

**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；

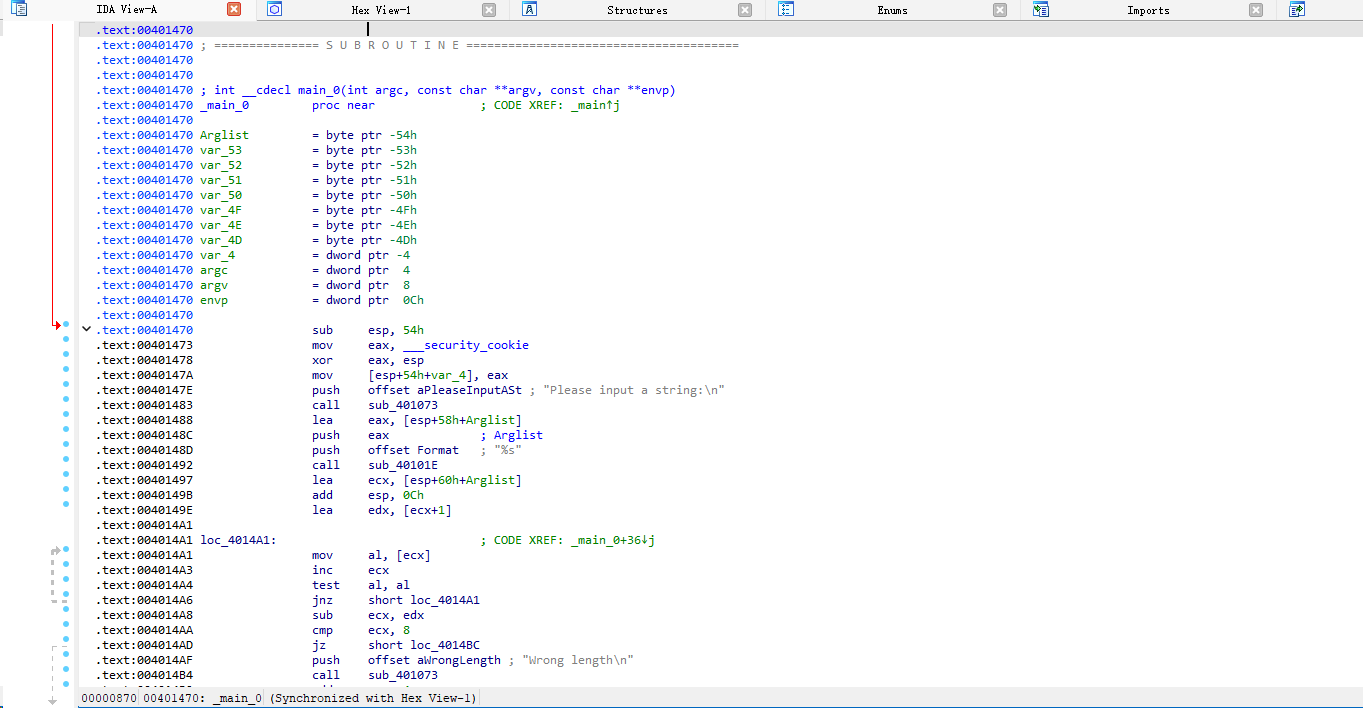
2.熟悉反汇编代码的逆向分析过程；

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。

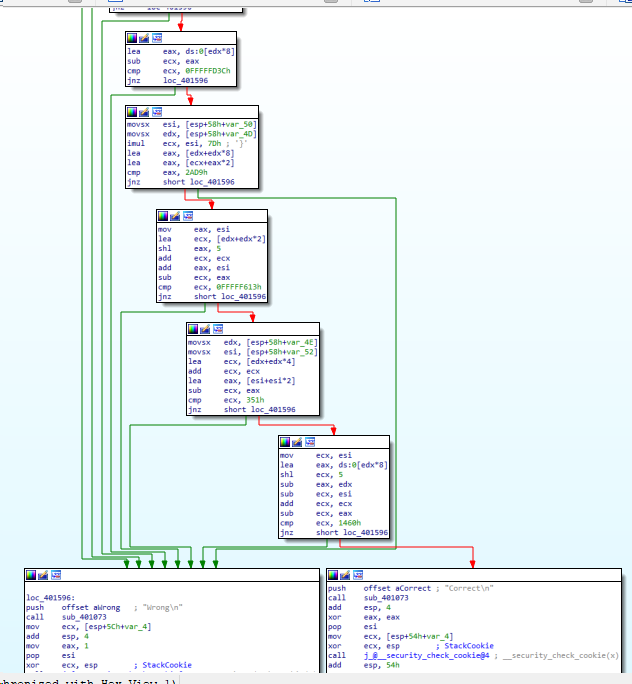
**二、实验原理**

1. **task1**

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

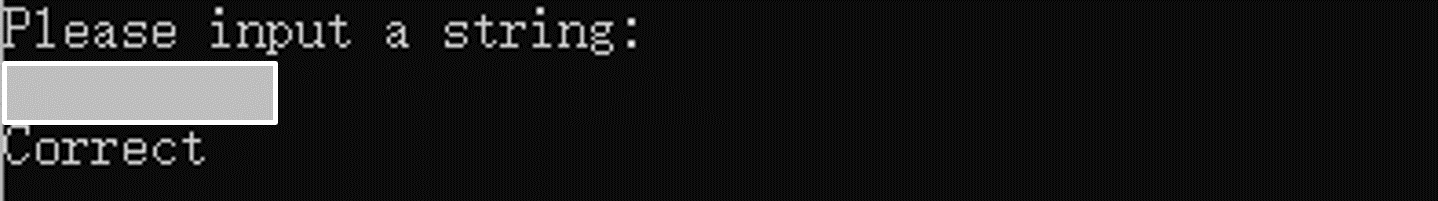


**图 1 task1.exe的反汇编代码**



**图 2 task1.exe反汇编代码的图形化显示**

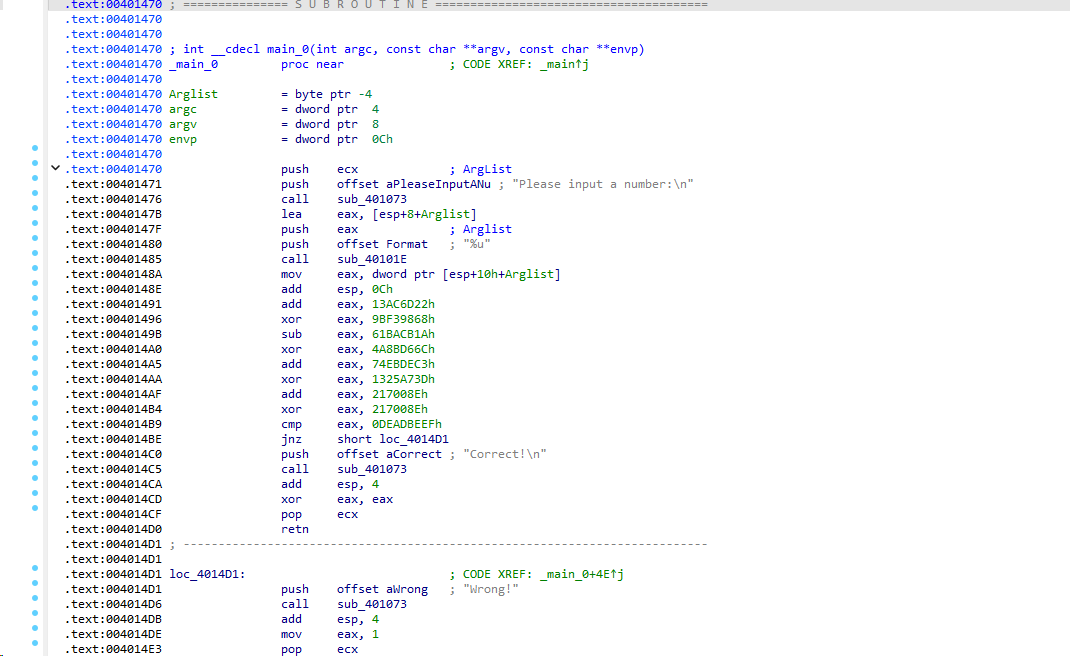
2.对反汇编代码和计算过程、条件判断、分支结构等信息进行分析，逆向推出程序的正确输入数据，完成逆向分析挑战。



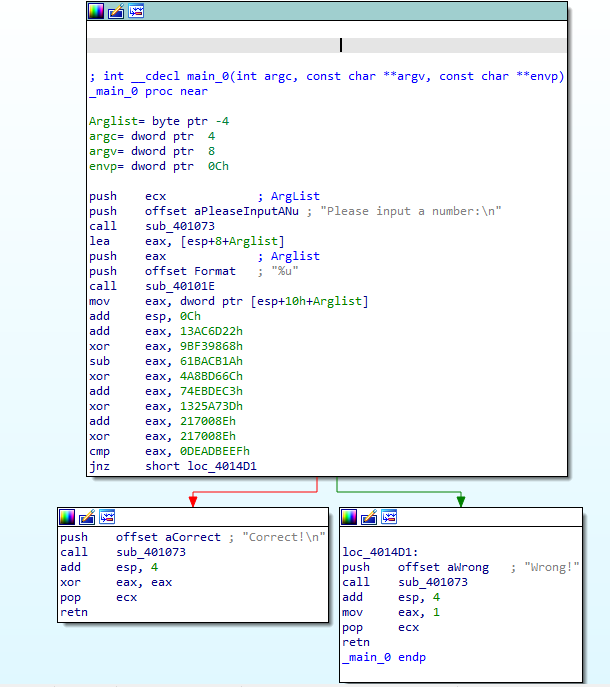
**图 3 逆向分析，完成task1练习**

1. **task2**

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

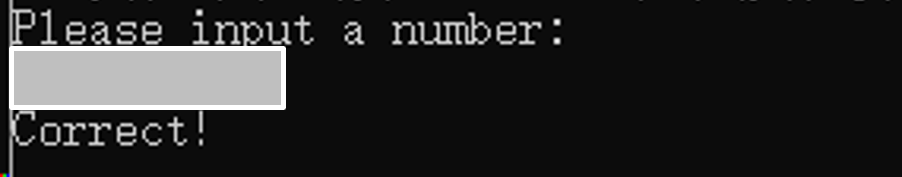


**图4 task2.exe的反汇编代码**



**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。



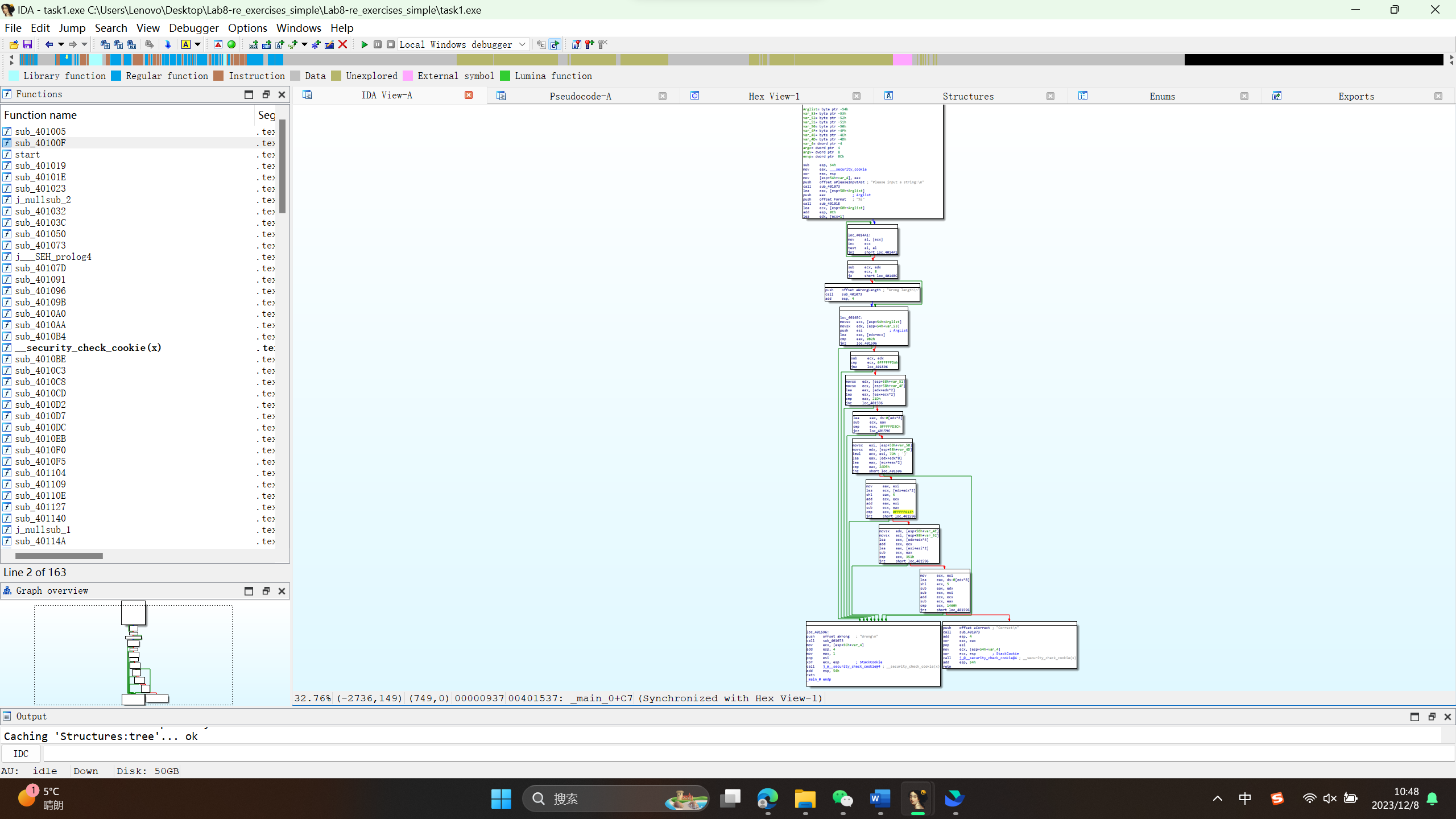
**图 6 逆向分析，完成task2练习**

**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。**

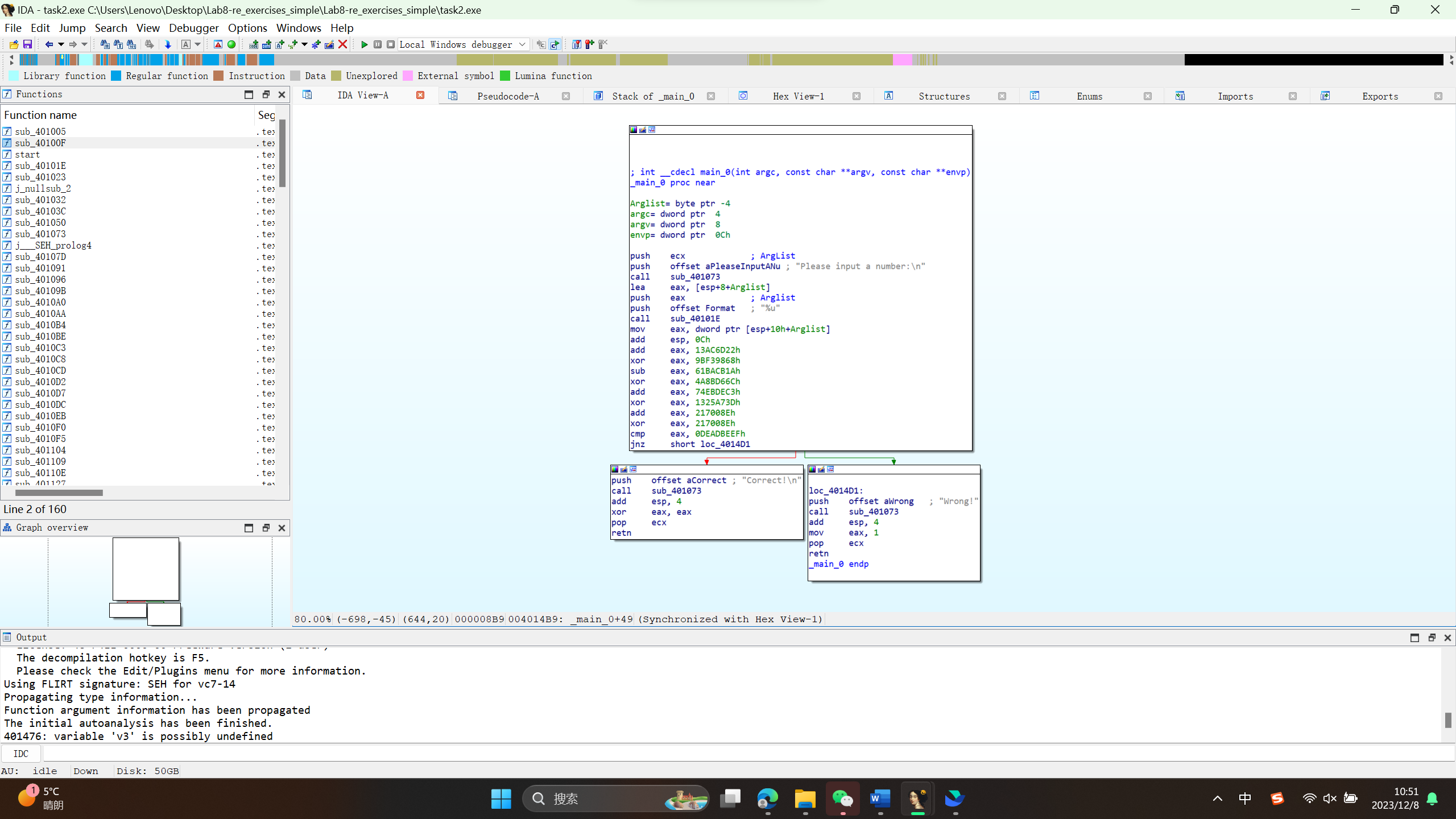
**3.1.1 task1部分截图**

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**

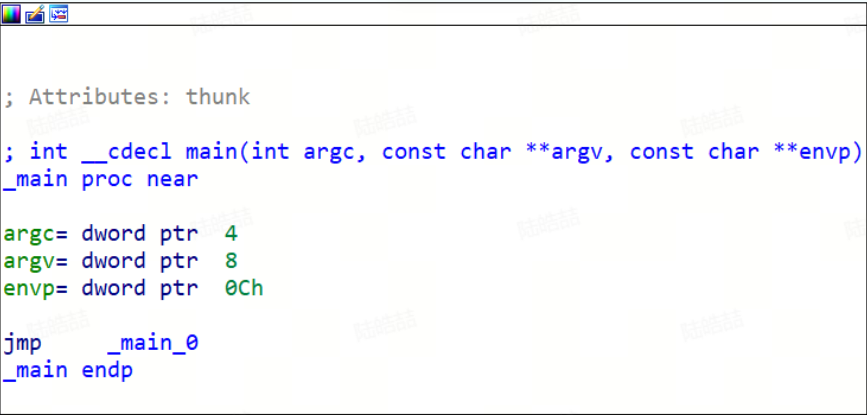
同理，将task2也拖动到IDA中打开，即可获得如下的反汇编代码块。

****

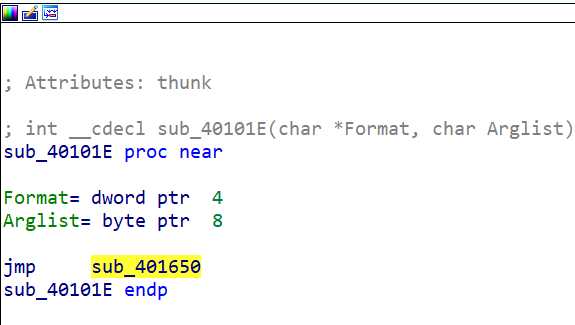
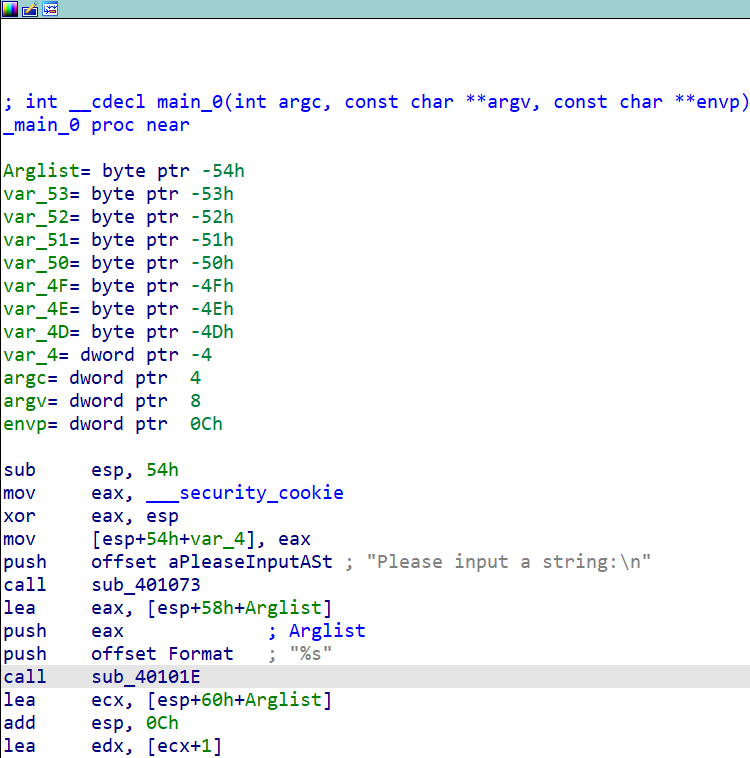
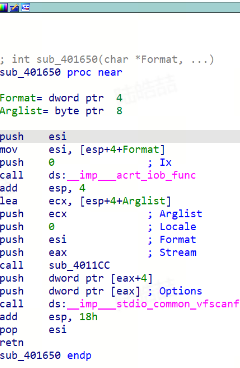
**3.2分别针对task1、task2反汇编代码的计算过程、数据结构、条件判断、分支结构等信息进行逆向分析，在实验报告中记录逆向分析的详细过程，并画出程序流程图。**

**3.2.1 task1部分分析**

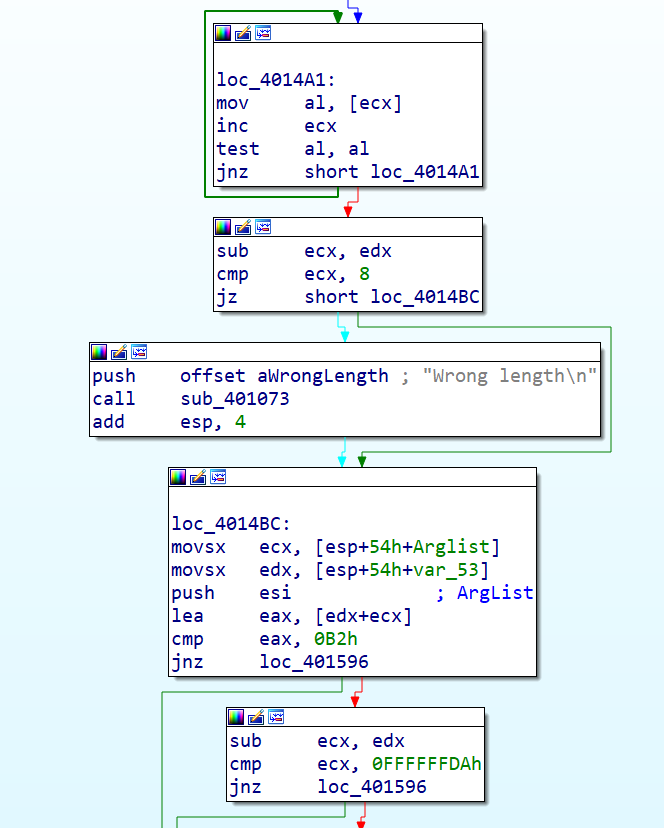
首先进入IDA中，我们发现该程序首先接收了程序传入的argc（外部参数个数）、argv（外部参数）与envp（环境变量），然后跳转到main函数当中去。



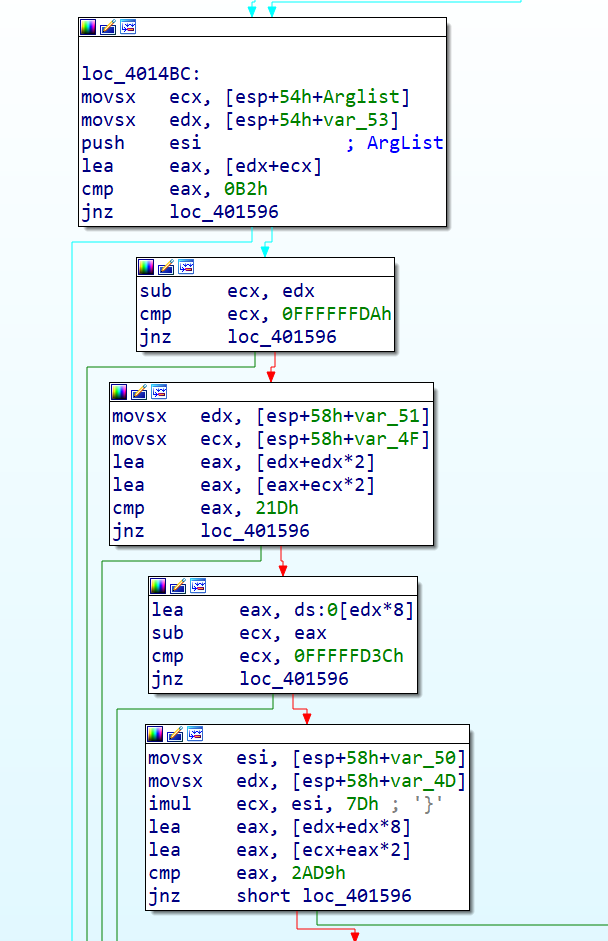
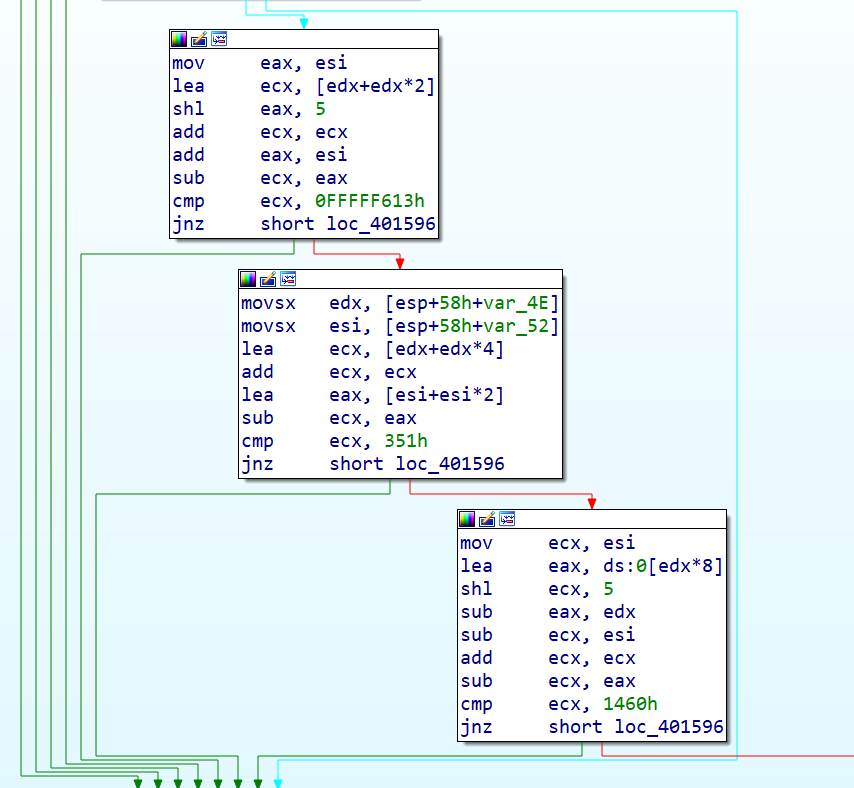
首先，主函数是一个监听字符串输入的函数sub\_40101E函数，双击进入函数，发现进入到sub\_401650中。该程序的功能应该是将一个字符串输入到Arglist中去。

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OFFFFFFDAh（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；

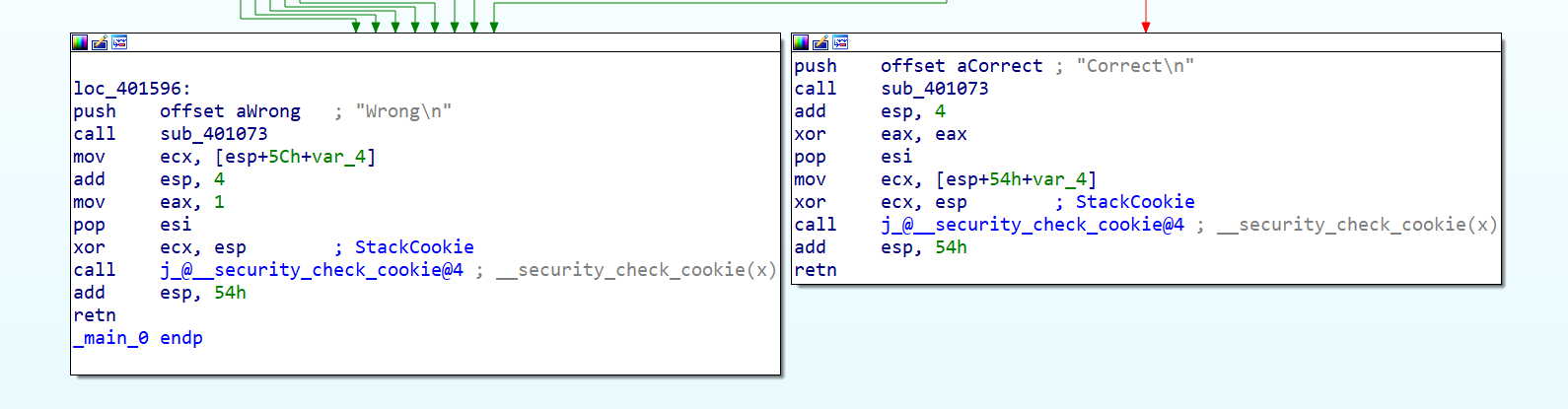
**第四步**判断ecx-edx\*8，即第六个字符-第四个字符的八倍是否与OFFFFFD3Ch（即-708）相等；

**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，化简后即为判断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0FFFFF613h（即-2541）相等；

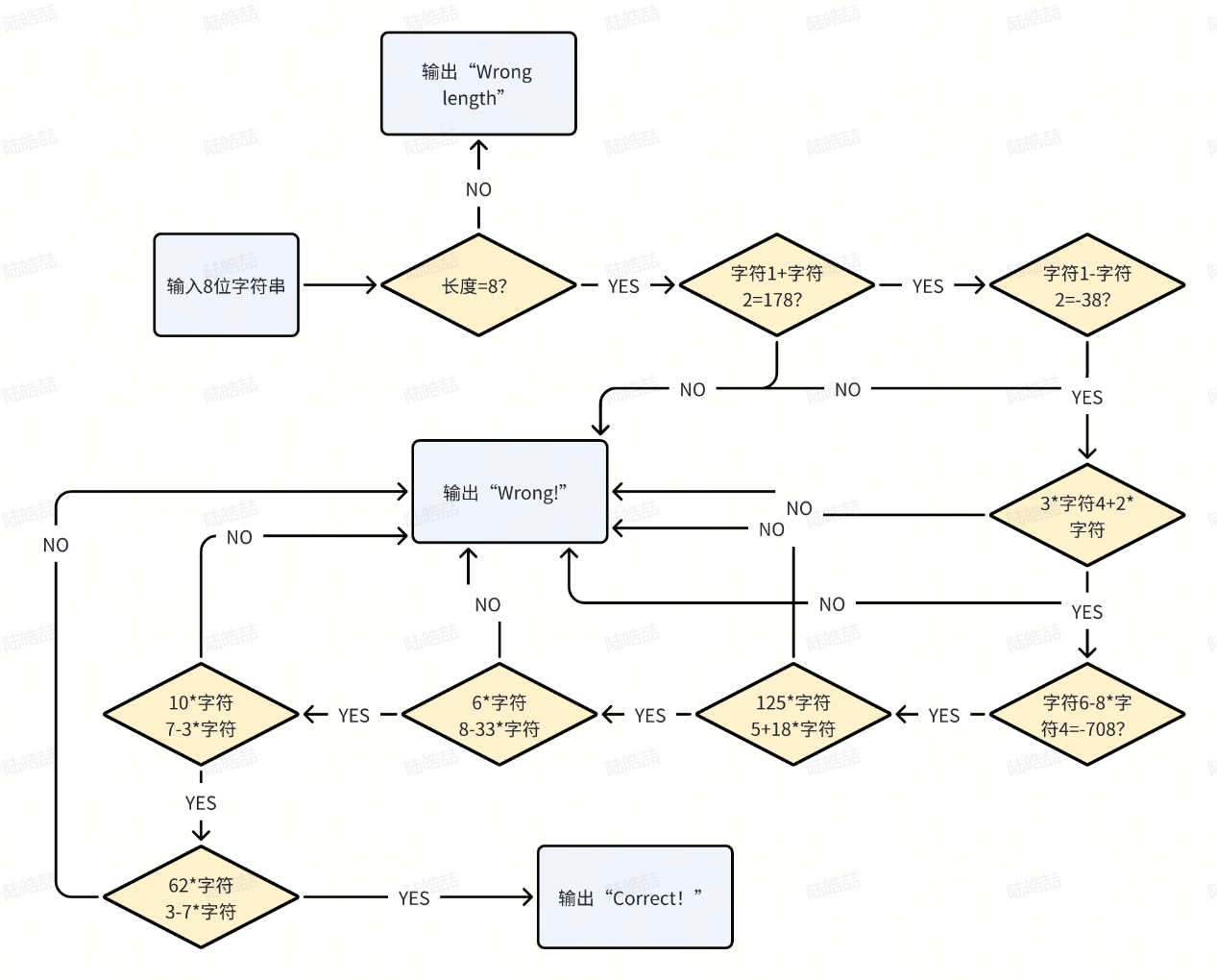
**第七步**是将第七个数存入edx中，第三个数存入esi中，判断10\*第七个数-3\*第三个数是否与351h（即849）相等；

第八步是判断62\*第三个数-7\*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。

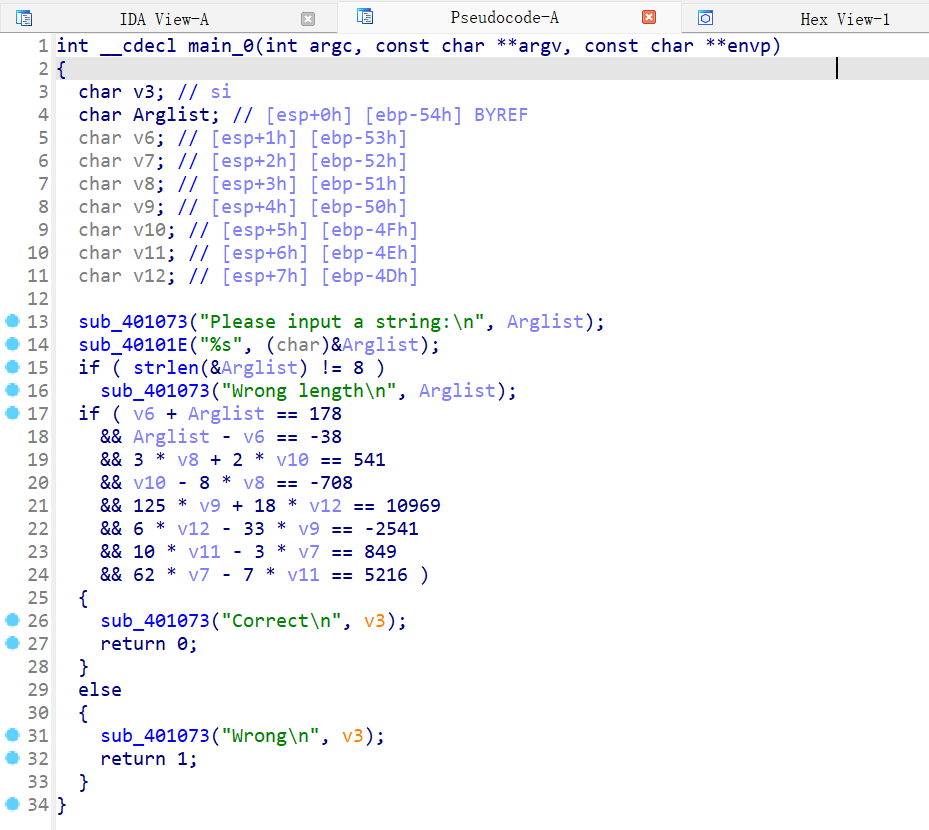


结束阶段，只要上述的八个条件有一个没有满足，那么就会输出“Wrong”,反之输出“Correct！”

根据上述的描述，我们画出该程序的流程图：

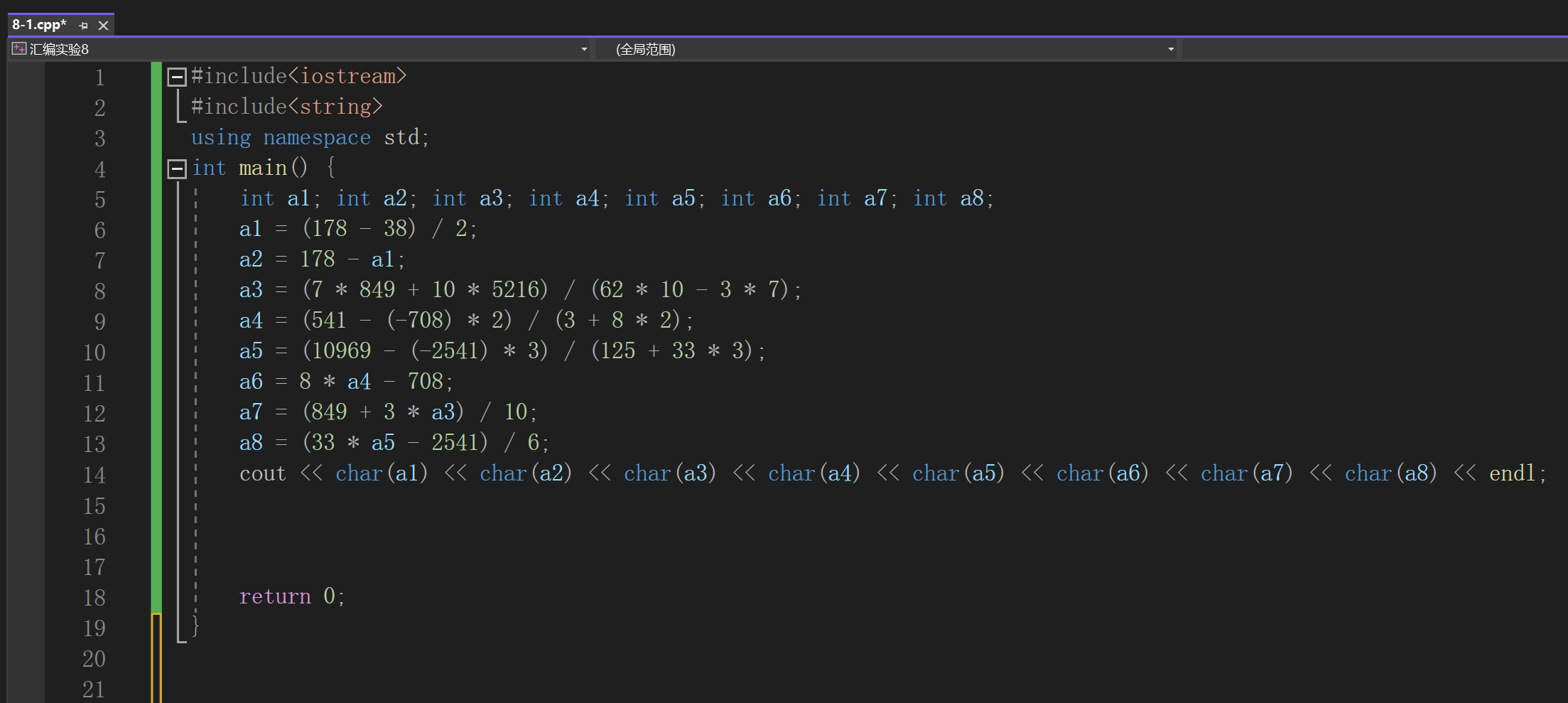


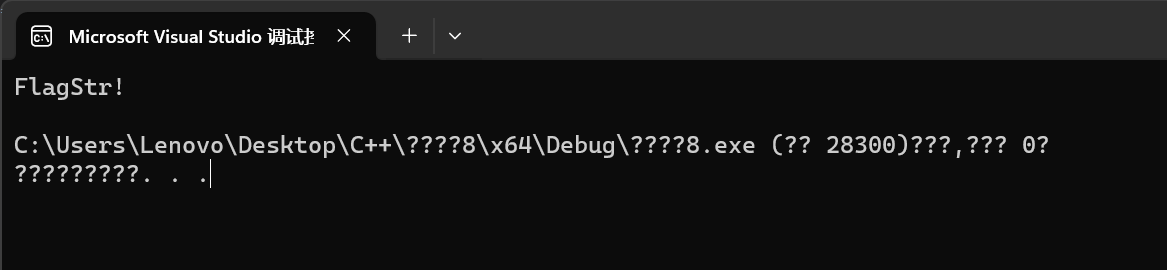
我们在分析完具体思路之后，按下“F5”进行检验，发现逆向程序的基本思路是与我们相同的。



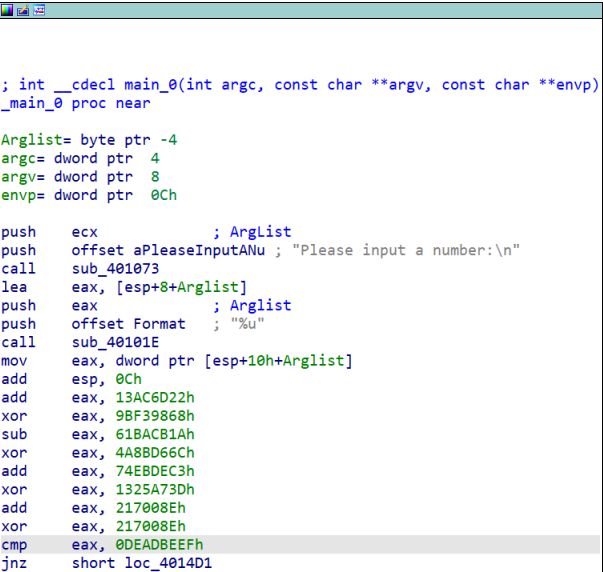
我们完成了对task1的反向分析，现在我们需要编写一个C++程序来进行对输入的解析。

C++代码如下所示：



如图所示，运行结果是FlagStr!

**3.2.2 task2部分分析**



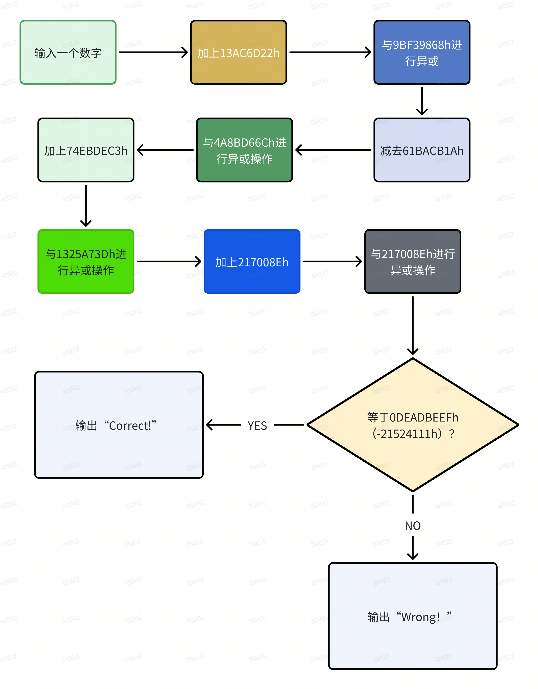
从上面的反汇编代码中，我们对其进行分析。

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

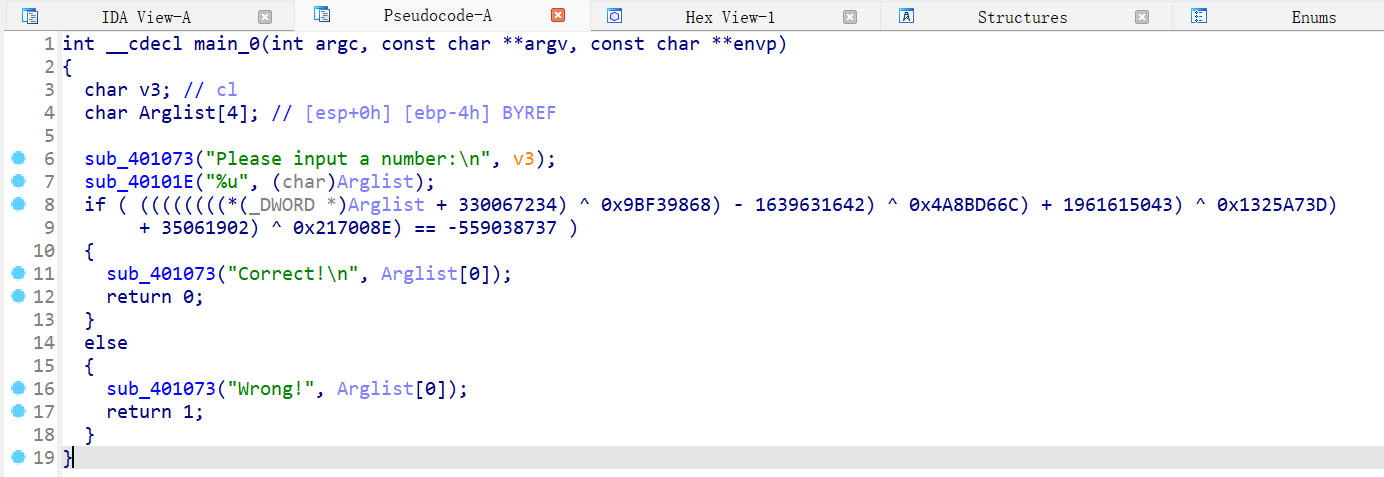
然后我们输入一个数字，将其赋值到寄存器eax中。然后我们开始计算，先

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：

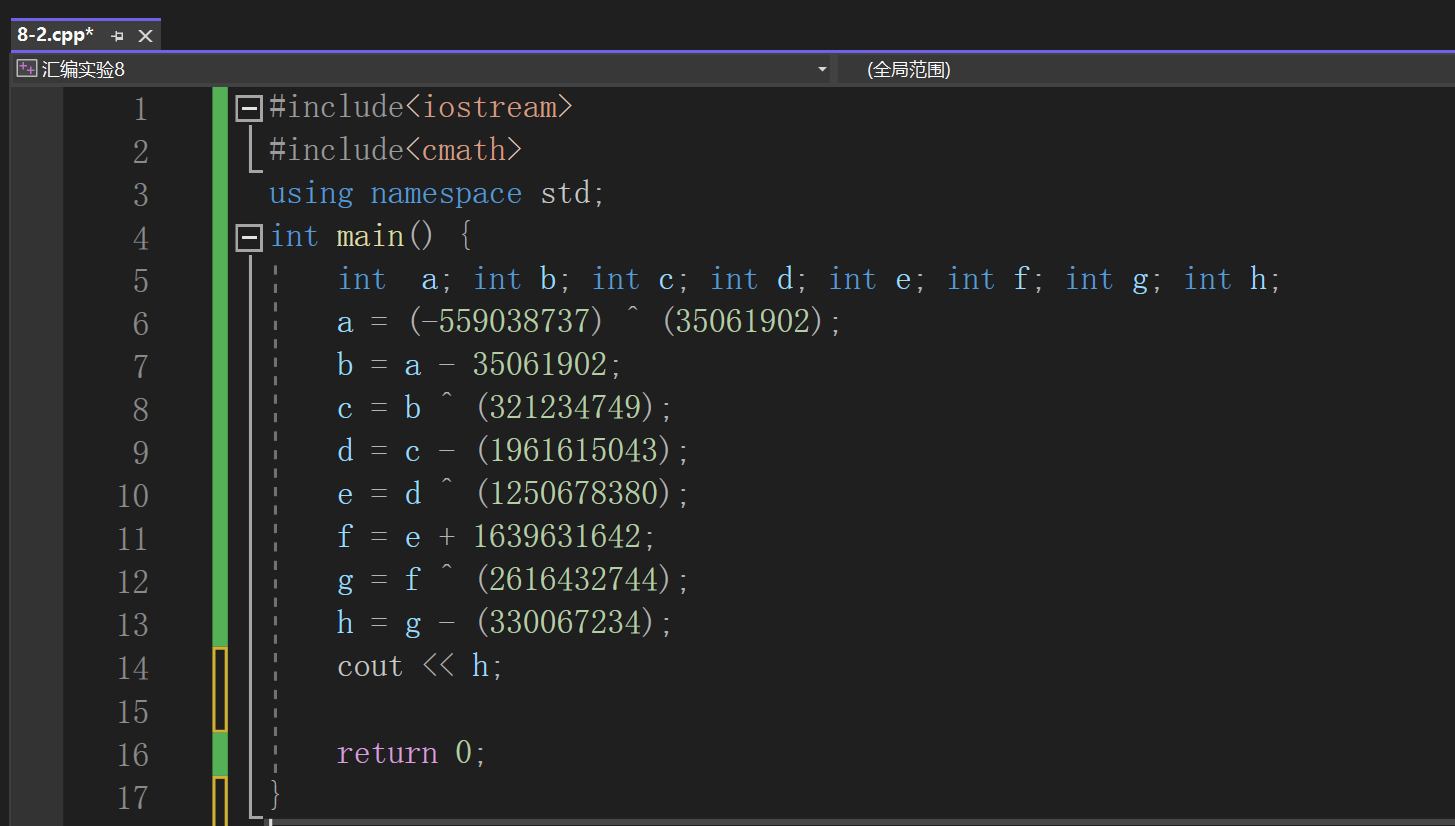


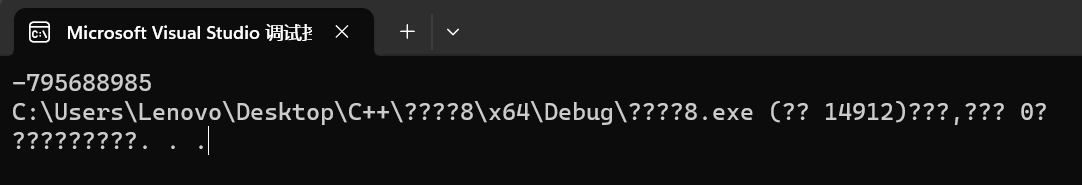
我们在分析完具体思路之后，按下“F5”进行检验，发现逆向程序的基本思路是与我们相同的。



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。

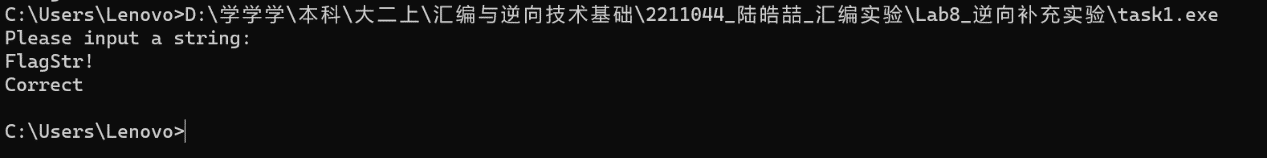
C++代码如下所示：



如图所示，运行结果是-795688985。

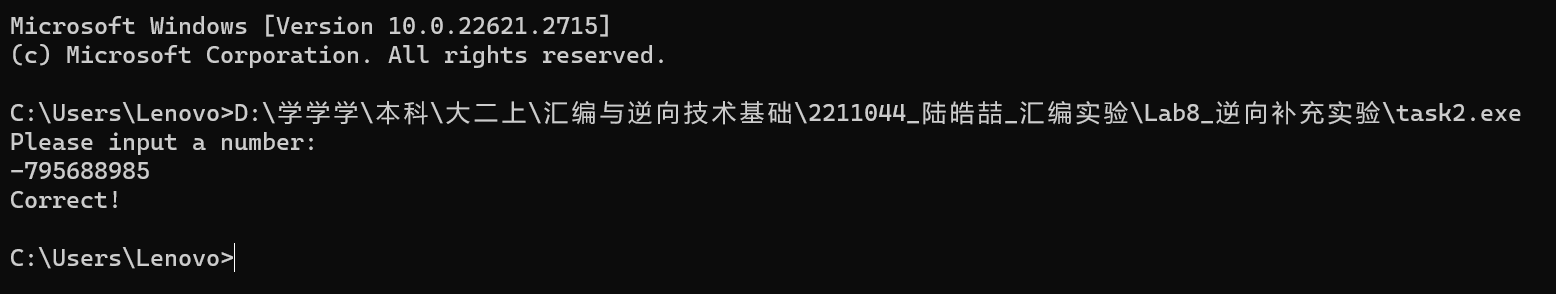
如果我们不考虑负数的情况时，我们也能够得到一个答案，我们经过计算可得，该值为3499278311。

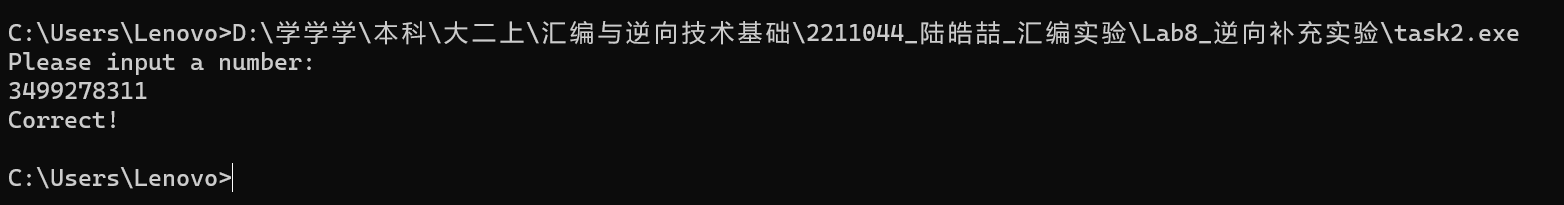
**3.3分别运行程序task1、task2，根据提示输入逆向挑战的结果，获得“Correct”输出，提供截图。**

对于task1，我们输入C++程序所解出来的结果，得到以下的结果：

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：





可以发现输出“Correct！”，实验成功！