

**汇编语言与逆向技术课程实验报告**

**实验九： re\_exercises\_advanced**

****

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、进一步熟悉静态反汇编工具IDA Freeware；

2、熟悉将反汇编代码进行反编译的过程；

3、掌握对于反编译伪代码的逆向分析；

4、运用熟悉的编程语言，实现简单的脚本编写。

**二、实验原理**

**（一）task3**

1、通过IDA Freeware得到task3.exe的反汇编代码，如图1和图2所示。

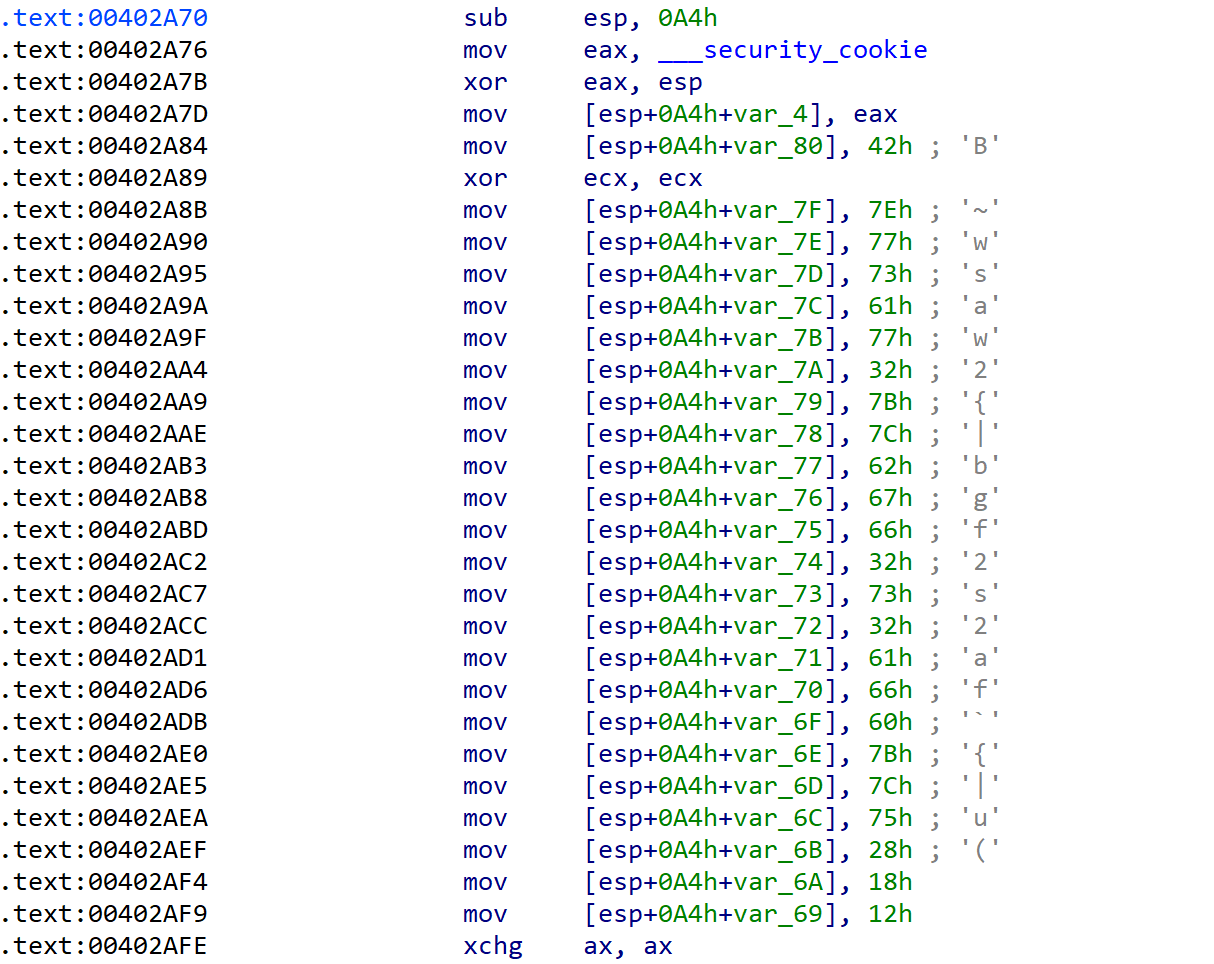


图 1 task3.exe的反汇编代码

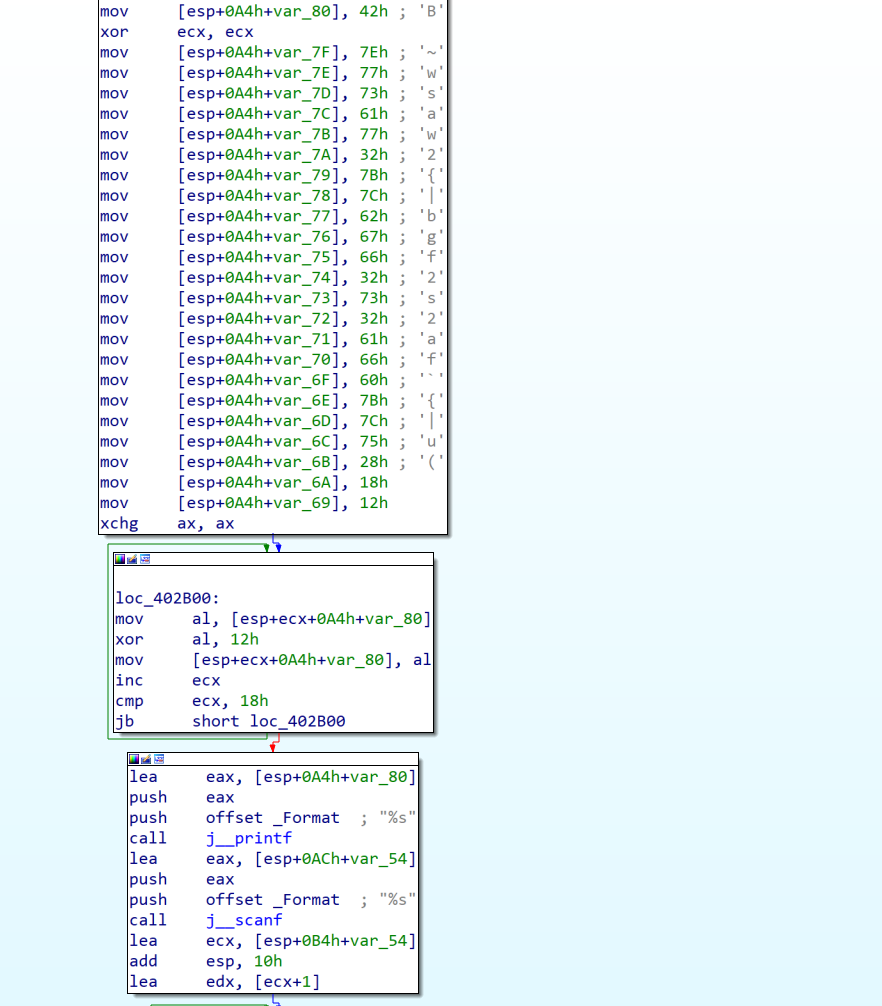


图 2 task3.exe反汇编代码的图形化显示

2、使用IDA的反编译功能（F5快捷键）得到伪代码，如图3所示。

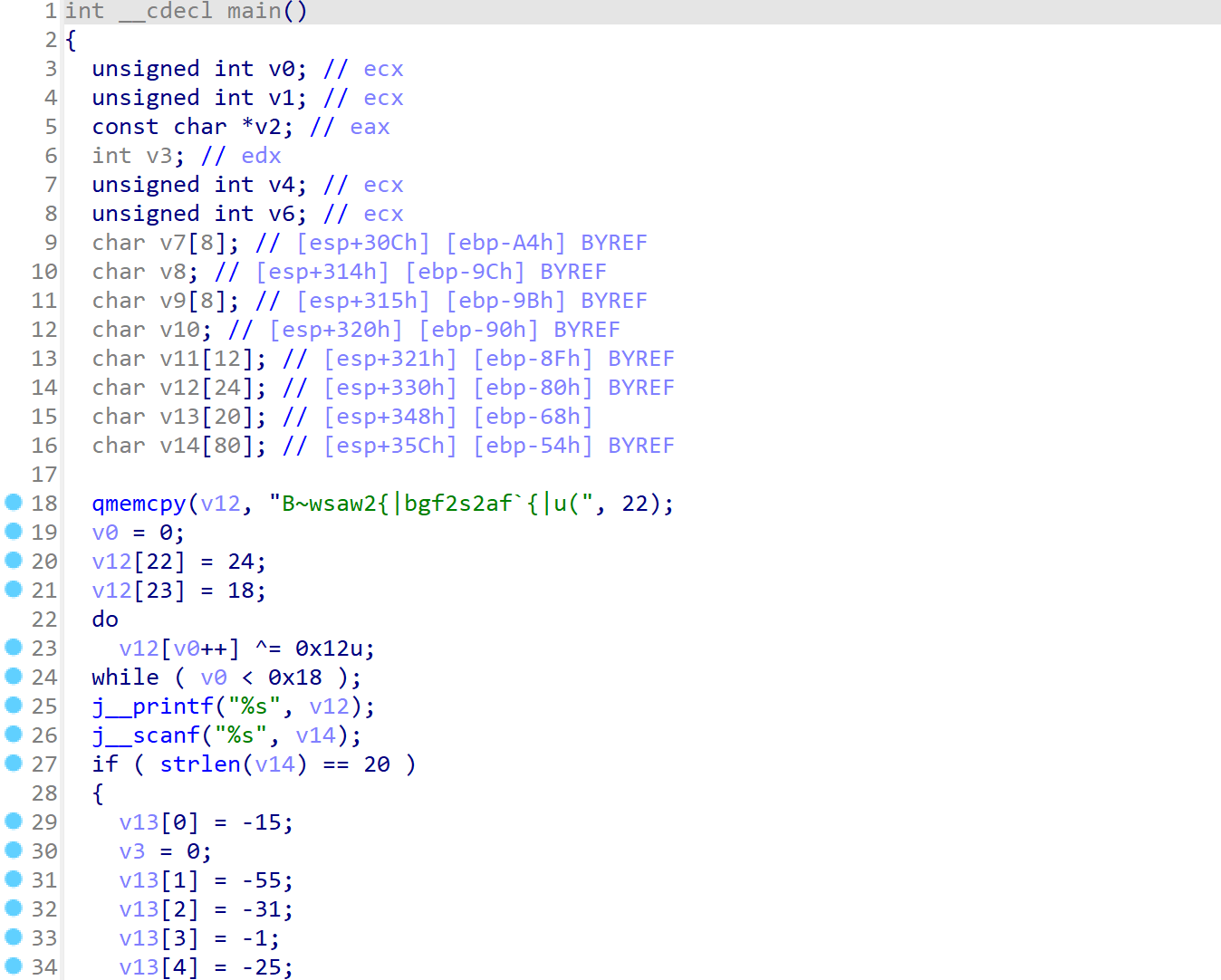


图 3 task3.exe的反编译伪代码

3、通过对反汇编命令及反编译伪代码的分析，逆向推理出待输入字符串的计算公式；

4、使用熟悉的编程语言（C++、Java、Python等）对待输入字符串进行计算，完成逆向分析挑战。

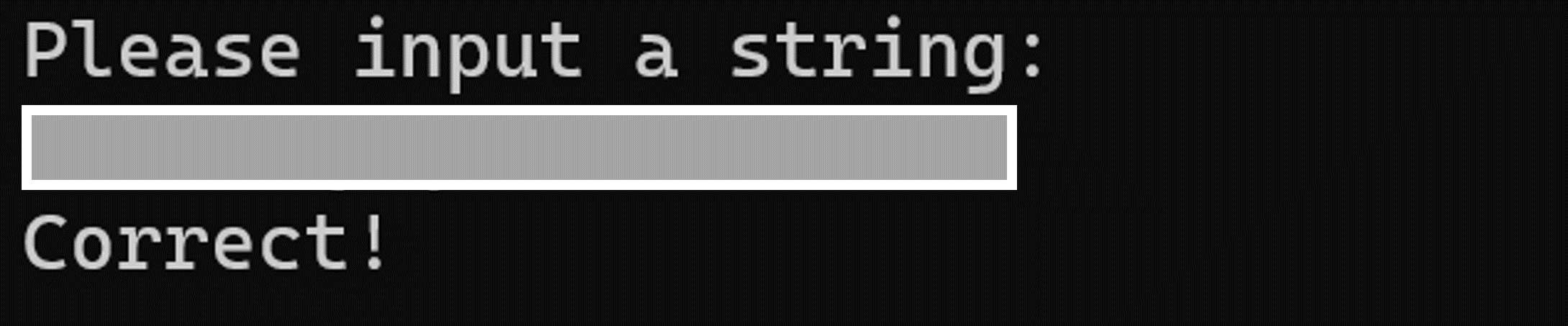


图 4 逆向分析，完成task3练习

**（二）task4**

1、通过IDA Freeware得到task4.exe的反汇编代码，如图5和图6所示。

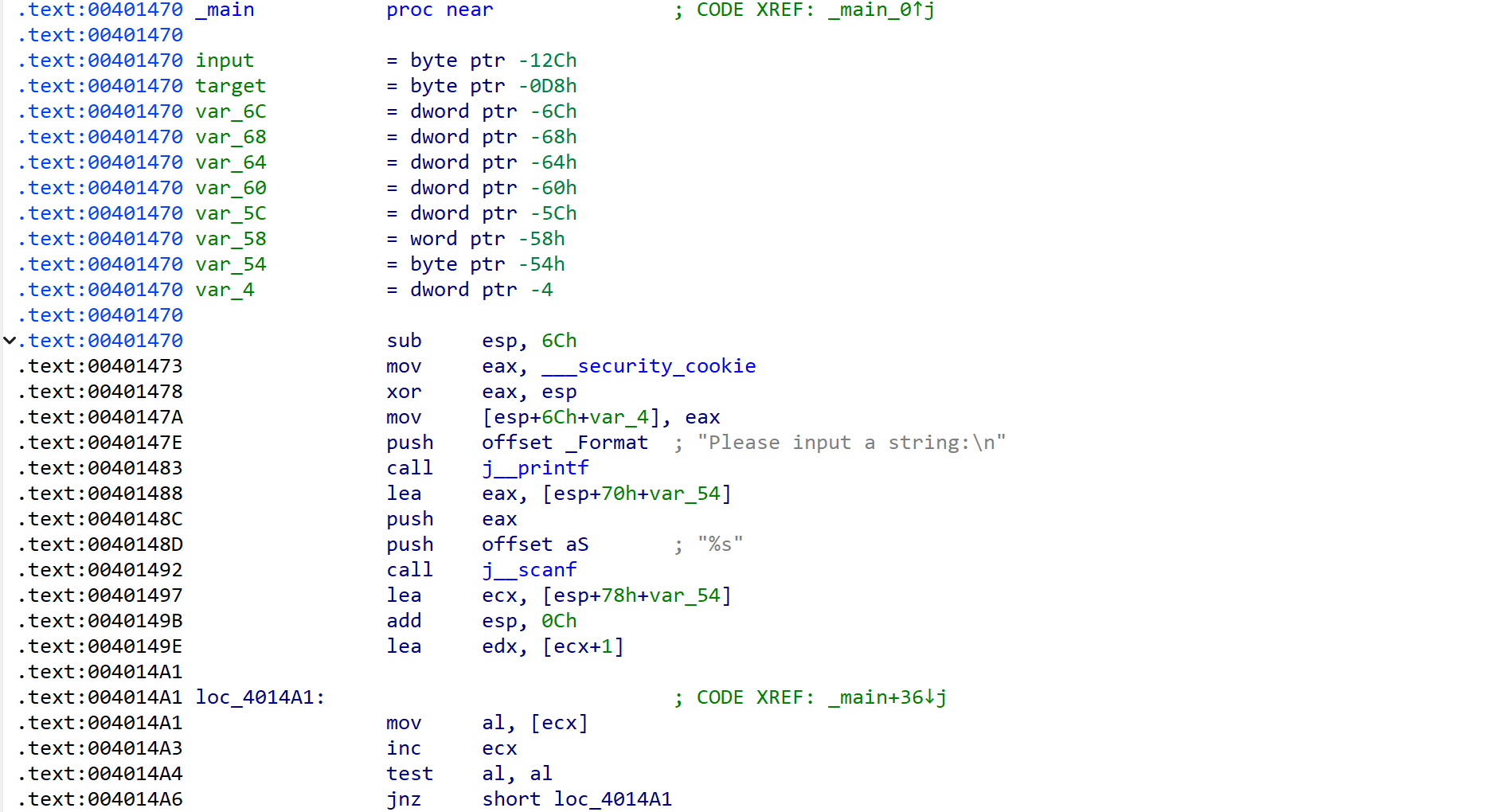


图 5 task4.exe的反汇编代码

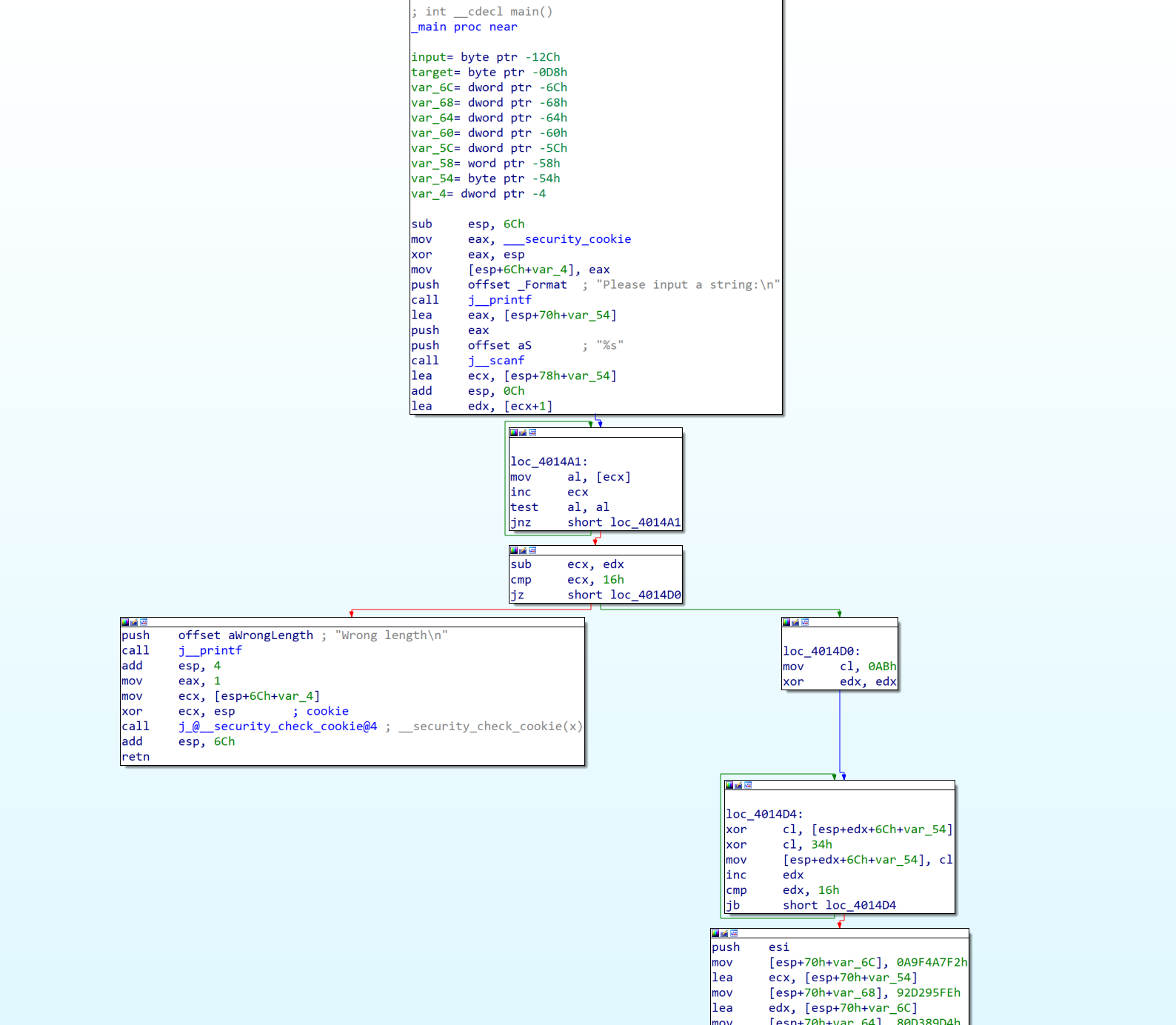


图6 task4.exe反汇编代码的图形化显示

2、使用IDA的反编译功能（F5快捷键）得到伪代码，如图7所示。右键点击数字对象可实现数制转换。

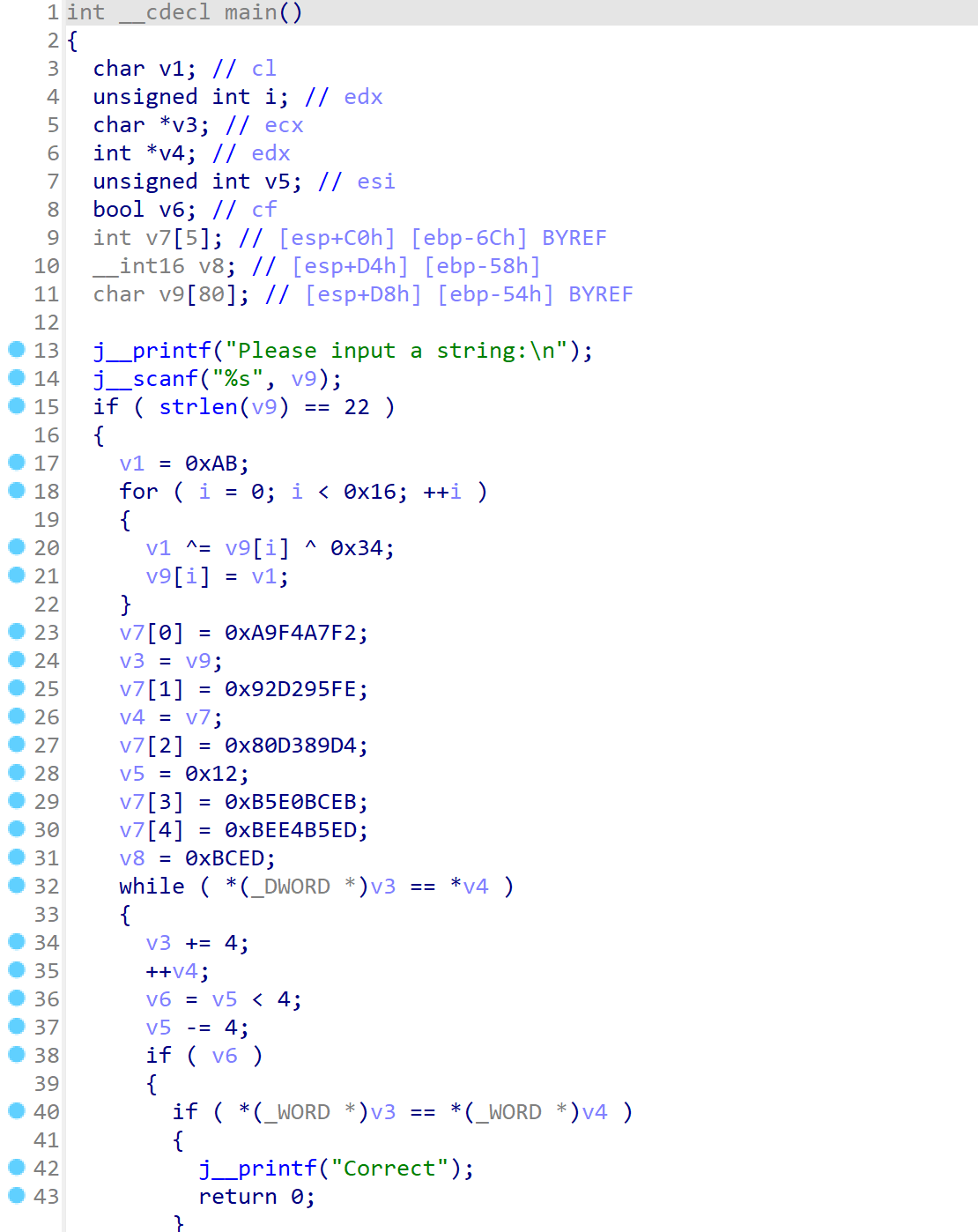


图7 task4.exe的反编译伪代码

3、通过对反汇编命令及反编译伪代码的分析，逆向推理出待输入字符串的计算公式

4、使用熟悉的编程语言（C++、Java、Python等）对待输入字符串进行计算，完成逆向分析挑战。



图8 逆向分析，完成task4练习

**三、实验报告**

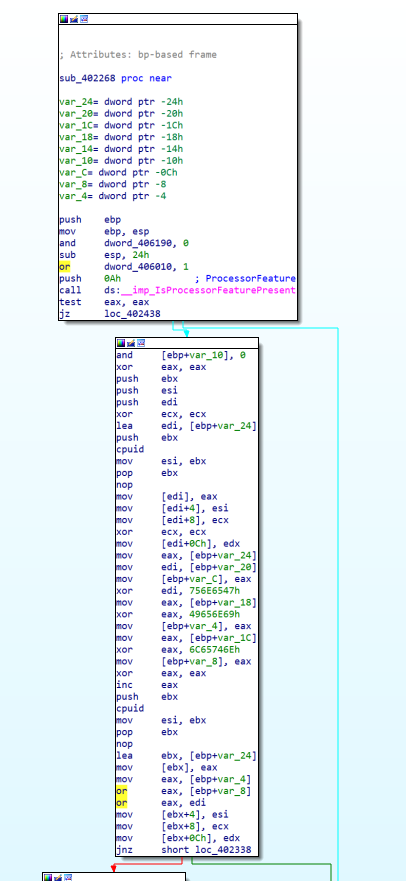
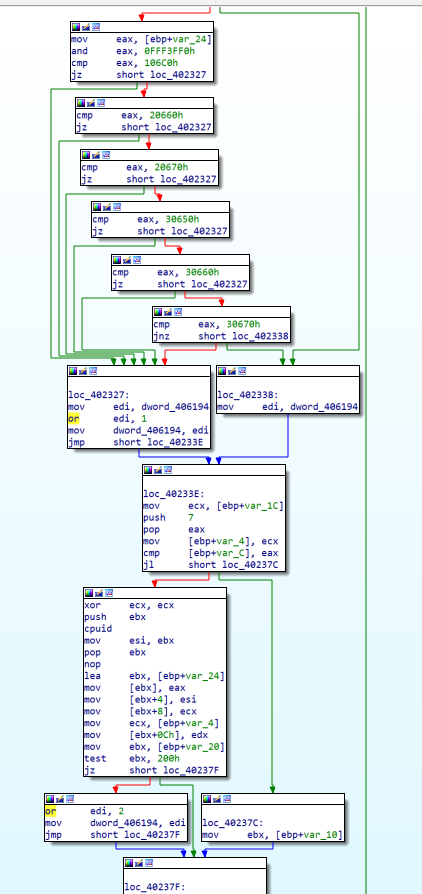
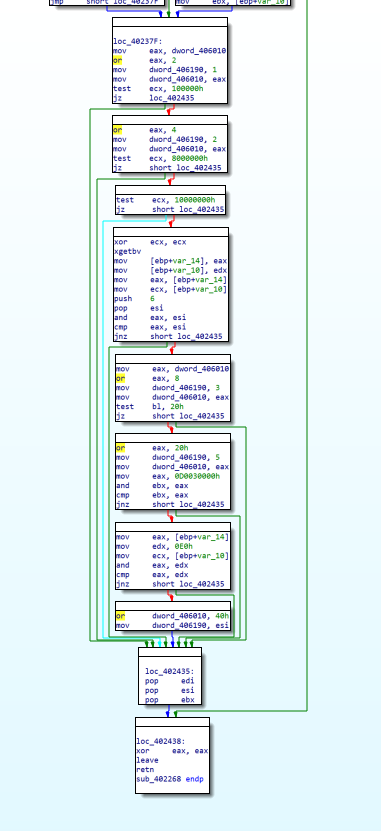
**3.1分别针对task3、task4使用IDA Freeware，获得可执行文件的反汇编代码及反编译伪代码，提供截图。**

**3.1.1 task3**

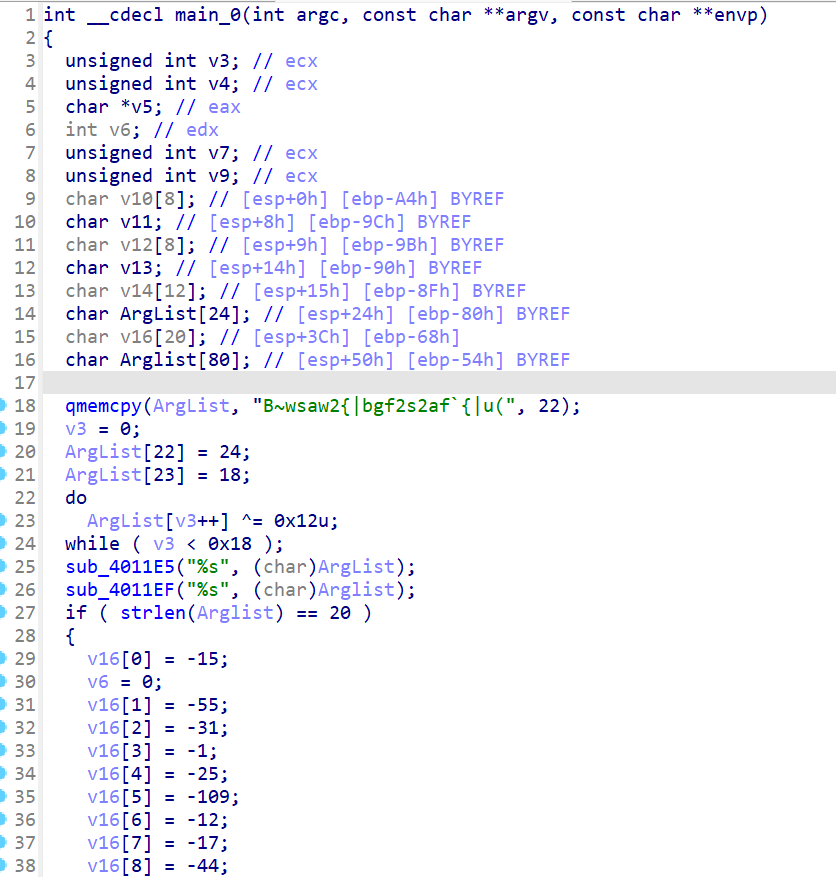
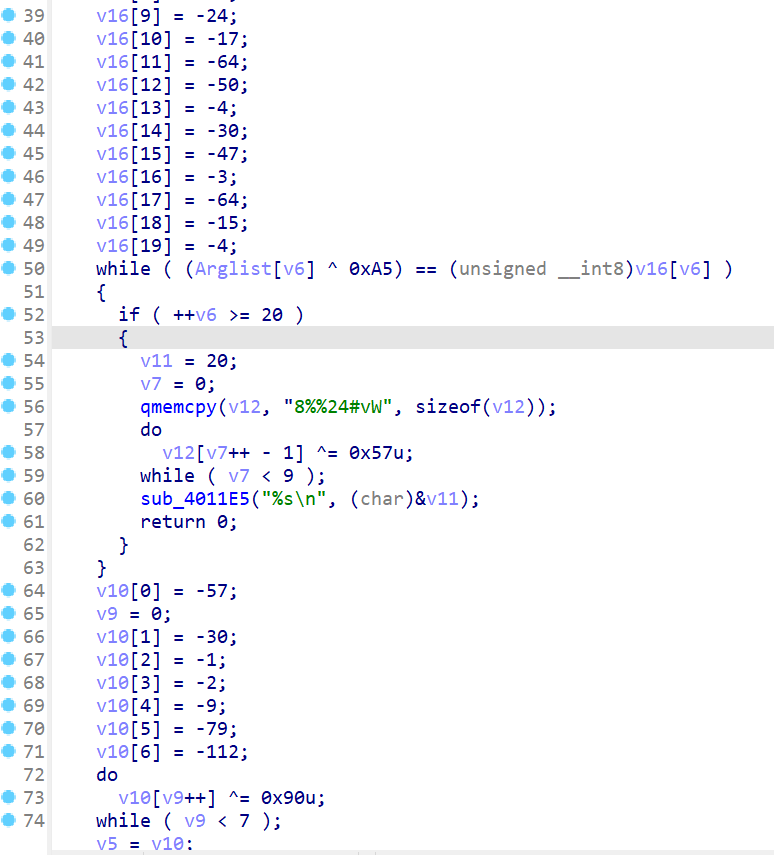
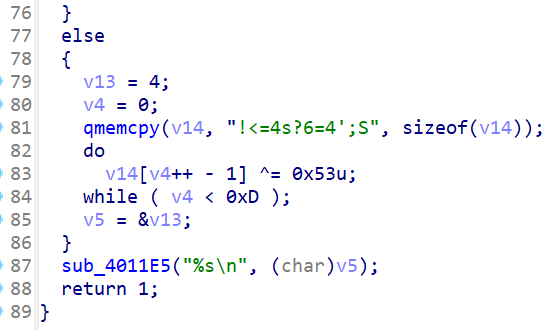
反汇编代码：



反汇编代码图形化显示：

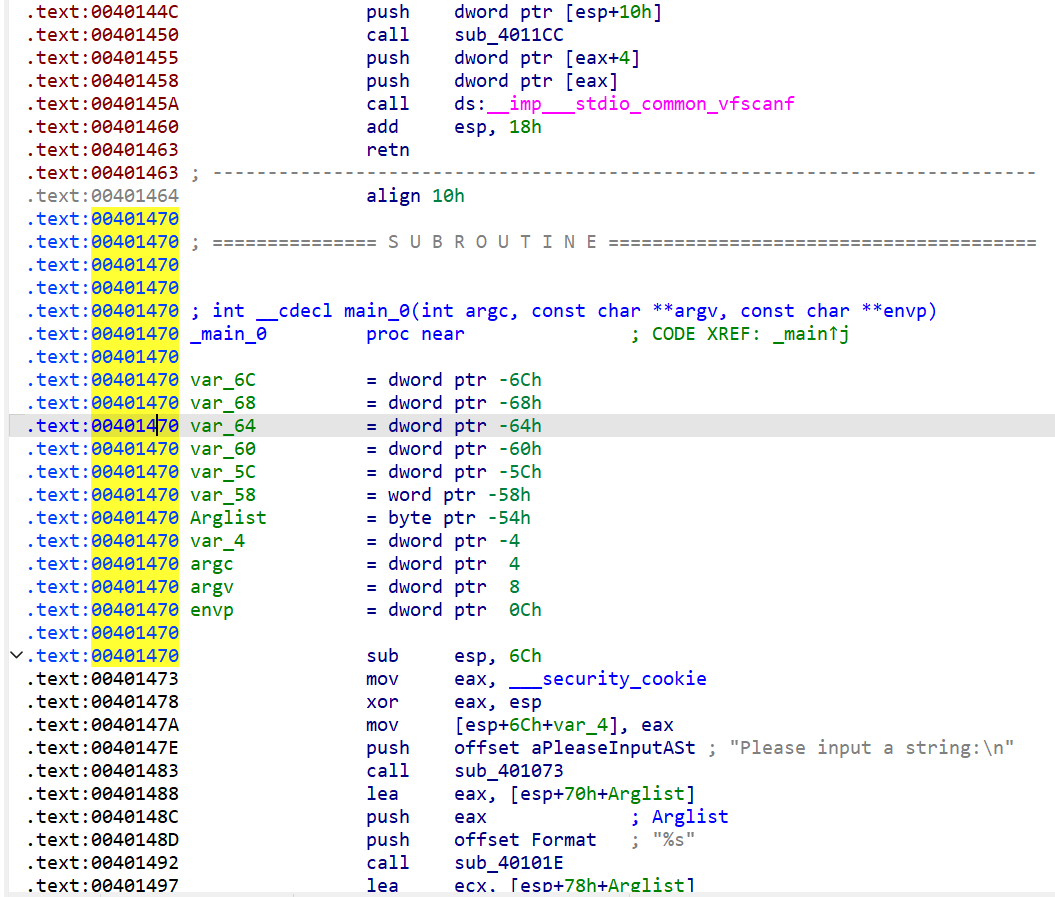
  

反编译伪代码：

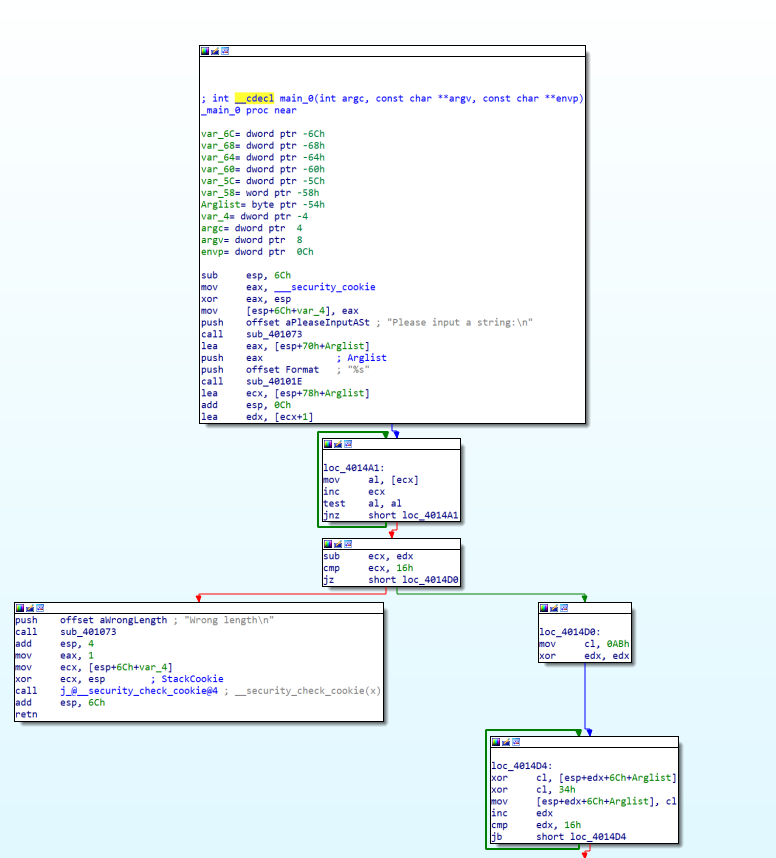
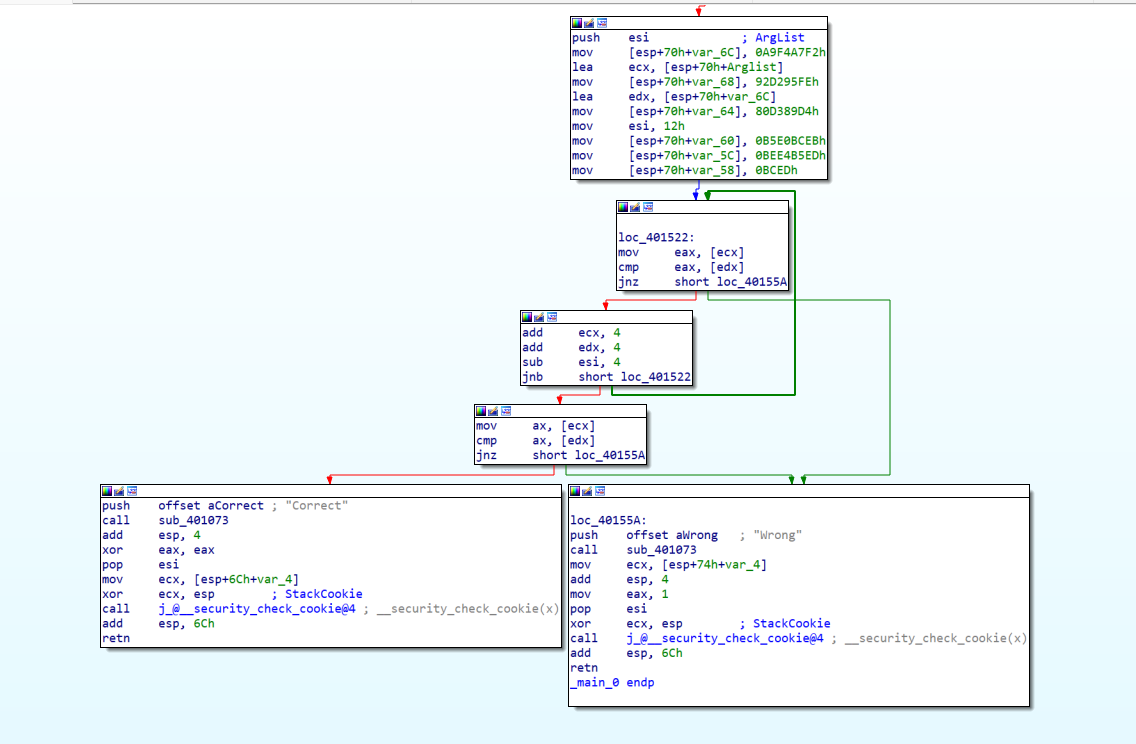
**** **** 

**3.1.2 task4**

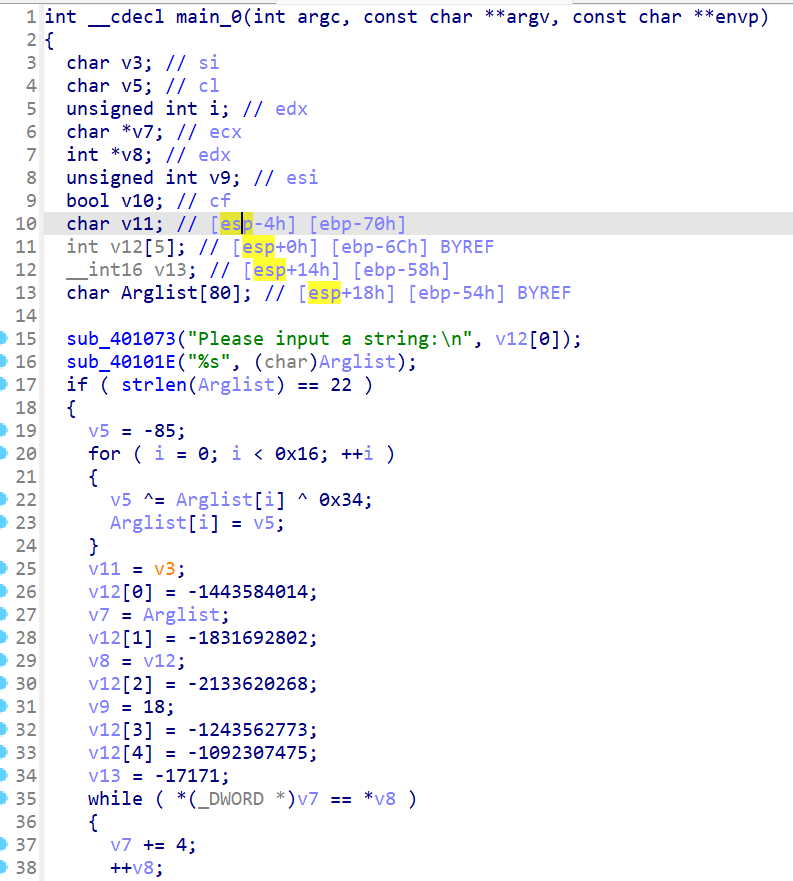
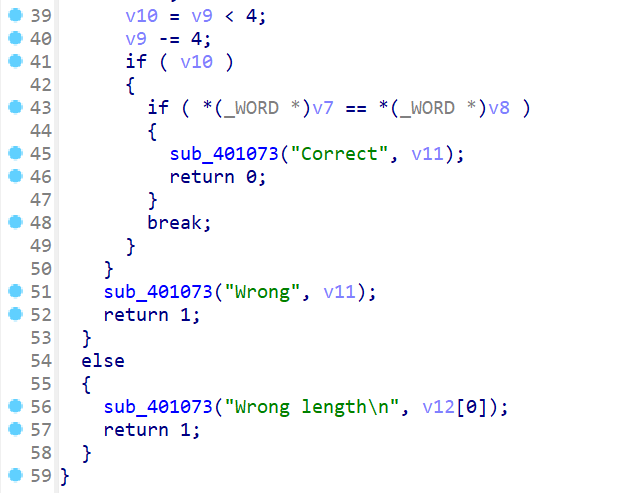
反汇编代码:



反汇编代码图形化显示：

反编译伪代码：

**** ****

**3.2分别针对task3、task4反编译伪代码的计算过程、数据结构、条件判断等信息进行逆向分析，列出正确输入字符串的计算公式。**

**3.2.1 task3**

观察代码，发现有很多地方都出现了加密的过程，比如说开始时的qmemcpy(ArgList, "B~wsaw2{|bgf2s2af`{|u(", 22)这句话就是对字母进行了加密，我们编写程序（具体程序全部放到后面一个版块），来求解字符串原本的含义。得出第一处的含义为“Please input a string”。

接下来是一个输入函数，将输入的字符串存入Arglist中，然后判断其长度是否为20，对其进行相应处理，最后是结果是输出v5的字符串，我们先对长度不为20的部分（即错误部分）进行分析。

如果长度不为20的话，错误部分首先将v13赋值4，将一个字符串赋给v14，从0-1至0xd-1进行循环遍历v14字符串进行异或，注意到下图v13为esp+14h，v14起始为esp+15h，因此，v14[-1]即为v13，最后将v5指向v13的地址，最后输出v5的值，所以v5输出的即为v13开始到v14末尾的值。

我们还是用前面的方法解密出字符串，此处字符串的含义为“Wrong length”。

现在对正确的处理逻辑进行分析，当输入的字符串长度为20时，先对v16进行了一串的赋值操作，然后对输入的数据每一位与0xA5进行异或和v16每一位进行比较，当遍历到20个字符（即输入的字符串已经遍历完毕）且全部匹配时，与之前的Wrong length逻辑相同，v11和v12也是相邻的变量，首先将v11和v12初始化，然后对其每一位与0x57进行异或，最后输出结果。

我们先解密出最后的输出结果（猜测是输出correct），最后解密完发现确实是“correct！”！那么我们就有了思路，发现这一块长度为20的代码就是我们要求的那一部分。

还有一处，就是当输出while时，那一处遍历出现问题时就会输出“Wrong！”。

最后，写一个程序来计算出我们的求解结果：只需要用原来的一串数字和0xA5进行异或操作，公式如下所示：







最后得出正确的结果“TLDZB6QJqMJekYGtXeTY”。

**3.2.2 task4**

先输出一个“Please input a string:”，接着输入一个字符，如果字符的长度不等于22的话，就进不了if循环，直接跳转到else界面，输出“wrong length”。现在我们输入一个字符长度为22的字符，先给v5赋值为-85，接下来，进入for循环，做22次操作，将输入值的每一位都与0x34进行异或运算，再跟v5进行异或运算，再将值赋值给输入的第几位。做完22次运算后，我们进入下一个环节，先将操作完的字符串赋值给v7，然后将v12的五位数字都进行赋值。

所以在逆向编写C++程序的时候，我们对于20次异或运算的操作需要进行反向的编写，具体思路就是用本位与前一位进行异或，再与0x34进行异或。第一位的话我们用第一位与0x34进行异或，然后与-85进行异或。

最后进行比较环节，就是先初始化v12，v13，然后让v7和v8的字节每四位比较一次，如果全部相同，那么就输出“correct”，否则就输出“Wrong”。

**输入字符串的计算公式为：**



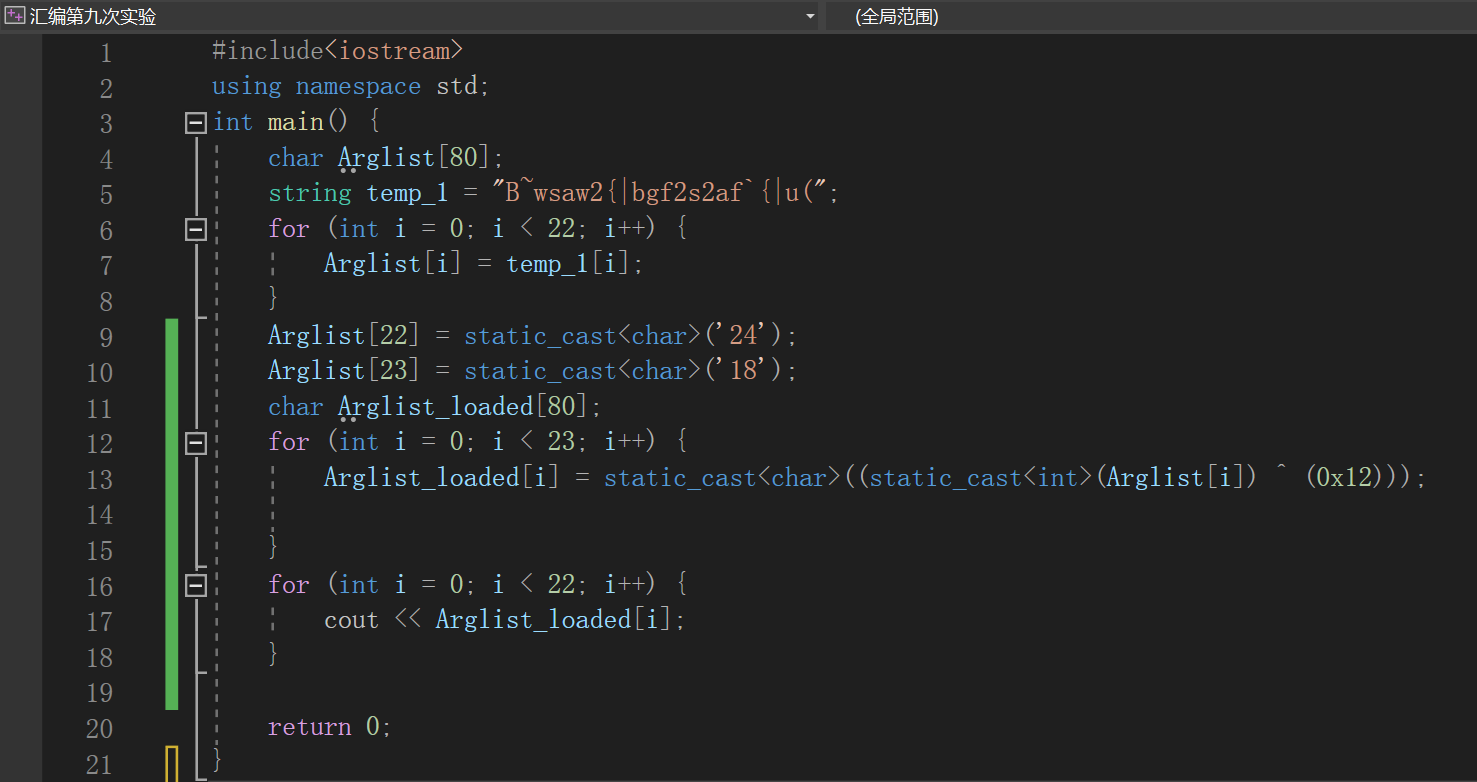


最后得出正确的结果：“magic\_string\_challenge”。

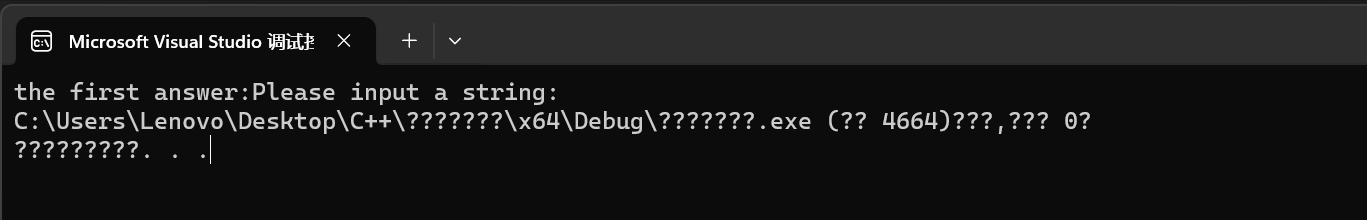
**3.3使用熟悉的编程语言，分别针对task3、task4编写脚本，计算出正确的字符串，提供脚本输出结果的截图**

**3.3.1 task3**

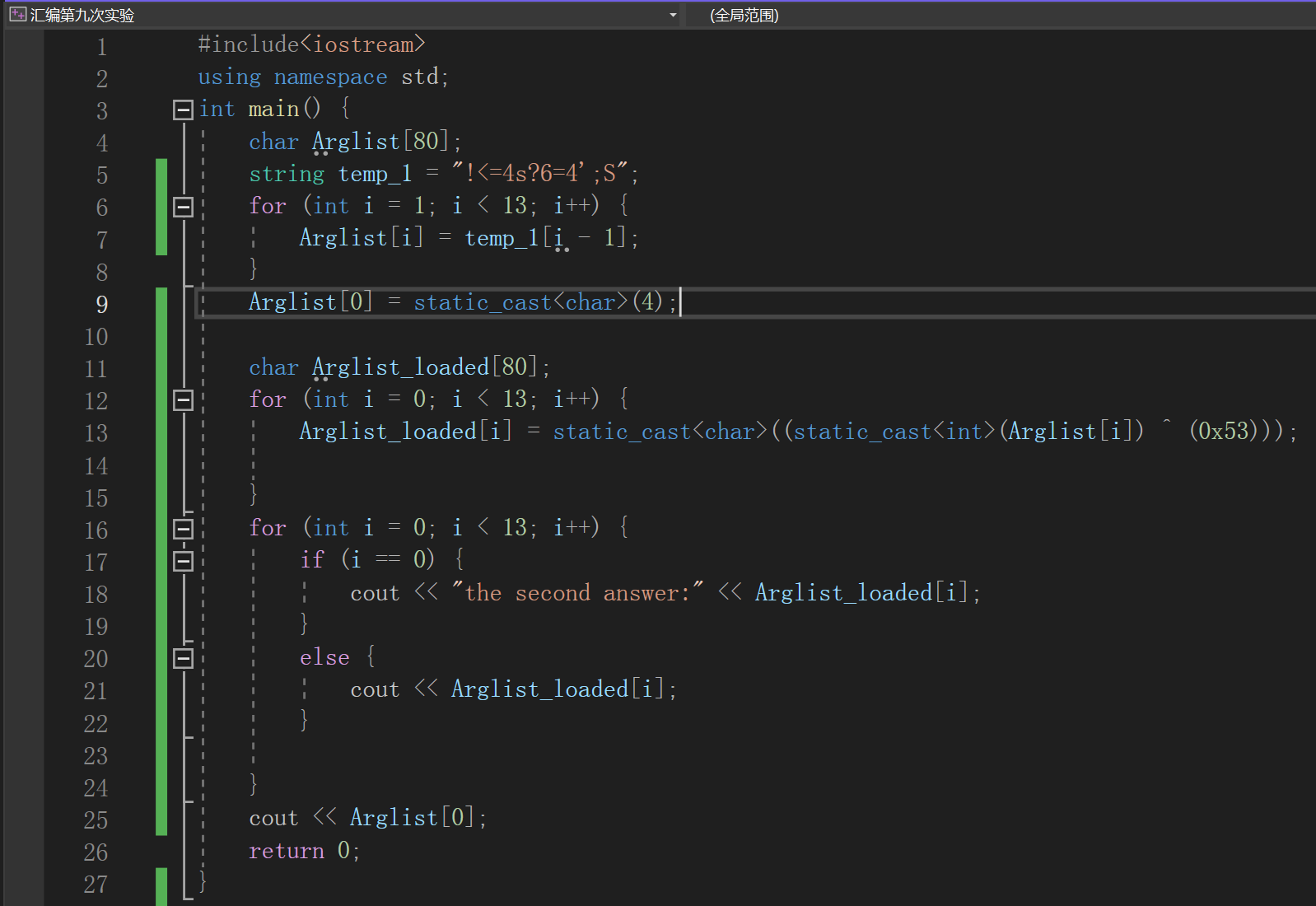
计算第一部分的内容的C++代码：

****

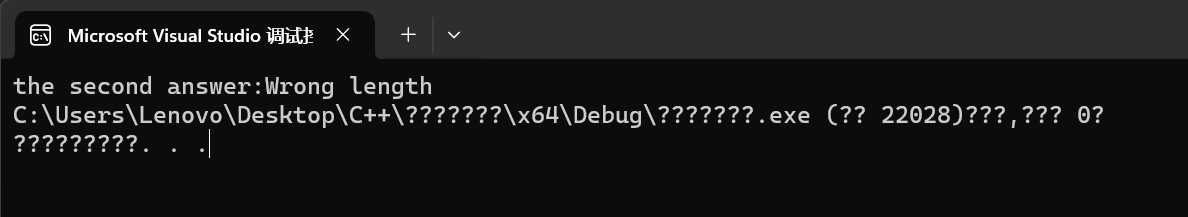
第一处的结果：

****

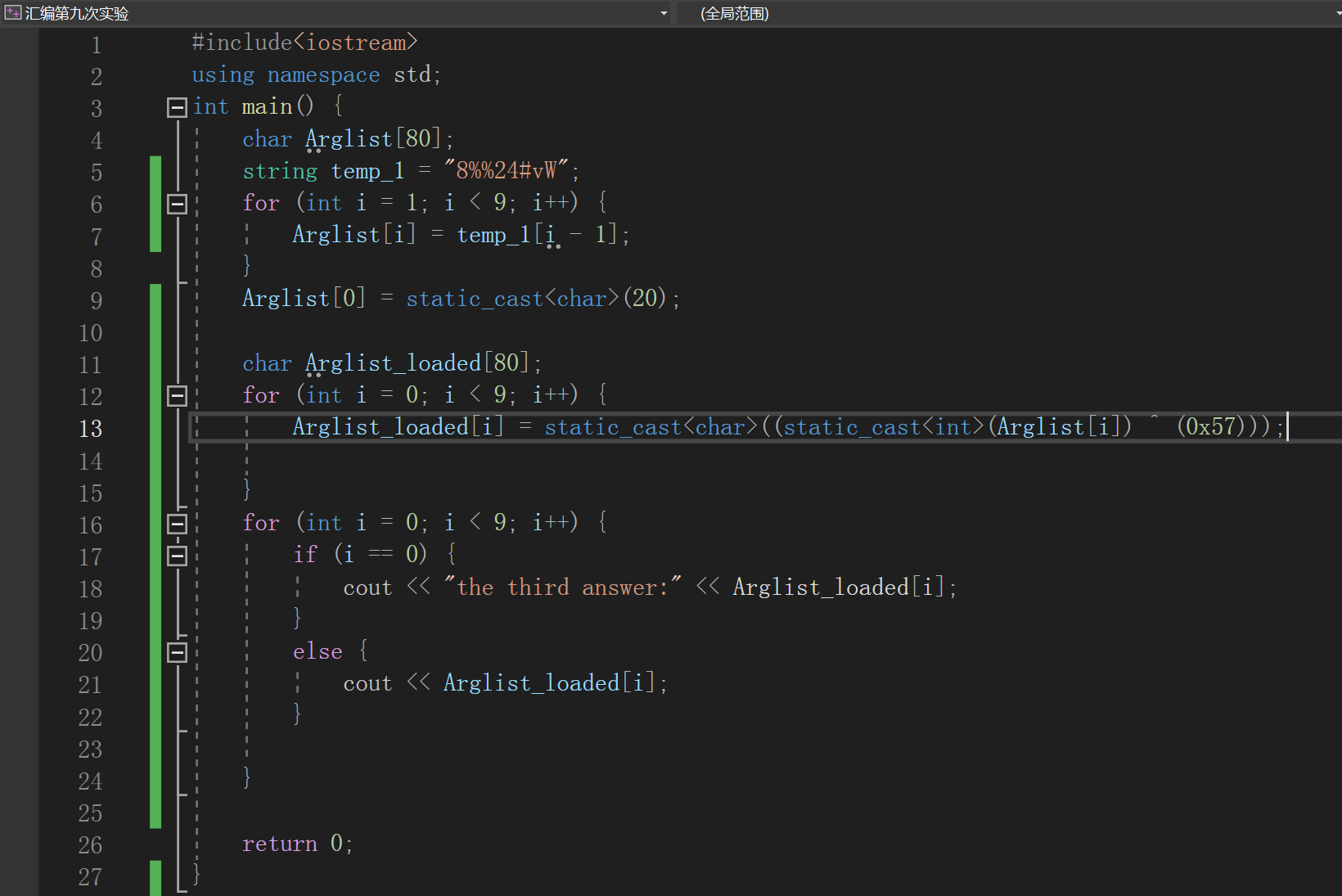
第二部分的代码：



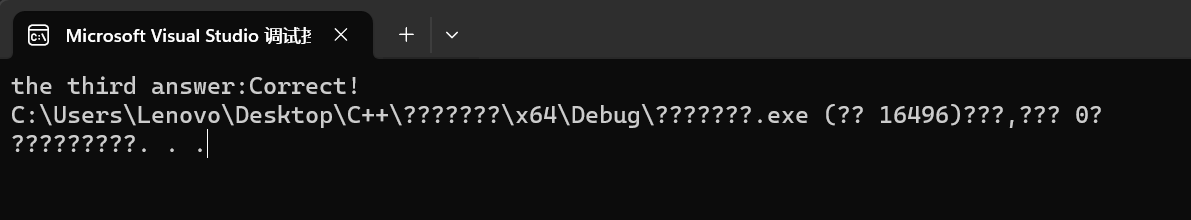
第二部分的输出结果：

****

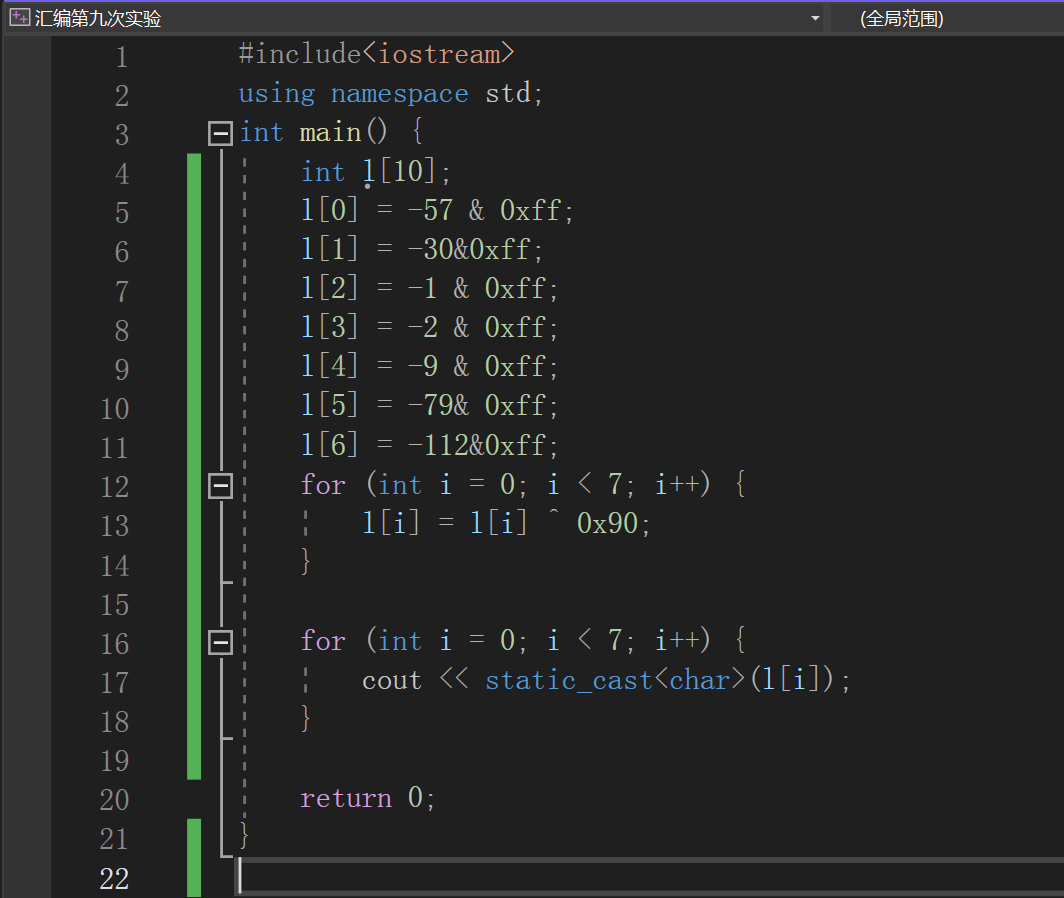
第三部分的代码：



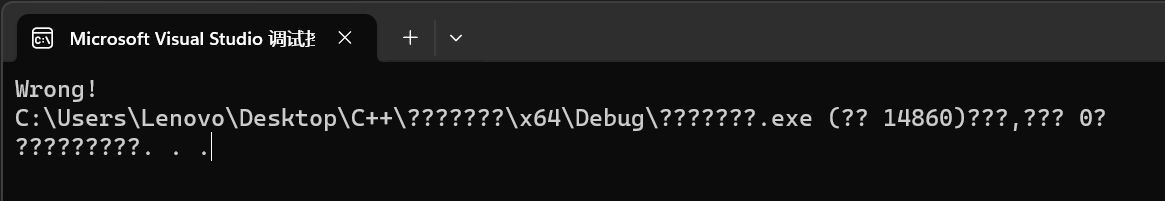
第三部分的结果：

****

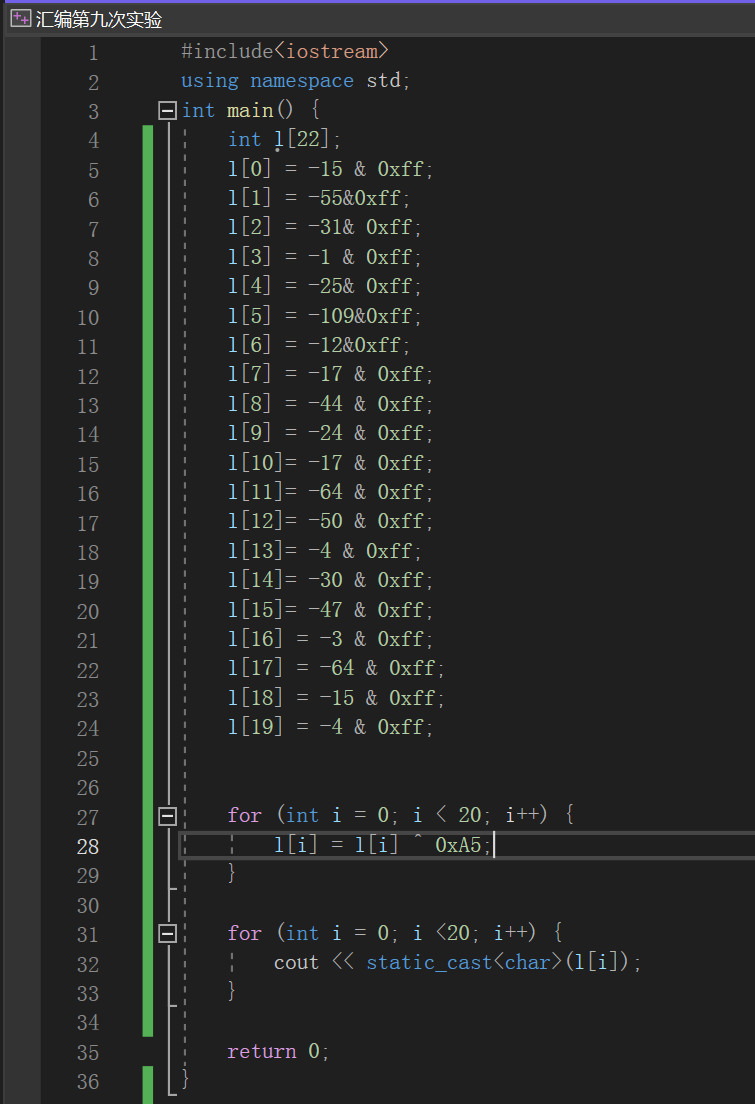
第四部分（while结构）的代码：



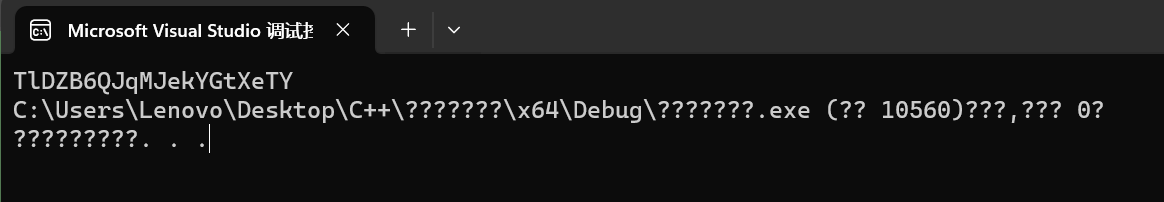
第四部分的结果：

****

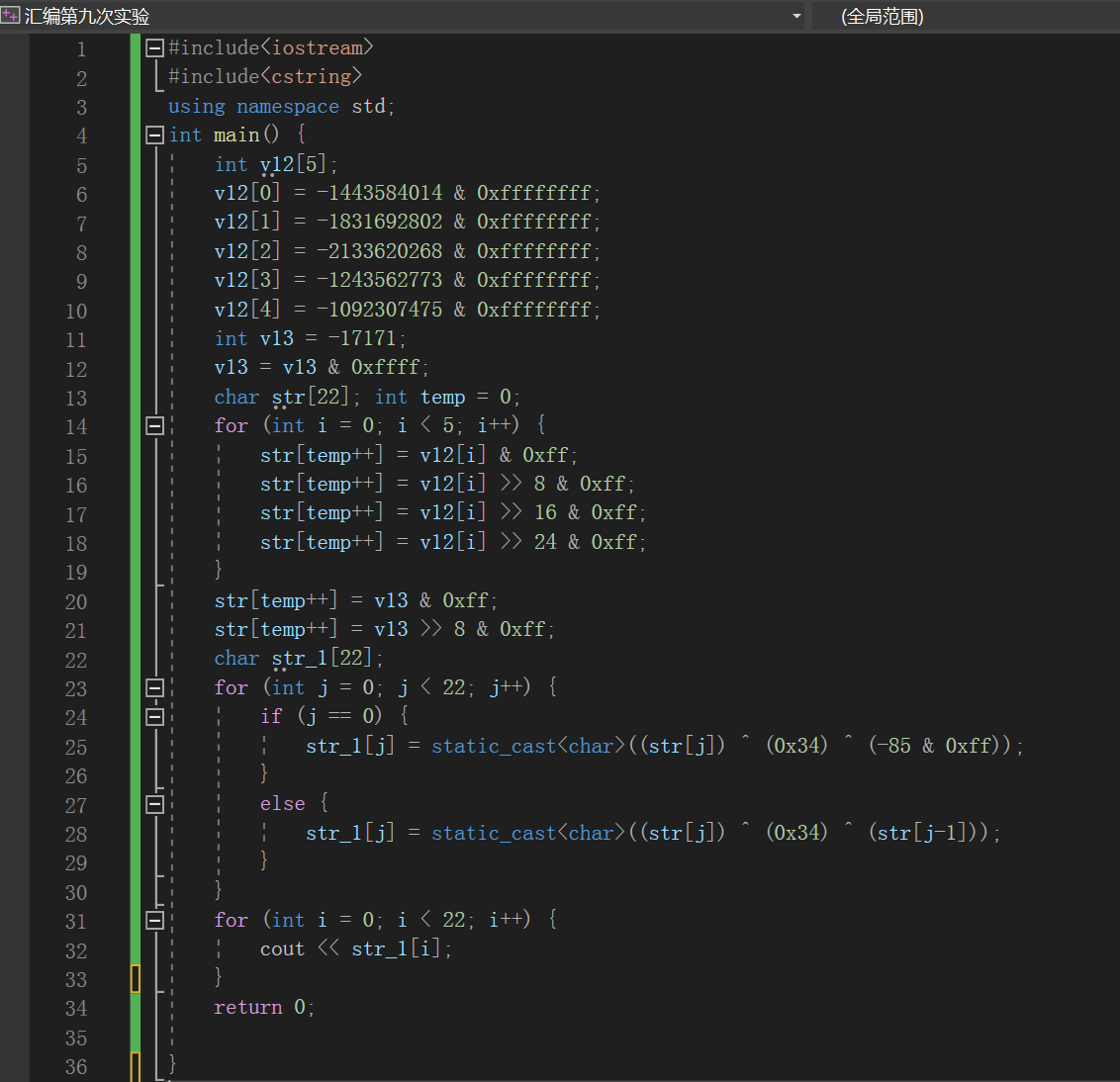
**最后的结果求解代码：**

****

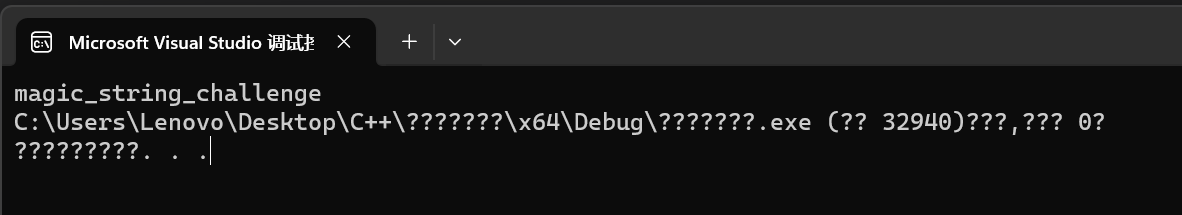
**求解结果：**

** 3.3.2 task4**

我们使用C++程序进行逆向的编写，求解出最后的结果。

****

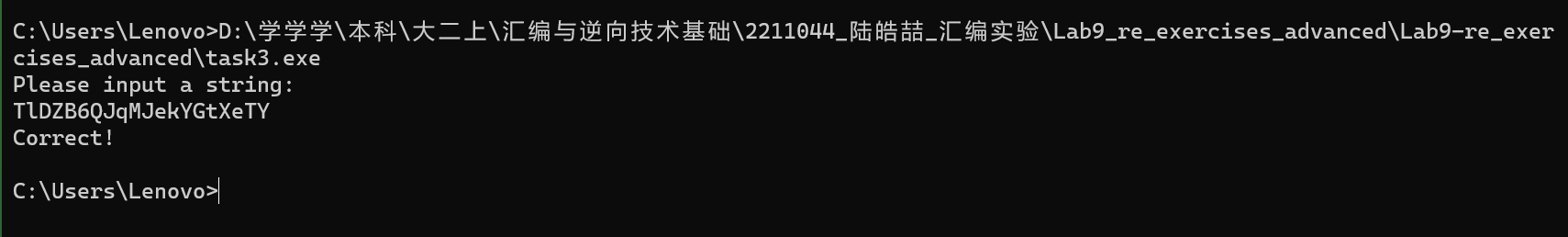
**最后得出结果：**

****

**3.4分别运行程序task3、task4，输入计算得到的字符串进行验证，获得“Correct”输出，提供截图。**

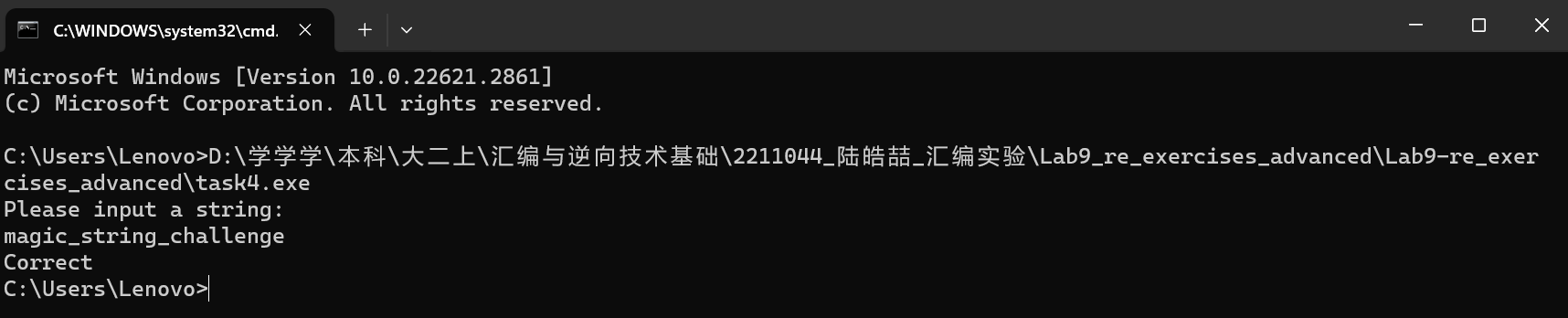
**3.4.1 task3**

如图，我们输入TlDZB6QjqMJekYGtXeTY，得到结果“Correct！”，实验成功！



**3.4.2 task4**

如图，我们输入上面得到的字符串“magic\_string\_challenge”,最后输出结果“Correct”，实验成功！

****

**四、总结与反思**

从本次实验中，我学到了更多的逆向分析的知识与方法，处理了两个带有for循环、while循环等复杂结构的逆向分析处理，对逆向分析的思路以及IDA的使用更加熟练了，希望之后我还能继续用自己的汇编知识去解决一些更难的问题。