

**实验报告**



**题目： 拆解二进制炸弹**

**班 级： 2021211304**

**学 号： 2021211035**

**姓 名： 赵一鸣**

**学 院： 计算机**

**2022年 10 月 19 日**

一、实验目的  
1.理解C语言程序的机器级表示。  
2.初步掌握GDB调试器的用法。  
3.阅读C编译器生成的x86-64机器代码，理解不同控制结构生成的基本指令模式，过程的实现。

1. 实验环境
2. Windows PowerShell（10.120.11.12）
3. Linux
4. Objdump命令反汇编
5. GDB调试工具
6. 。。。。。

三、实验内容

登录bupt1服务器，在home目录下可以找到Evil博士专门为你量身定制的一个bomb，当运行时，它会要求你输入一个字符串，如果正确，则进入下一关，继续要求你输入下一个字符串；否则，炸弹就会爆炸，输出一行提示信息并向计分服务器提交扣分信息。因此，本实验要求你必须通过反汇编和逆向工程对bomb执行文件进行分析，找到正确的字符串来解除这个的炸弹。

本实验通过要求使用课程所学知识拆除一个“binary bombs”来增强对程序的机器级表示、汇编语言、调试器和逆向工程等方面原理与技能的掌握。 “binary bombs”是一个Linux可执行程序，包含了5个阶段（或关卡）。炸弹运行的每个阶段要求你输入一个特定字符串，你的输入符合程序预期的输入，该阶段的炸弹就被拆除引信；否则炸弹“爆炸”，打印输出 “BOOM!!!”。炸弹的每个阶段考察了机器级程序语言的一个不同方面，难度逐级递增。

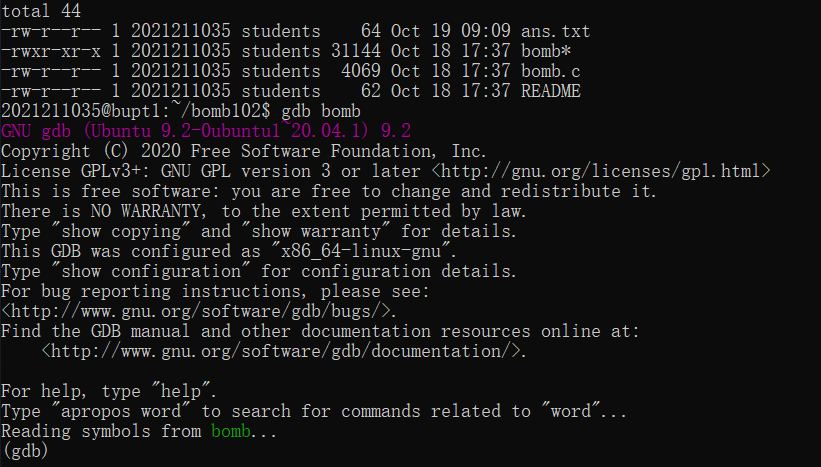
为完成二进制炸弹拆除任务，需要使用gdb调试器和objdump来反汇编bomb文件，可以单步跟踪调试每一阶段的机器代码，也可以阅读反汇编代码，从中理解每一汇编语言代码的行为或作用，进而设法推断拆除炸弹所需的目标字符串。实验2的具体内容见实验2说明。

四、实验步骤及实验分析

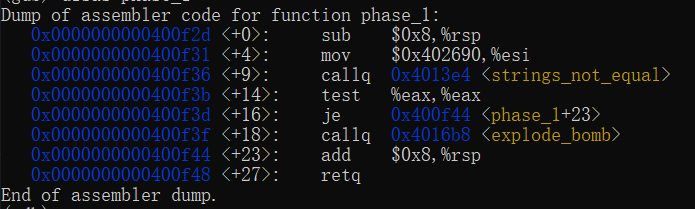
建议按照：准备工作、阶段1、阶段2、…等来组织内容

各阶段需要有操作步骤、运行截图、分析过程的内容

准备工作：使用tar指令解压炸弹压缩包，在gdb中加载可执行文件。



查看bomb.c得知，对输入的检测在phase\_n函数中进行。先对phase\_1反汇编。

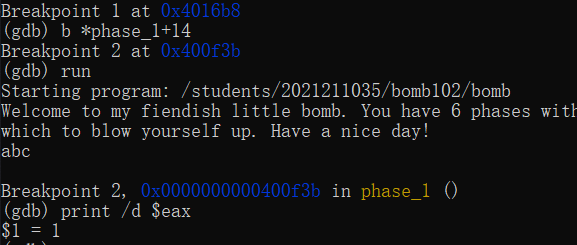


可见函数<explode\_bomb>,推测为输入错误炸弹引爆的函数，在explode\_bomb设置断点防爆方便调试。



阶段1:

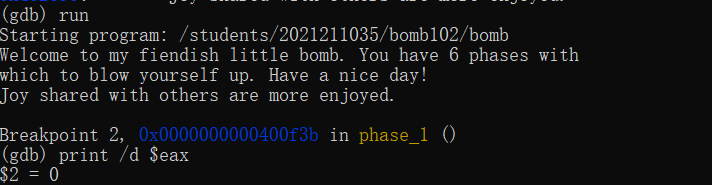
观察phase\_1的反汇编结果，发现调用了一个函数strings\_not\_equal，推测为字符串比较函数，第一个参数沿用传入phase\_1函数的参数，第二个参数来自内存地址0x402690。为监测该函数的功能，在test前设定断点，随意输入一个字符串，查看其返回值。



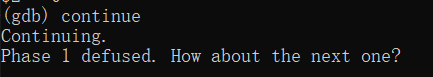
返回值为一，键入的输入明显是不正确的，说明这个函数在字符串比对成功时返回0，在下面的条件跳转中跳过爆炸函数。那么答案应该是内存地址0x402690开头的字符串，使用x指令查看。



得到明显规律的字符串，作为输入再次尝试，再查看比对函数返回值

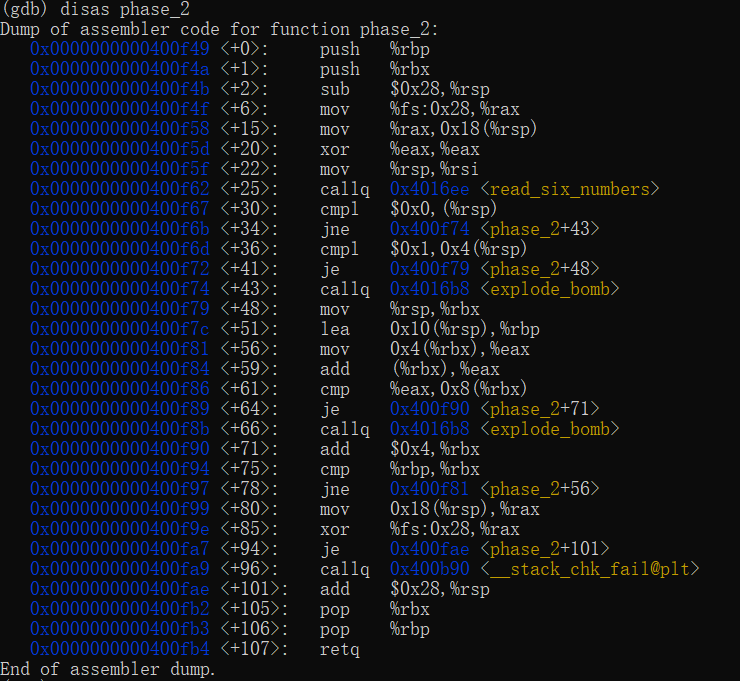


值为0，判断成功。继续运行通过阶段1。



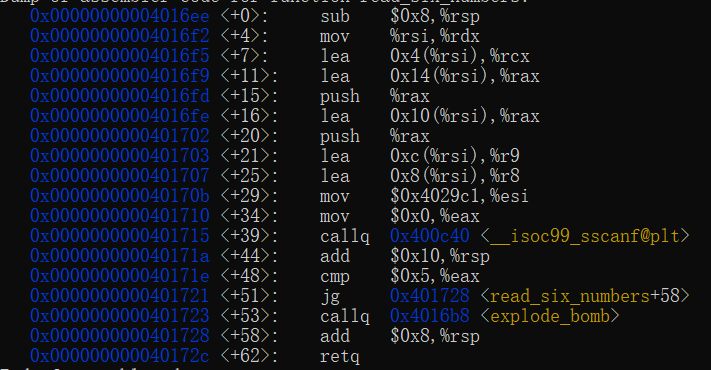
阶段2：

反汇编phase\_2函数



观察到<read\_six\_numbers>函数，推测功能是读取六个数。

反汇编查看



可以看到，函数将rsi的值分别加以0,4,8,12,16,20放到参数寄存器，调用scanff，推测从输入的串中读取六个整数，存到rsi为数组指针的数组中。

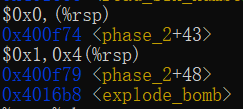
查看模式串对应的变量，其地址下的内容



证实是读取六个数字。

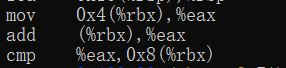
若返回值，即读入的数不足六个会爆炸。

观察到对rsp指向的位置比较0，rsp+4指向的位置为1。



数组应为{0,1，…..}否则爆炸。

观察到78行存在一个跳转到56行的指令，推测是循环，条件是rbp=rbx。rbp在循环内没有变化，着重观察rbp。



显然，这里把rbx指向的第一个元素和下一个元素相加与第三个元素比较,不同则爆炸。

以此检测输入的正确性

循环结束后rbx会自增4。

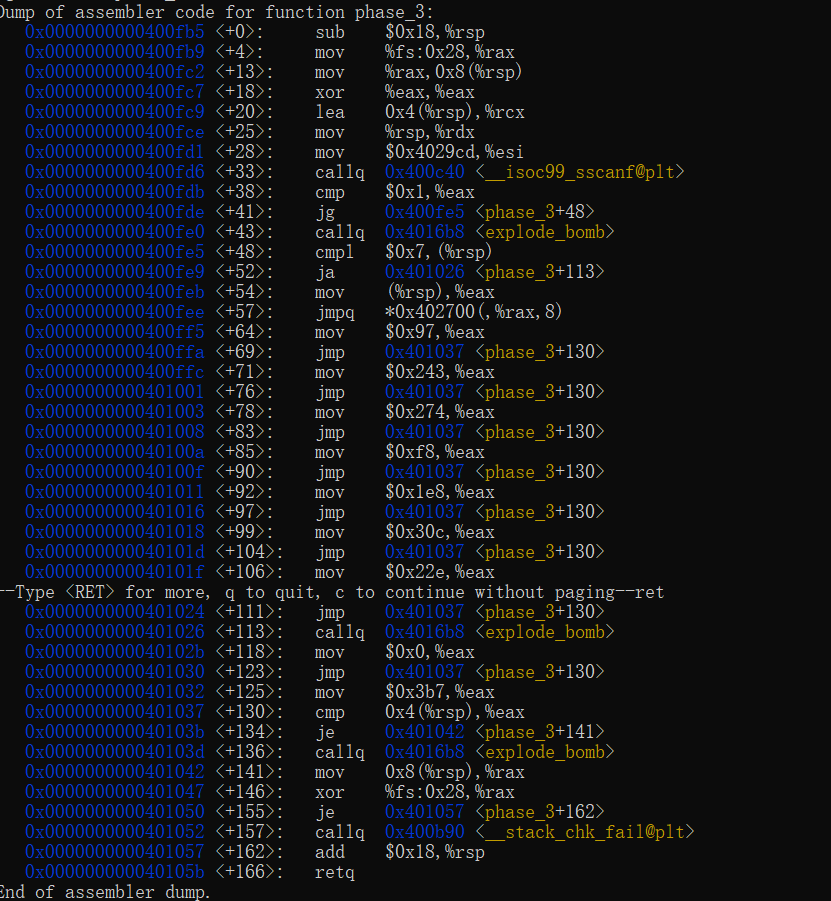
当rbx等于rbp时，数组的第六位被比较。

可以看出，输入应是初始项为0，1的斐波那契数列：

{0,1,1,2,3,5}。

阶段3：

反汇编phase\_3



熟悉的scanf，先查看模式串的内容：



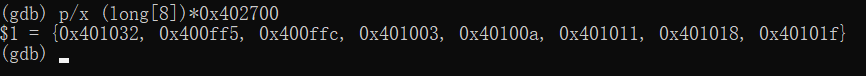
可见输入是两个数

下面检测输入个数，不足2就爆炸。

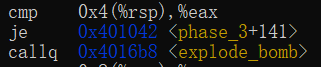
设输入为a，b，下面将a与7比较，大于7就会爆炸。

观察到存在一个变址跳转，以及很多无条件跳转，推测是switch结构。

着重观察变址跳转：



Switch语句的跳转表。

这里比较switc

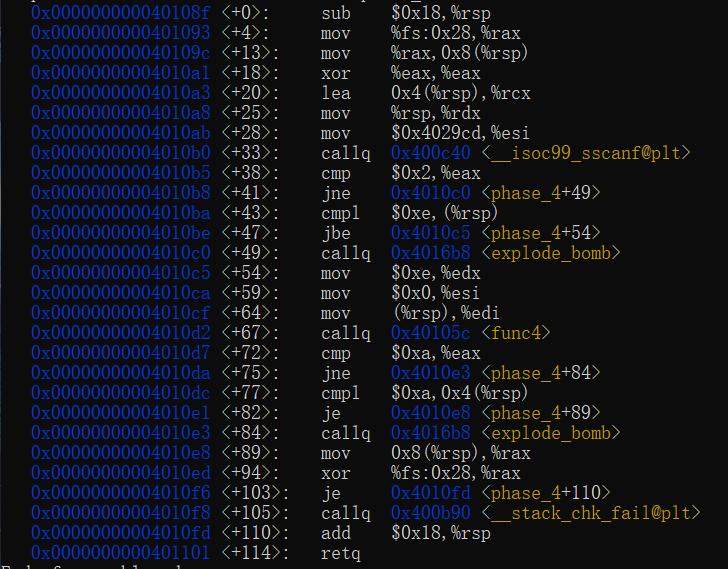
h中赋值与b。

所以任选a<=7，b取值对着跳转表找即可。

其中的一对答案：2 579。

阶段4：

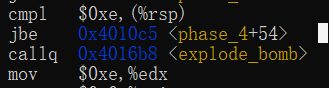
对phase\_4反汇编：



查看输入的模式串



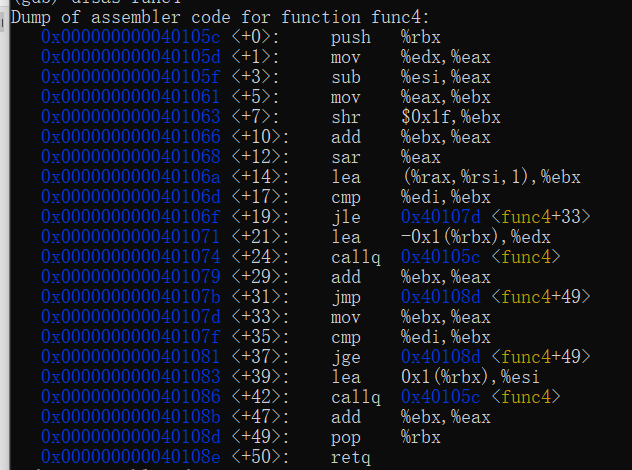
两个输入，设为a,b。



第一个输入不能大于14

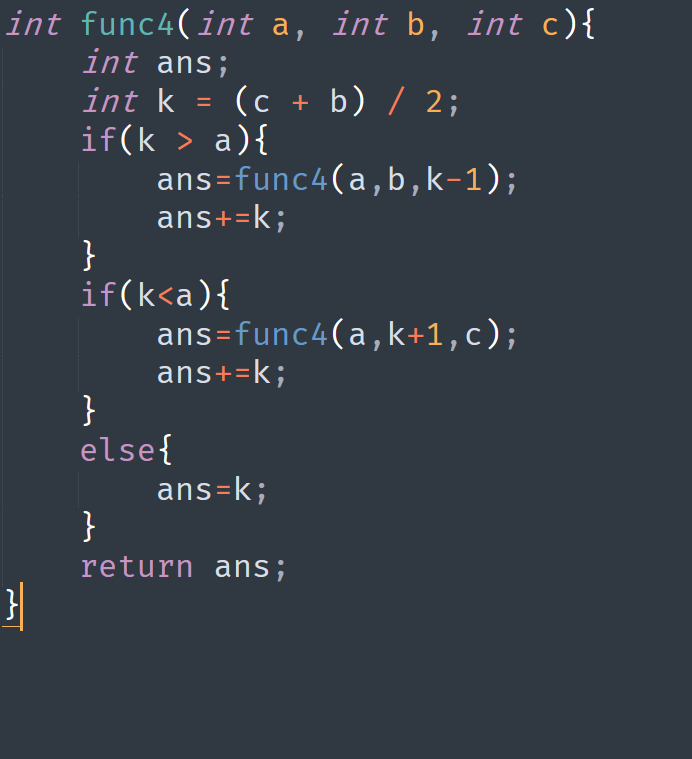
下方调用了一个函数func4,参数为a,0,14。

反汇编func4



观察到有嵌套调用，应该是递归函数。

人脑反编译后：



可以看出，这是在b和c之间查找a的二分匹配，并返回中值的累加。

在返回后，将ans和10比较，不等就爆炸。

第一次调用函数时，ans+7

第二层时，可能累加3或11,3+7=10，因此a应该小于7。

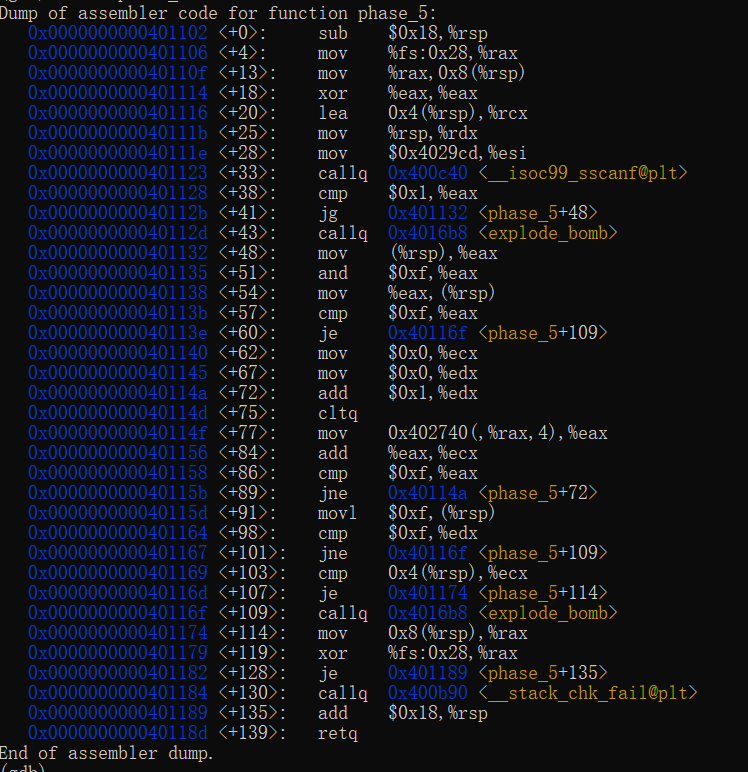
第二层时，应直接返回3，因此a=3。

函数phase\_4的结尾，将b与10比较，不相等就爆炸，因此b应该等于10.

综上所述，答案为3 10.

阶段5：

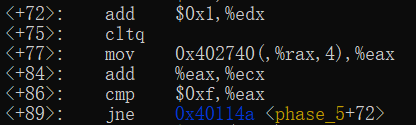
反汇编phase\_5：



查看输入模式串



观察到一个循环体



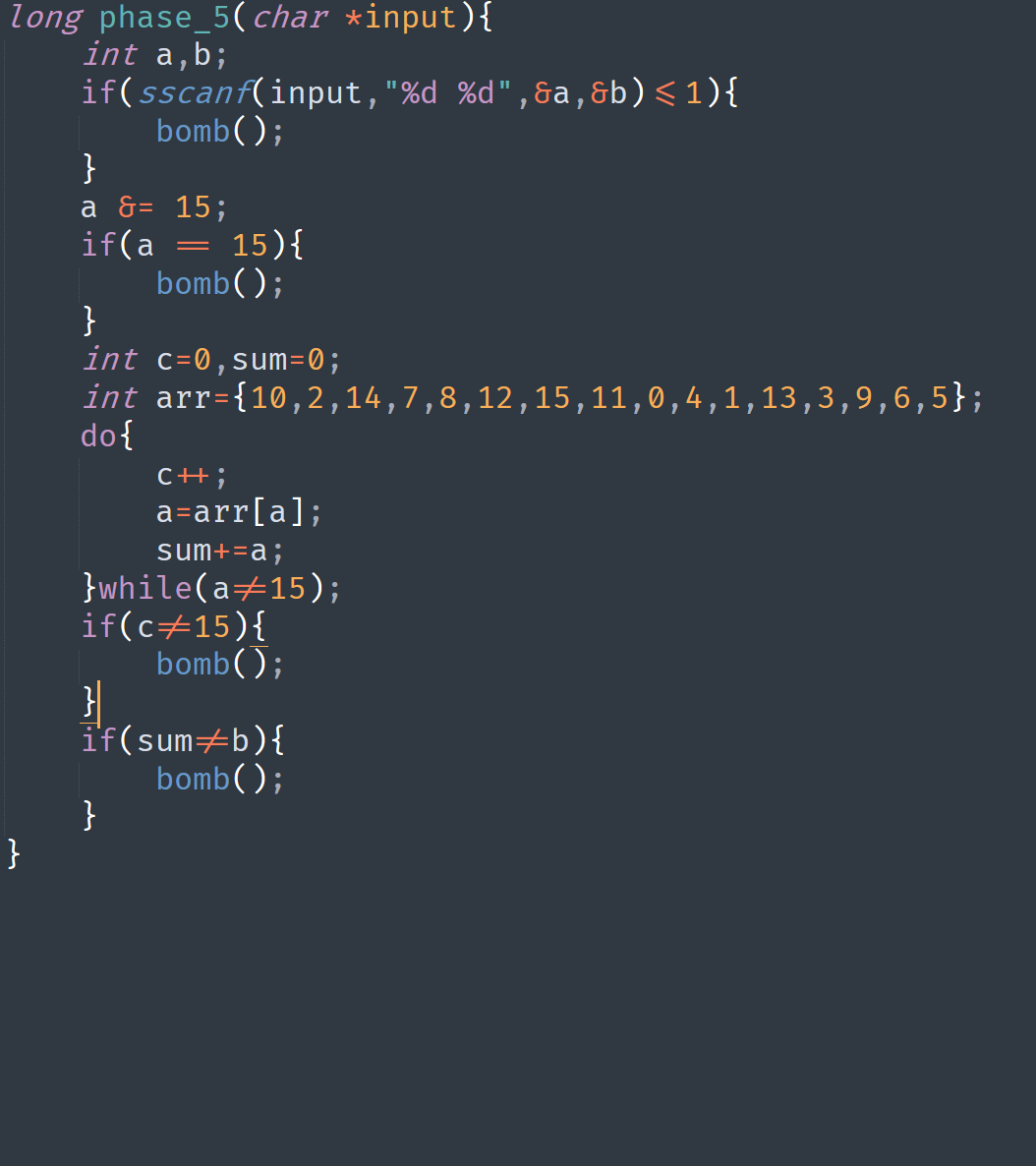
可以看到rax不断以自身为变量变址寻址后赋值，类似i=a[i];

Rax在ecx中有累加。

查看0x402740指向的数组：

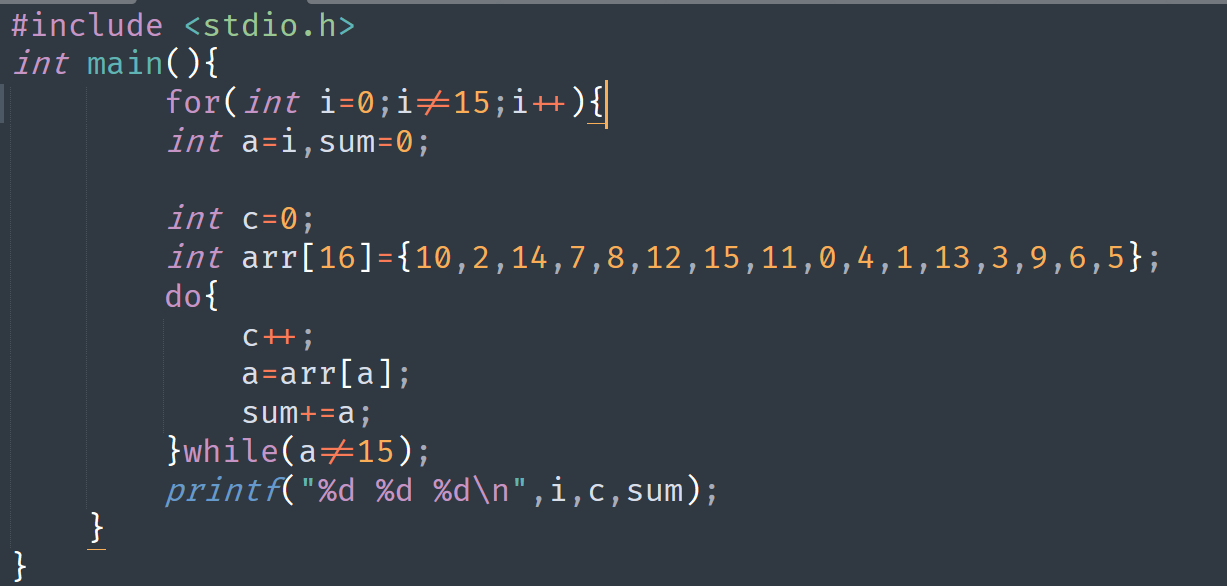


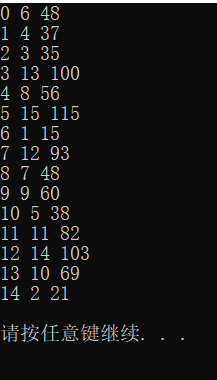
结合下文的条件判断，反编译如下：



a迭代15次，终值为15，b应等于a的累加。

设计程序找出满足条件的a。





可以看到 5 115为满足要求的a，b输入。

五、总结体会

总结心得（包括实验过程中遇到的问题、如何解决的、过关或挫败的感受、实验投入的时间和精力、意见和建议等）

问题： 刚开始不会输出数组，百度找到的方法

不是很分得清存放参数的寄存器，用多了熟悉多了。

时间： 平均一个阶段十分钟，看懂表述内容后再逆推输入就简单了。

建议： 可以加入其他经典算法，了解其汇编表示。