401470 ; int __cdecl main_0(int argc, const char **argv, const char **envp)
.text:00401470 _main_0 proc near ; CODE XREF: _main!j
.text:00401470
.text:00401470 Arglist = byte ptr -4
.text:00401470 argc = dword ptr 4
.text:00401470 argv = dword ptr 8
.text:00401470 envp = dword ptr 0Ch
.text:00401470
.text:00401470 push ecx ; Arglist
.text:00401471 push offset aPleaseInputANu ; "Please input a number:\n"
.text:00401476 call sub_401073
.text:0040147B lea eax, [esp+8+Arglist]
.text:0040147F push eax ; Arglist
.text:00401480 push offset Format ; "%u"
.text:00401485 call sub_40101E
.text:0040148A mov eax, dword ptr [esp+10h+Arglist]
.text:0040148E add esp, 0Ch
.text:00401491 add eax, 13AC6D22h
.text:00401496 xor eax, 9BF39868h
.text:0040149B sub eax, 618ACB1Ah
.text:004014A0 xor eax, 4A8BD66Ch
.text:004014A5 add eax, 74EBDEC3h
.text:004014AA xor eax, 1325A73Dh
.text:004014AF add eax, 217008Eh
.text:004014B4 xor eax, 217008Eh
.text:004014B9 cmp eax, 0DEADBEEFh
.text:004014BE jnz short loc_4014D1
.text:004014C0 push offset aCorrect ; "Correct!\n"
.text:004014C5 call sub_401073
.text:004014CD add esp, 4
.text:004014CF xor eax, eax
.text:004014D0 pop ecx
.text:004014D0 ret
.text:004014D1 ; -----
.text:004014D1
.text:004014D1 loc_4014D1:
.text:004014D1 push offset aWrong ; "Wrong!"
.text:004014D6 call sub_401073
.text:004014DB add esp, 4
.text:004014DE mov eax, 1
.text:004014E3 pop ecx

```

图 4 task2.exe 的反汇编代码



图 5 task2.exe 反汇编代码的图形化显示

2.对反汇编代码的计算过程、条件判断、分支结构等信息进行分析，逆向推出程序的正确输入数据，完成逆向分析挑战。

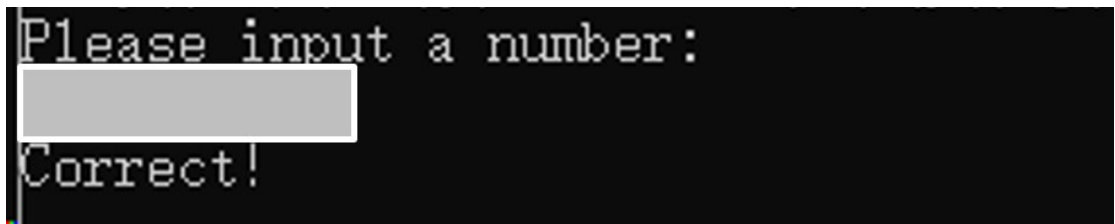


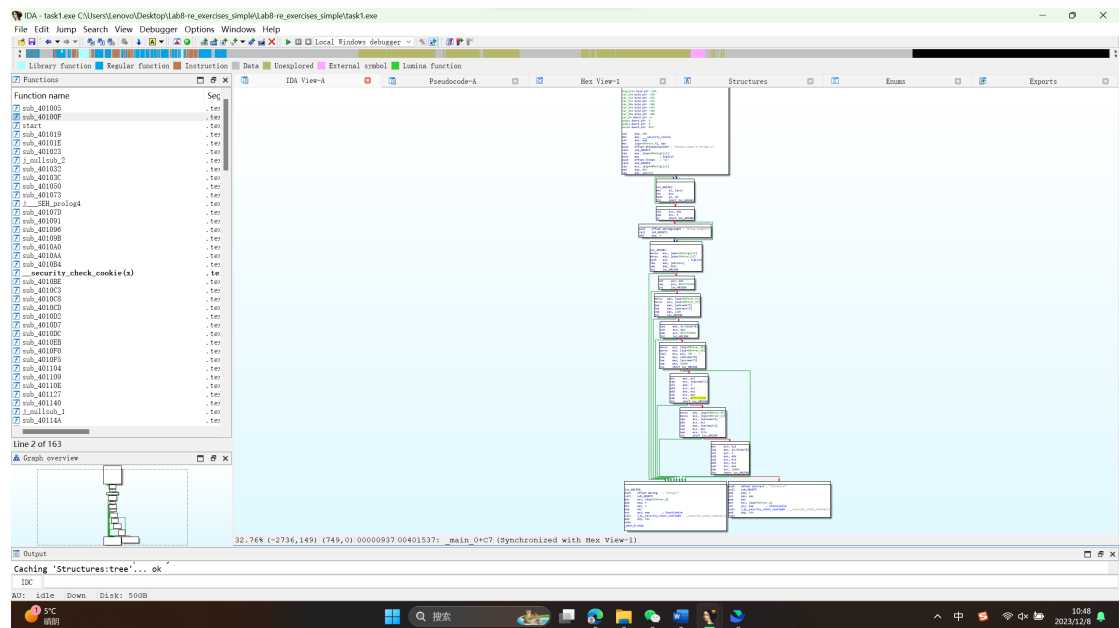
图 6 逆向分析，完成 task2 练习

三、实验报告

3.1 分别针对 task1、task2 使用 IDA Freeware，获得二进制代码的反汇编代码，提供截图。

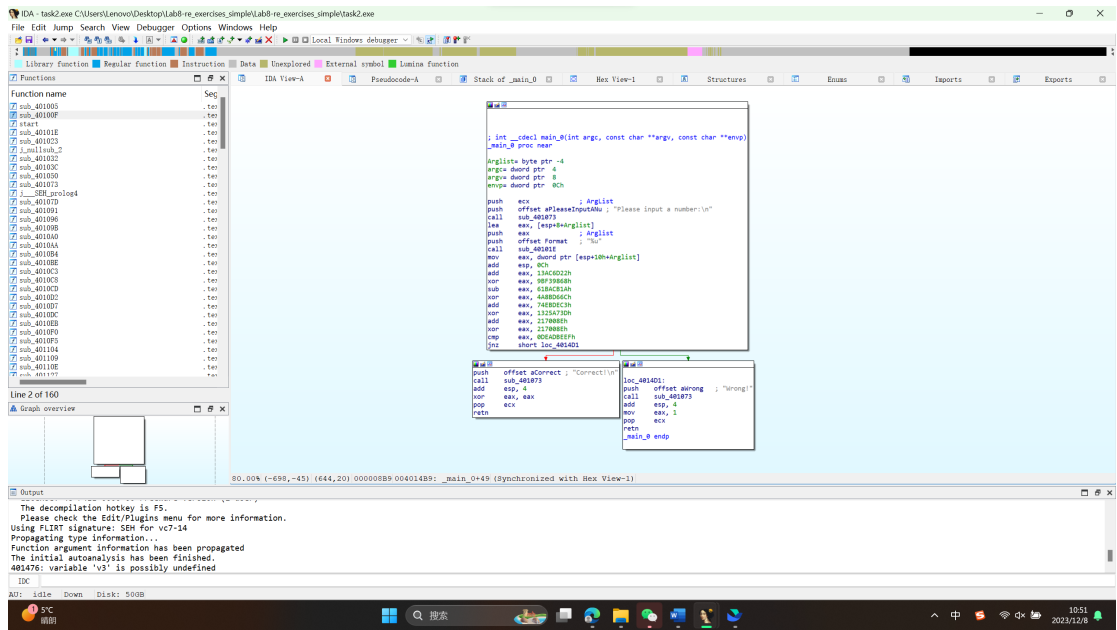
3.1.1 task1 部分截图

如图所示，将 task1 文件在 IDA 中打开，即可获得如下的反汇编代码块。



3.1.2 task2 部分截图

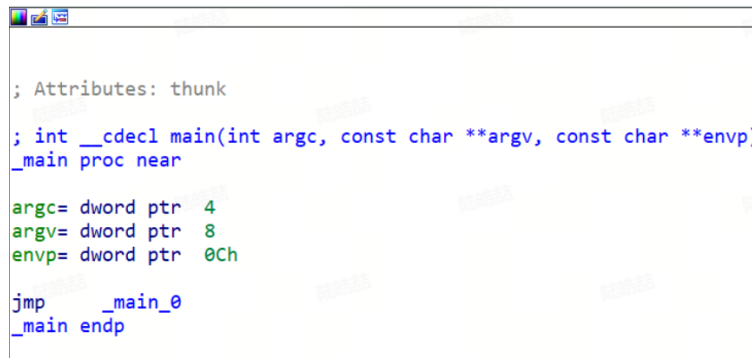
同理，将 task2 也拖动到 IDA 中打开，即可获得如下的反汇编代码块。



3.2 分别针对 task1、task2 反汇编代码的计算过程、数据结构、条件判断、分支结构等信息进行逆向分析，在实验报告中记录逆向分析的详细过程，并画出程序流程图。

3.2.1 task1 部分分析

首先进入 IDA 中，我们发现该程序首先接收了程序传入的 argc（外部参数个数）、argv（外部参数）与 envp（环境变量），然后跳转到 main 函数当中去。



首先，主函数是一个监听字符串输入的函数 sub_40101E 函数，双击进入函数，发现进入到 sub_401650 中。该程序的功能应该是将一个字符串输入到 Arglist 中去。

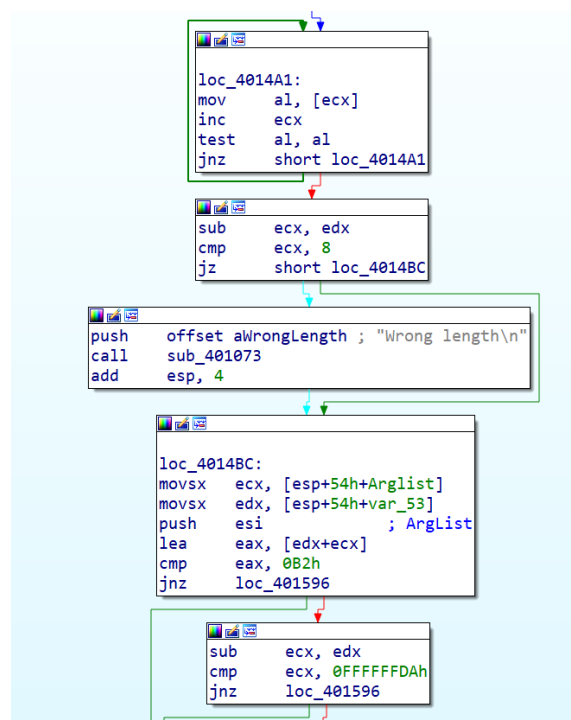
```

; Attributes: thunk
; int __cdecl sub_40101E(char *Format, char Arglist)
sub_40101E proc near
    Format= dword ptr 4
    Arglist= byte ptr 8
    jmp     sub_401650
sub_40101E endp

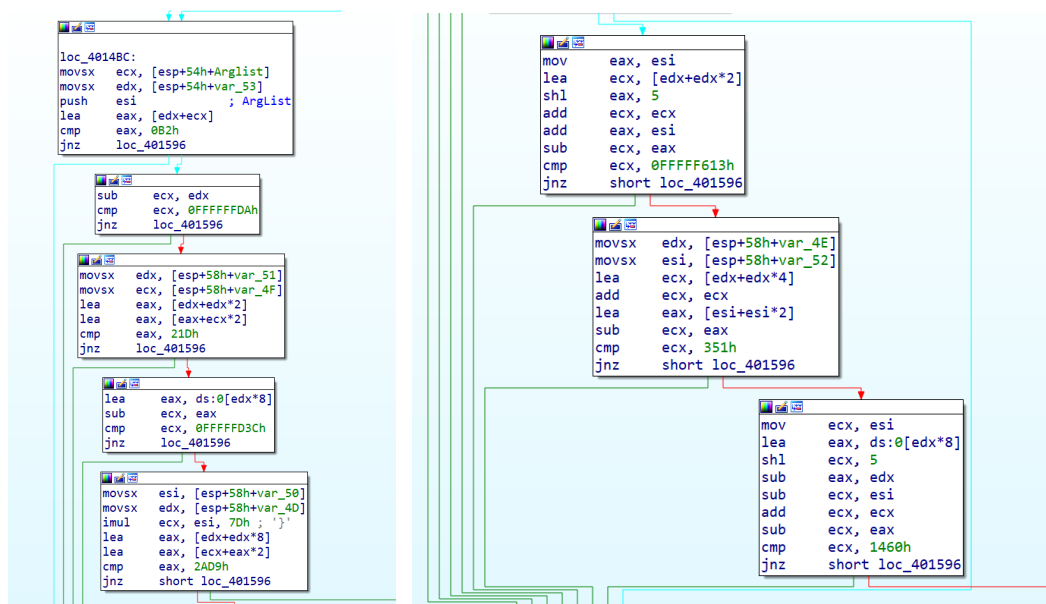
; int __cdecl main_0(int argc, const char **argv, const char **envp)
_main_0 proc near
    Arglist= byte ptr -54h
    var_52= byte ptr -52h
    var_51= byte ptr -51h
    var_50= byte ptr -50h
    var_4f= byte ptr -4fh
    var_4e= byte ptr -4eh
    var_4d= byte ptr -4dh
    var_4c= dword ptr -4
    argc= dword ptr 4
    argv= dword ptr 8
    envp= dword ptr 0Ch
    sub     esp, 54h
    mov     eax, ___security_cookie
    xor     eax, esp
    mov     [esp+54h+var_4], eax
    offset 0PleaseInputString ; "Please input a string:\n"
    call    sub_401073
    lea     eax, [esp+50h+Arglist]
    push    offset Format ; "s"
    call    sub_40101E
    lea     ecx, [esp+60h+Arglist]
    add     esp, 0Ch
    lea     edx, [ecx+1]
    ; int sub_401650(char *Format, ...)
sub_401650 proc near
    Format= dword ptr 4
    Arglist= byte ptr 8
    push    esi
    mov     esi, [esp+4+Format]
    push    0 ; ix
    call    ds:___imp___acrt_iow_func
    add     esp, 4
    lea     ecx, [esp+4+Arglist]
    push    ecx ; Arglist
    push    0 ; Locale
    push    esi ; Format
    push    eax ; Stream
    call    sub_4011CC
    push    dword ptr [eax+4] ; Options
    push    dword ptr [eax] ; Format
    call    ds:___imp___stdio_common_vfprintf
    add     esp, 10h
    pop     esi
    retn
sub_401650 endp

```

之后进入到一个循环中，功能应该是将字符串的长度与 8 进行比较，如果不相等就输出“Wrong length”，其中 sub_401073 为输出函数。



之后就进入到了该程序的核心部分，一共是八个步骤，需要分别进行判定字符串的部分是否符合要求。



之后进入到一个较长的 if 判断过程，首先：

第一步是将第一个字符存入 `ecx` 中，将第二个字符存入 `edx` 中，将两个数相加判断其是否与 `0B2h`（即 178）相等，如果不相等直接跳转到最后的错误输出中，如果相等则继续判断；

第二步是将上面两个数相减判断其是否等于 `0FFFFFFDAh`（即-38），相等则继续判断；

第三步是将第四个字符放入 `edx` 中，第六个字符放入 `ecx` 中，判断第四个字符*3+第六个字符*2 和是否与 `21Dh`（即 541）相等，相等则继续；

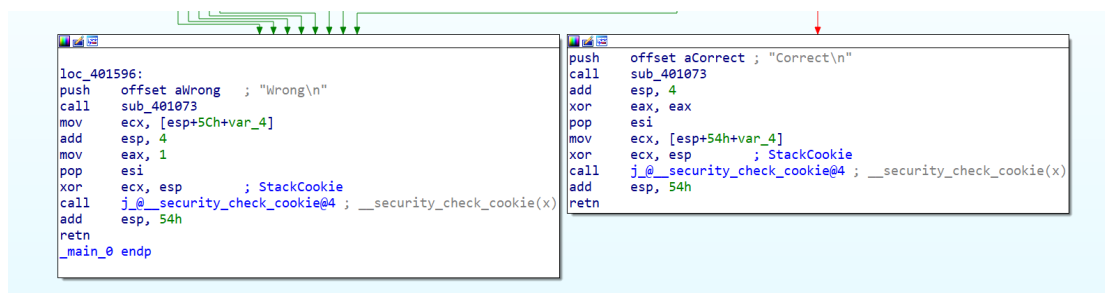
第四步判断 `ecx-edx*8`，即第六个字符-第四个字符的八倍是否与 `0FFFFFFD3Ch`（即-708）相等；

第五步是将第五个字符存入 `esi` 中，第八个字符存入 `edx` 中，首先将 `esi*7Dh`（即 125）放入 `ecx` 中，再进行 `18*edx+ecx` 的操作，化简后即判断 `125*第五个数+18*第八个数` 是否与 `2AD9h`（即 10969）相等；

第六步是判断 `6*第八个数-33*第五个数` 是否与 `0FFFFFF613h`（即-2541）相等；

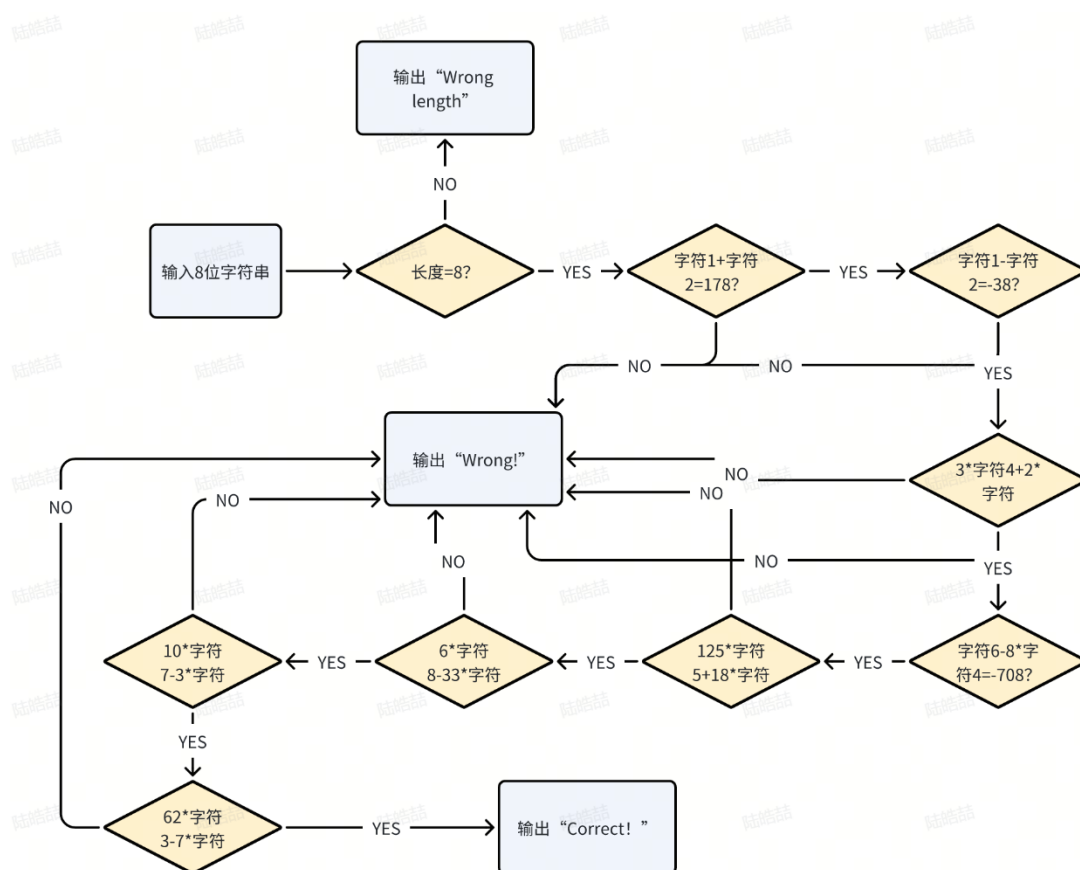
第七步是将第七个数存入 `edx` 中，第三个数存入 `esi` 中，判断 `10*第七个数-3*第三个数` 是否与 `351h`（即 849）相等；

第八步是判断 $62 \times \text{第三个数} - 7 \times \text{第七个数}$ 是否与 $1460h$ （即 5216 ）相等，如果以上判断均成立则跳转到成功的输出。



结束阶段，只要上述的八个条件有一个没有满足，那么就会输出“Wrong”，反之输出“Correct！”

根据上述的描述，我们画出该程序的流程图：



我们在分析完具体思路之后，按下“F5”进行检验，发现逆向程序的基本思路是与我们相同的。

```
IDA View-A Pseudocode-A Hex View-1
1 int __cdecl main_0(int argc, const char **argv, const char **envp)
2 {
3     char v3; // si
4     char Arglist; // [esp+0h] [ebp-54h] BYREF
5     char v6; // [esp+1h] [ebp-53h]
6     char v7; // [esp+2h] [ebp-52h]
7     char v8; // [esp+3h] [ebp-51h]
8     char v9; // [esp+4h] [ebp-50h]
9     char v10; // [esp+5h] [ebp-4Fh]
10    char v11; // [esp+6h] [ebp-4Eh]
11    char v12; // [esp+7h] [ebp-4Dh]
12
13    sub_401073("Please input a string:\n", Arglist);
14    sub_40101E("%s", (char)&Arglist);
15    if ( strlen(&Arglist) != 8 )
16        sub_401073("Wrong length\n", Arglist);
17    if ( v6 + Arglist == 178
18        && Arglist - v6 == -38
19        && 3 * v8 + 2 * v10 == 541
20        && v10 - 8 * v8 == -708
21        && 125 * v9 + 18 * v12 == 10969
22        && 6 * v12 - 33 * v9 == -2541
23        && 10 * v11 - 3 * v7 == 849
24        && 62 * v7 - 7 * v11 == 5216 )
25    {
26        sub_401073("Correct\n", v3);
27        return 0;
28    }
29    else
30    {
31        sub_401073("Wrong\n", v3);
32        return 1;
33    }
34 }
```

我们完成了对 task1 的反向分析，现在我们需要编写一个 C++ 程序来进行对输入的解析。

C++ 代码如下所示：

```
8-1.cpp* = X
1 #include<iostream>
2 #include<string>
3 using namespace std;
4 int main() {
5     int a1; int a2; int a3; int a4; int a5; int a6; int a7; int a8;
6     a1 = (178 - 38) / 2;
7     a2 = 178 - a1;
8     a3 = (7 * 849 + 10 * 5216) / (62 * 10 - 3 * 7);
9     a4 = (541 - (-708) * 2) / (3 + 8 * 2);
10    a5 = (10969 - (-2541) * 3) / (125 + 33 * 3);
11    a6 = 8 * a4 - 708;
12    a7 = (849 + 3 * a3) / 10;
13    a8 = (33 * a5 - 2541) / 6;
14    cout << char(a1) << char(a2) << char(a3) << char(a4) << char(a5) << char(a6) << char(a7) << char(a8) << endl;
15
16
17
18    return 0;
19 }
20
21
```

```
Microsoft Visual Studio 调试
FlagStr!
C:\Users\Lenovo\Desktop\C++\x64\Debug\?.exe (?? 28300)???,?? 0?
?????????. . .|
```

如图所示，运行结果是 FlagStr!

3.2.2 task2 部分分析

```
; int __cdecl main_0(int argc, const char **argv, const char **envp)
_main_0 proc near

    Arglist= byte ptr -4
    argc= dword ptr 4
    argv= dword ptr 8
    envp= dword ptr 0Ch

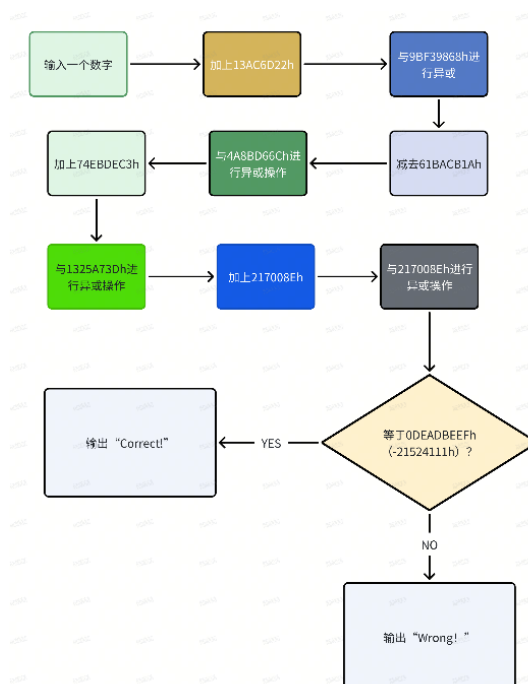
    push    ecx                ; Arglist
    push    offset aPleaseInputANu ; "Please input a number:\n"
    call    sub_401073
    lea     eax, [esp+8+Arglist]
    push    eax                ; Arglist
    push    offset Format        ; "%u"
    call    sub_40101E
    mov     eax, dword ptr [esp+10h+Arglist]
    add     esp, 0Ch
    add     eax, 13AC6D22h
    xor     eax, 9BF39868h
    sub     eax, 61BACB1Ah
    xor     eax, 4A8BD66Ch
    add     eax, 74EBDEC3h
    xor     eax, 1325A73Dh
    add     eax, 217008Eh
    xor     eax, 217008Eh
    cmp     eax, 0DEADBEEFh
    jnz     short loc_4014D1
```

从上面的反汇编代码中，我们对其进行分析。


首先程序进入主程序，先输出一个 “Please input a number!”

然后我们输入一个数字，将其赋值到寄存器 `eax` 中。然后我们开始计算，先将 `eax` 的值加上 `13AC6D22h`，然后再与 `9BF39868h` 进行异或操作，然后减去 `61BACB1Ah`，再与 `4A8BD66Ch` 进行异或操作，接着加上 `74EBDEC3h`，接着与 `1325A73Dh` 进行异或操作，然后加上 `217008Eh`，再与 `217008Eh` 进行异或操作，最后与 `0DEADBEEFh` (`-21524111h`) 进行比较，如果相等的话就输出 “Correct! ”，反之输出 “Wrong! ”

我们进一步的画出流程图：



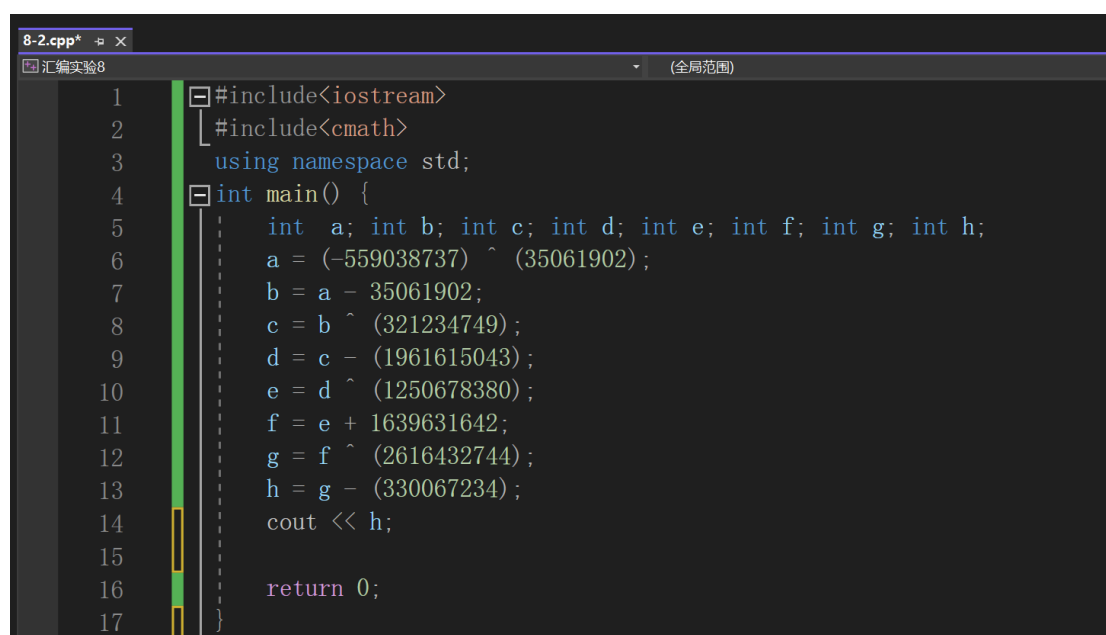
我们在分析完具体思路之后，按下“F5”进行检验，发现逆向程序的基本思路是与我们相同的。



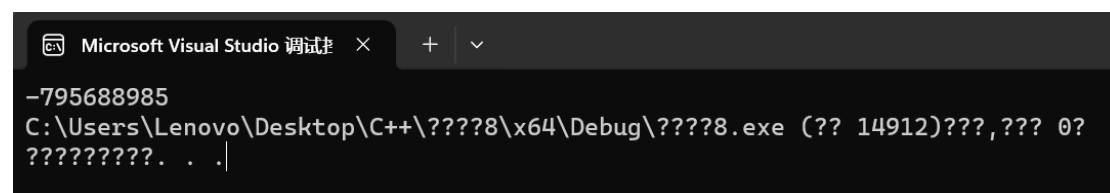
```
1 int __cdecl main_0(int argc, const char **argv, const char **envp)
2 {
3     char v3; // c1
4     char Arglist[4]; // [esp+0h] [ebp-4h] BYREF
5
6     sub_401073("Please input a number:\n", v3);
7     sub_40101E("%u", (char)Arglist);
8     if ( (((((((*(DWORD *)Arglist + 330067234) ^ 0x9BF39868) - 1639631642) ^ 0x4A8BD66C) + 1961615043) ^ 0x1325A73D)
9         + 35061902) ^ 0x217008E) == -559038737 )
10     {
11         sub_401073("Correct!\n", Arglist[0]);
12         return 0;
13     }
14     else
15     {
16         sub_401073("Wrong!", Arglist[0]);
17         return 1;
18     }
19 }
```

我们完成了对 task2 的反向分析，现在我们需要编写一个 C++ 程序来进行对输入的解析。

C++代码如下所示：



```
1 #include<iostream>
2 #include<cmath>
3 using namespace std;
4 int main() {
5     int a; int b; int c; int d; int e; int f; int g; int h;
6     a = (-559038737) ^ (35061902);
7     b = a - 35061902;
8     c = b ^ (321234749);
9     d = c - (1961615043);
10    e = d ^ (1250678380);
11    f = e + 1639631642;
12    g = f ^ (2616432744);
13    h = g - (330067234);
14    cout << h;
15
16    return 0;
17 }
```



```
Microsoft Visual Studio 调试
-795688985
C:\Users\Lenovo\Desktop\C++\????8\x64\Debug\????8.exe (?? 14912)???,??? 0?
????????? . . |
```

如图所示，运行结果是-795688985。

如果我们不考虑负数的情况时，我们也能够得到一个答案，我们经过计算可得，该值为 3499278311。

3.3 分别运行程序 task1、task2，根据提示输入逆向挑战的结果，获得“Correct”输出，提供截图。

对于 task1，我们输入 C++ 程序所解出来的结果，得到以下的结果：

```
C:\Users\Lenovo>D:\学学学\本科\大二上\汇编与逆向技术基础\2211044_陆皓喆_汇编实验\Lab8_逆向补充实验\task1.exe
Please input a string:
FlagStr!
Correct
C:\Users\Lenovo>
```

可以发现输出 “Correct”，实验成功！

对于 task2，我们输入 C++ 程序所解出来的结果“-795688985”与“3499278311”，得到以下的结果：

```
Microsoft Windows [Version 10.0.22621.2715]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Lenovo>D:\学学学\本科\大二上\汇编与逆向技术基础\2211044_陆皓喆_汇编实验\Lab8_逆向补充实验\task2.exe
Please input a number:
-795688985
Correct!
C:\Users\Lenovo>
```

```
C:\Users\Lenovo>D:\学学学\本科\大二上\汇编与逆向技术基础\2211044_陆皓喆_汇编实验\Lab8_逆向补充实验\task2.exe
Please input a number:
3499278311
Correct!
C:\Users\Lenovo>
```

可以发现输出“Correct！”，实验成功！