同济大学 TONGJI UNIVERSITY

《高级语言程序设计》实验报告
报告名称：VS2022调试工具的使用与总结

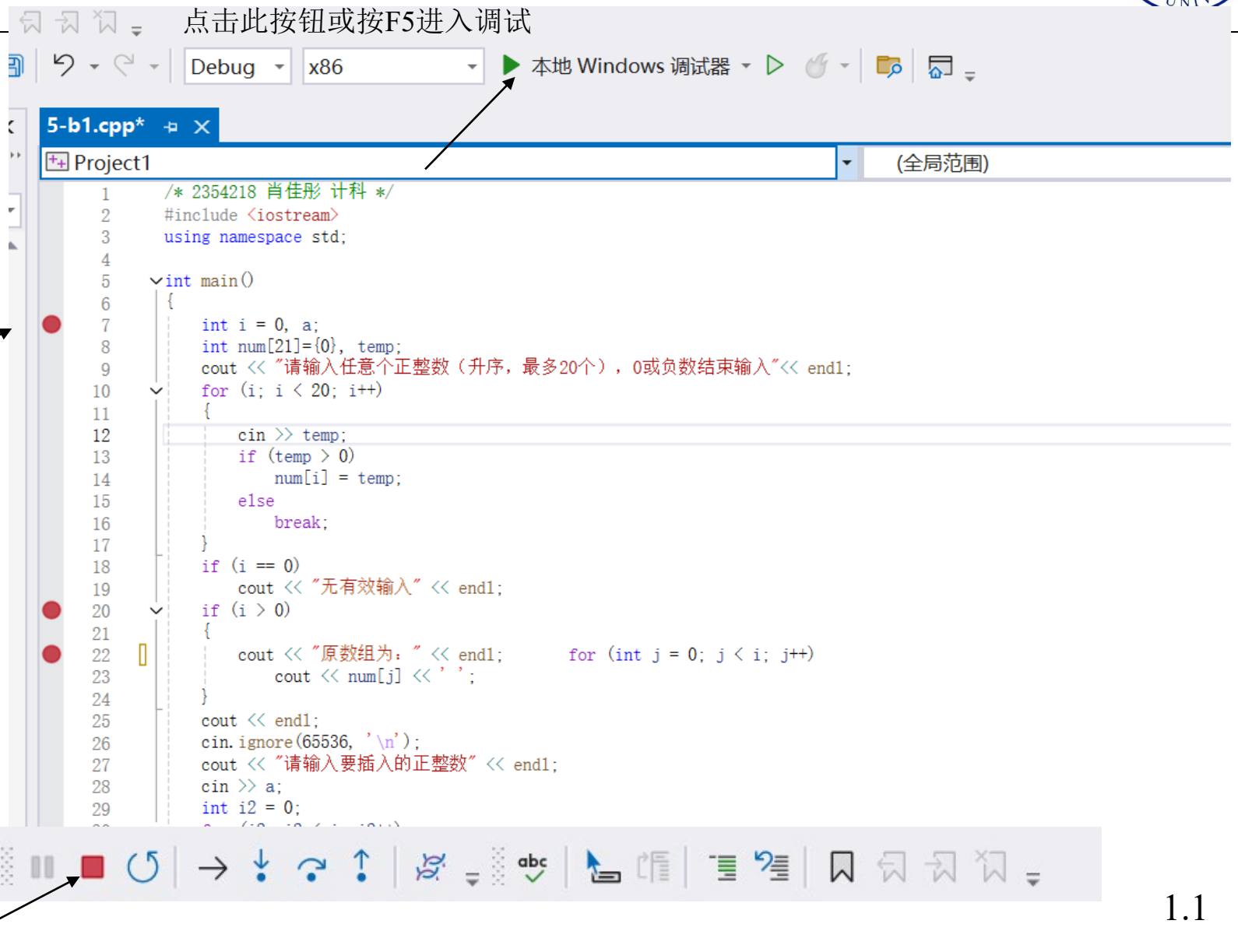
班级：计算机科学与技术

学号：2354218

姓名：肖佳彤

完成日期：2024年12月8日

1.1 开始/结束调试与配置

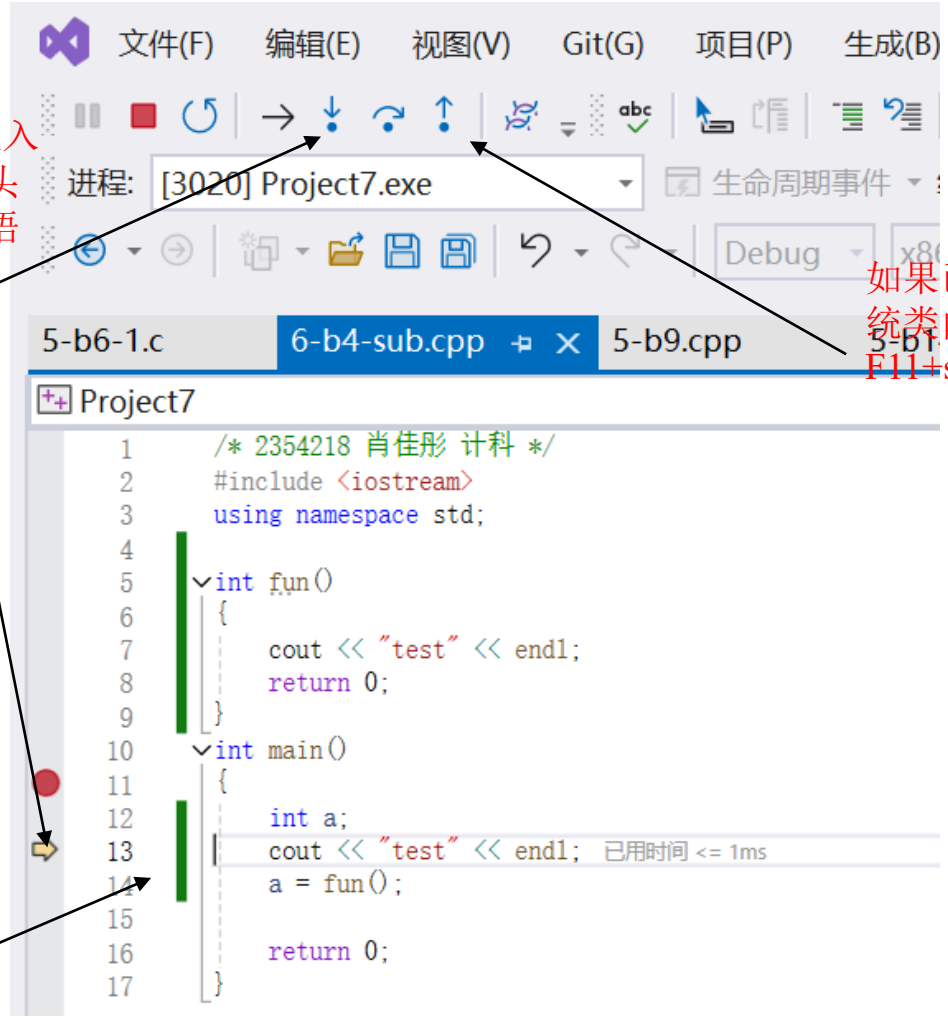


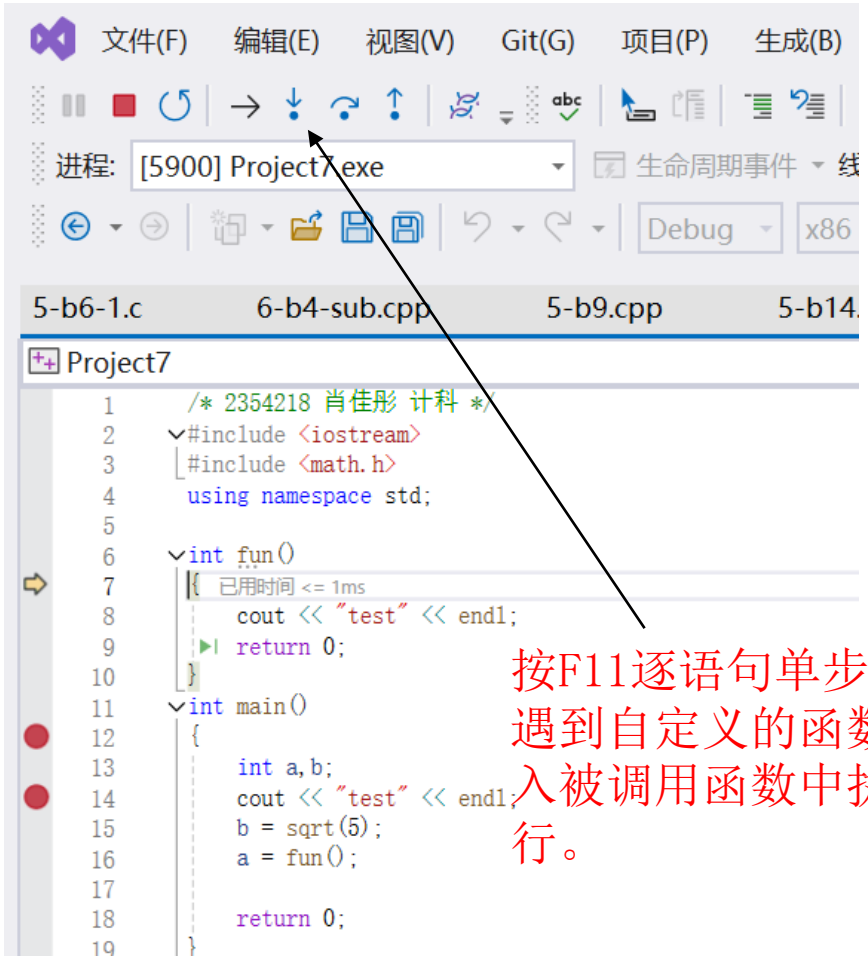

```
/* 2354218 肖佳彤 计科 */
#include <iostream>
using namespace std;

int main()
{
    int i = 0, a;
    int num[21]={0}, temp;
    cout << "请输入任意个正整数（升序，最多20个），0或负数结束输入"<< endl;
    for (i; i < 20; i++)
    {
        cin >> temp;
        if (temp > 0)
            num[i] = temp;
        else
            break;
    }
    if (i == 0)
        cout << "无有效输入" << endl;
    if (i > 0)
    {
        cout << "原数组为：" << endl;       for (int j = 0; j < i; j++)
            cout << num[j] << ' ';
    }
    cout << endl;
    cin.ignore(65536, '\n');
    cout << "请输入要插入的正整数" << endl;
    cin >> a;
    int i2 = 0;
```

按此按钮或者F11进入逐语句调试，小箭头指向为下一步执行语句

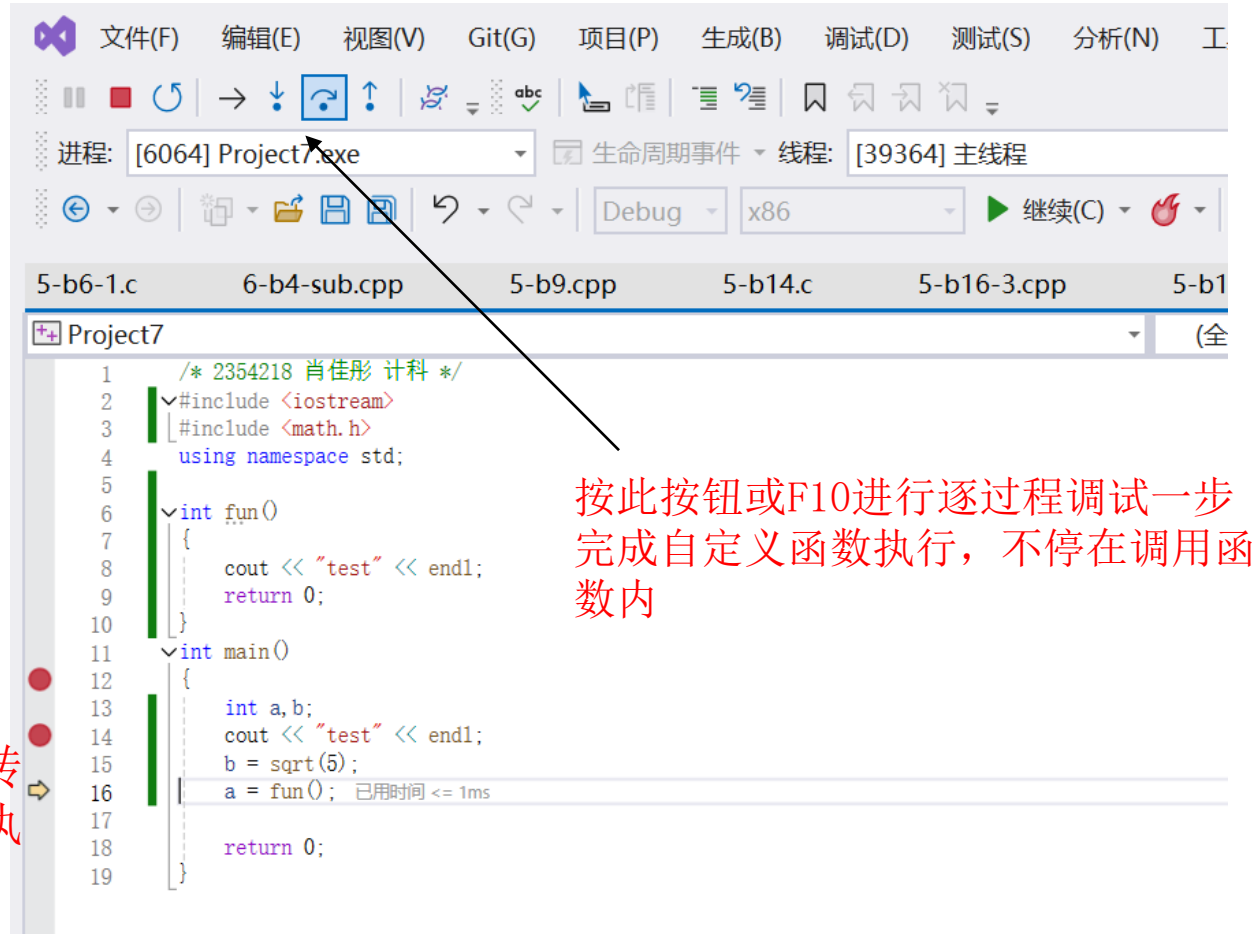
如果已经进入系统函数，系统类内部按这个按钮或按F11+shift跳出函数

非自定义的系统函数/系统类的视为一个完整的语句，单步执行默认不会进入其内部。





按F11逐语句单步执行时，遇到自定义的函数，会跳转入被调用函数中执行单步执行。



按此按钮或F10进行逐过程调试一步完成自定义函数执行，不停在调用函数内

2.查看各种生存期/作用域变量

2.1,2.2 形参/自动变量与静态局部变量

局部变量栏中显示变量的变化情况。b为形参/自动变量，c为静态局部变量，当调试到该形参/自动变量和静态局部变量时所在函数体内才能观察到变量的值和其变化，跳出该函数则不能观测到。

```
1   /* 2354218 肖佳彤 计科 */
2   #include <iostream>
3   #include <math.h>
4   using namespace std;
5
6   void fun(int b)
7   {
8       cout << "test" << endl;
9       static int c = 5;
10      c++;
11      b++;
12  }  已用时间 <= 1ms
13  int main()
14  {
15      int a=5;
16
17      fun(a);
18
19      return 0;
20  }
```

% 未找到相关问题

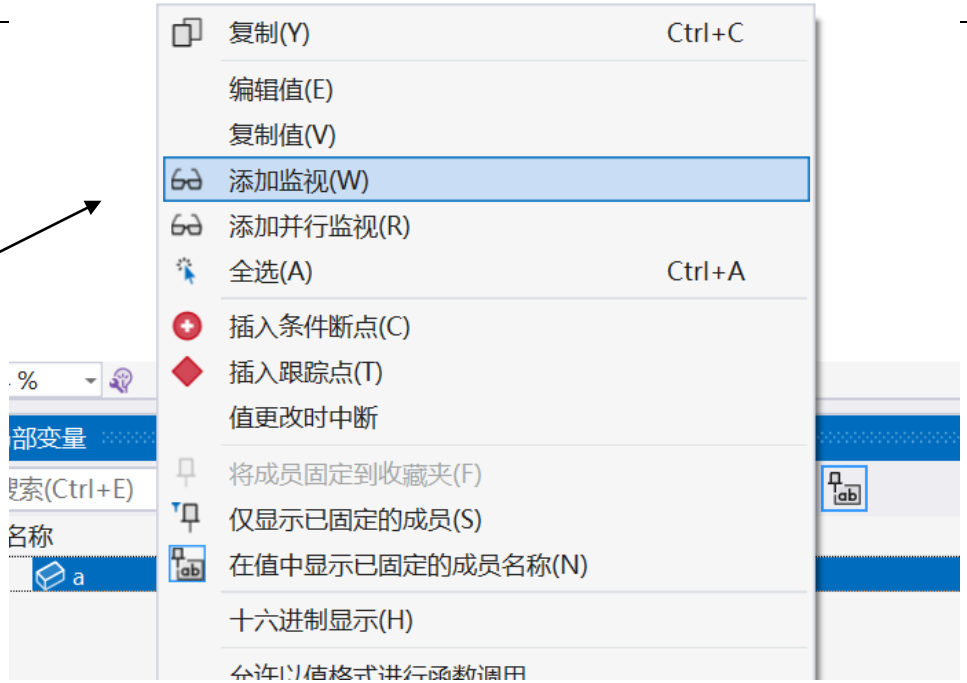
部变量

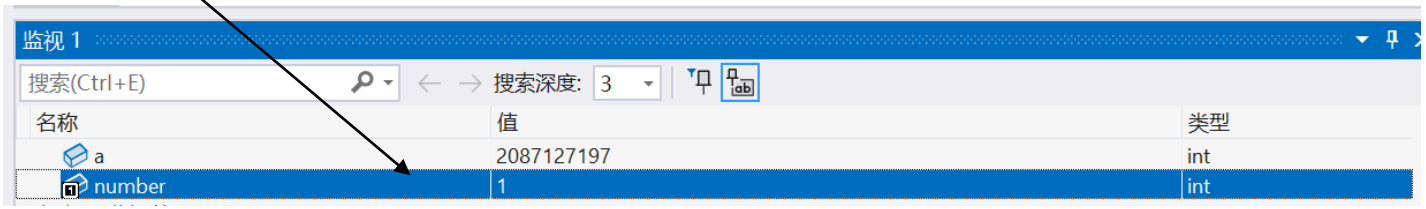
索(Ctrl+E) 搜索深度: 3

名称	值	类型
b	6	int
c	6	int

2.3 静态全局变量

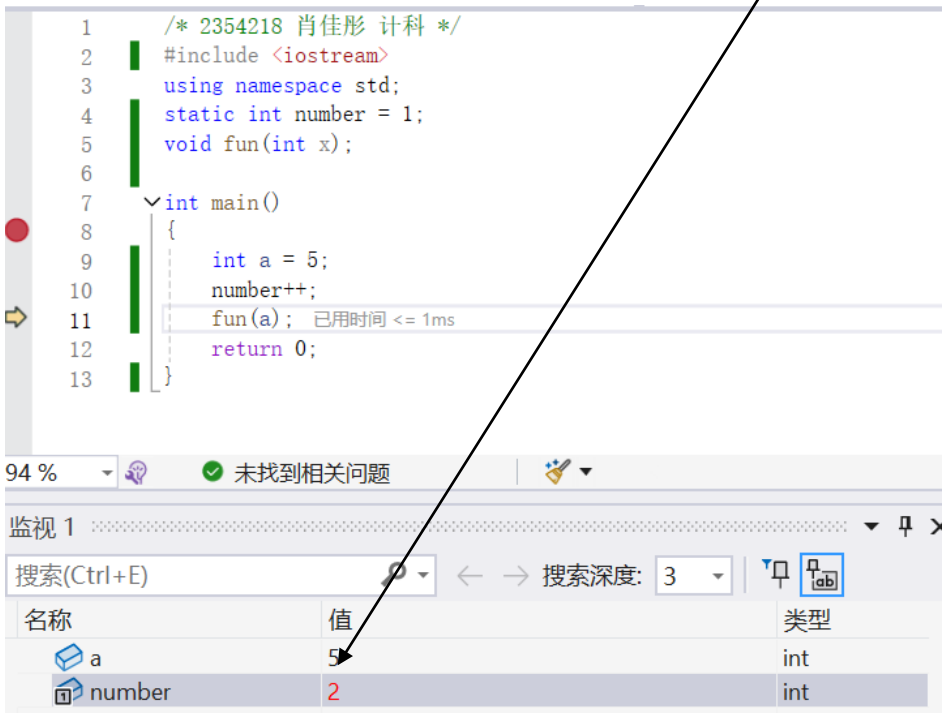
全局变量需要
手动添加监视，
输入变量名

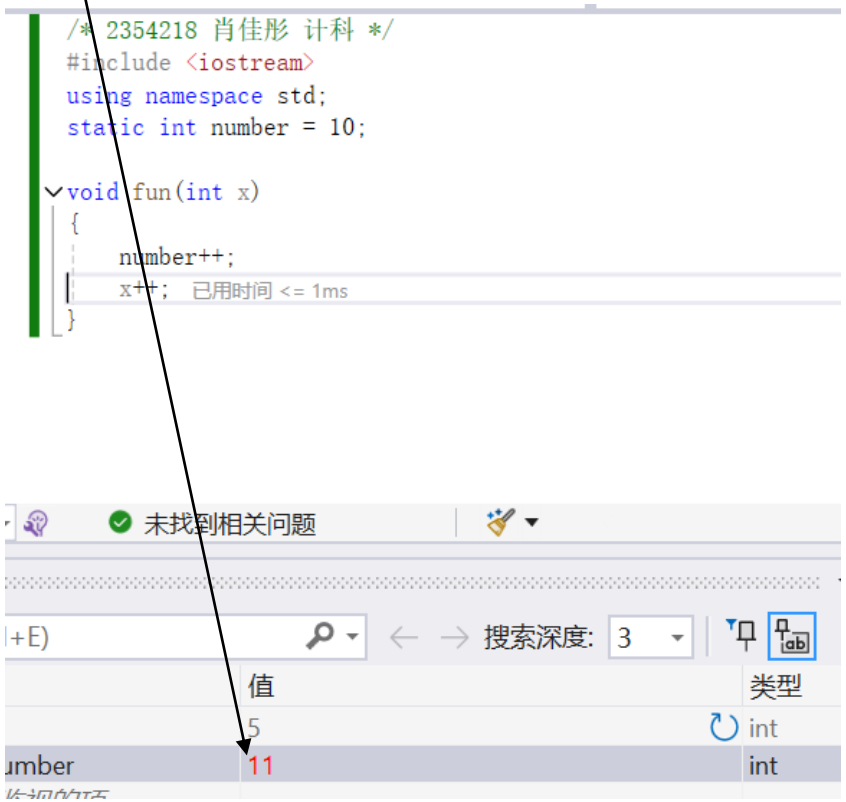




2.3 静态全局变量

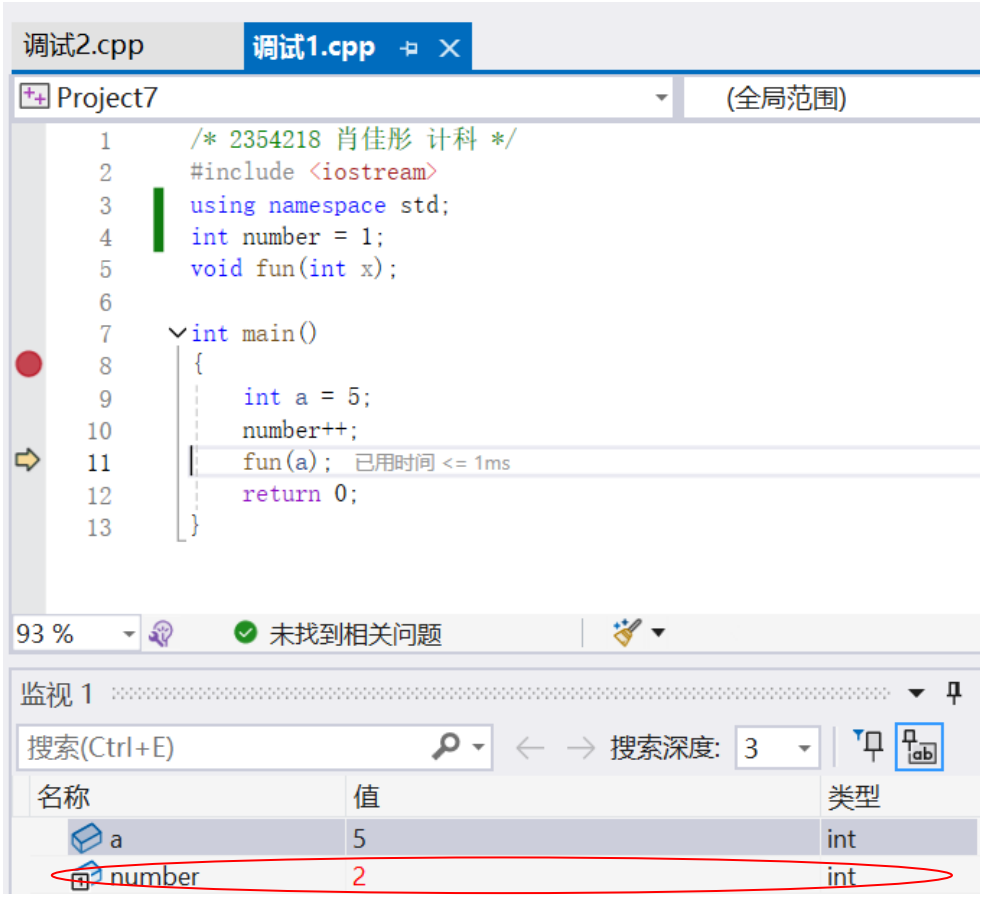
调试1和调试2中命名了相同变量，监视的number值为各自程序内的静态全局变量的值

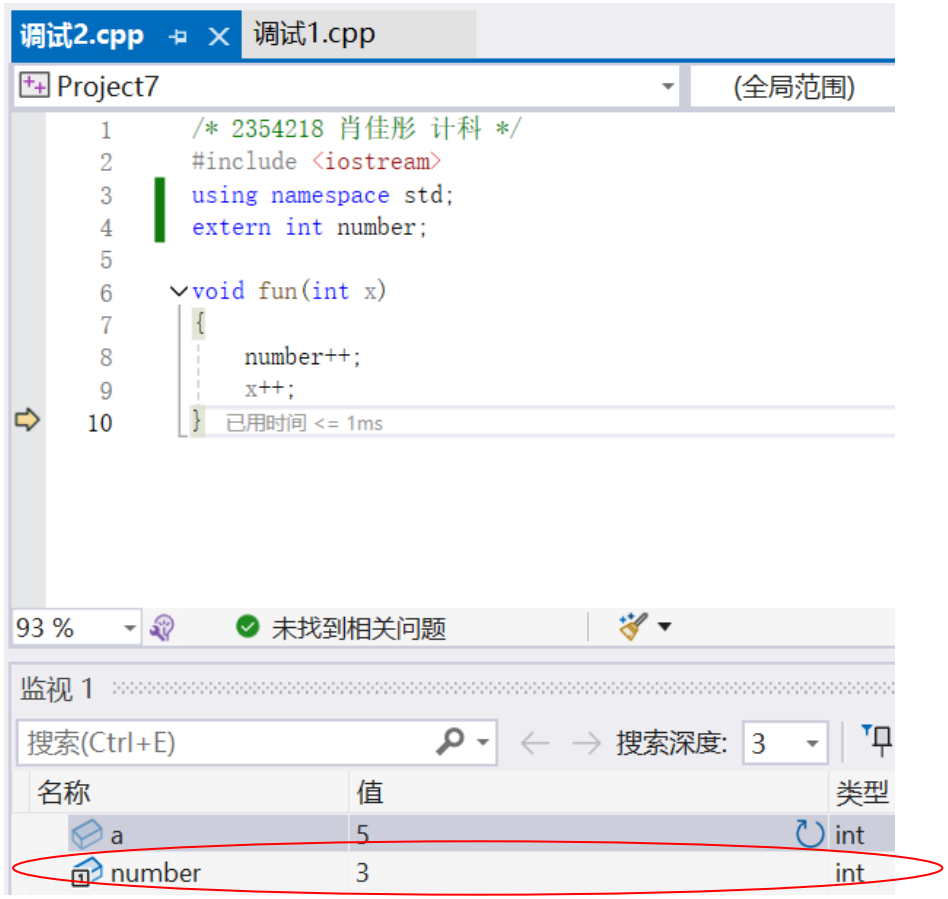




2.4 外部全局变量

同样全局变量需要手动添加监视，添加方法与2.3一致。当一个cpp 有定义、另一个cpp 中有extern 说明时，在下方的监视窗口中的number 是同一个。

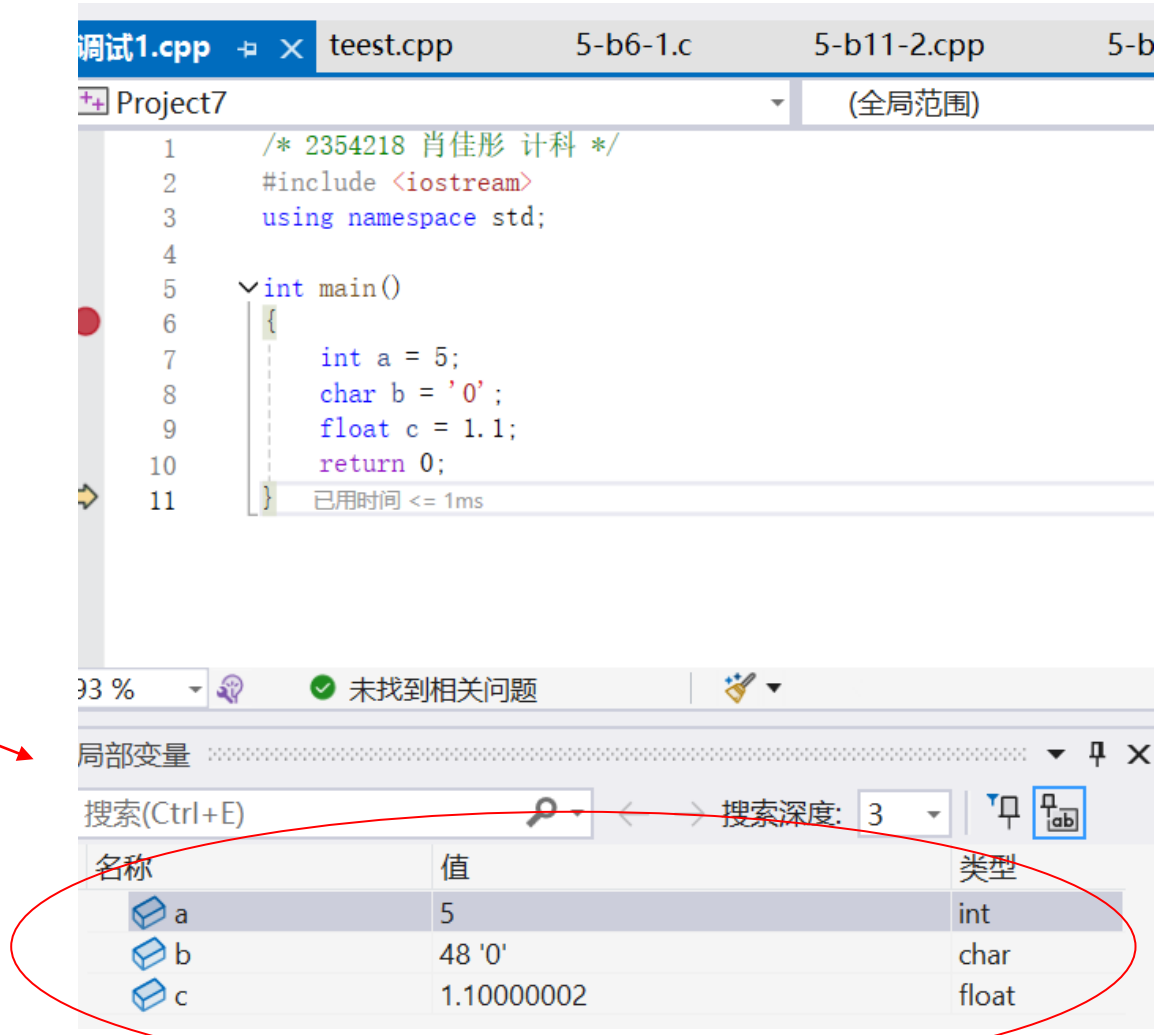




3.查看不同类型的变量

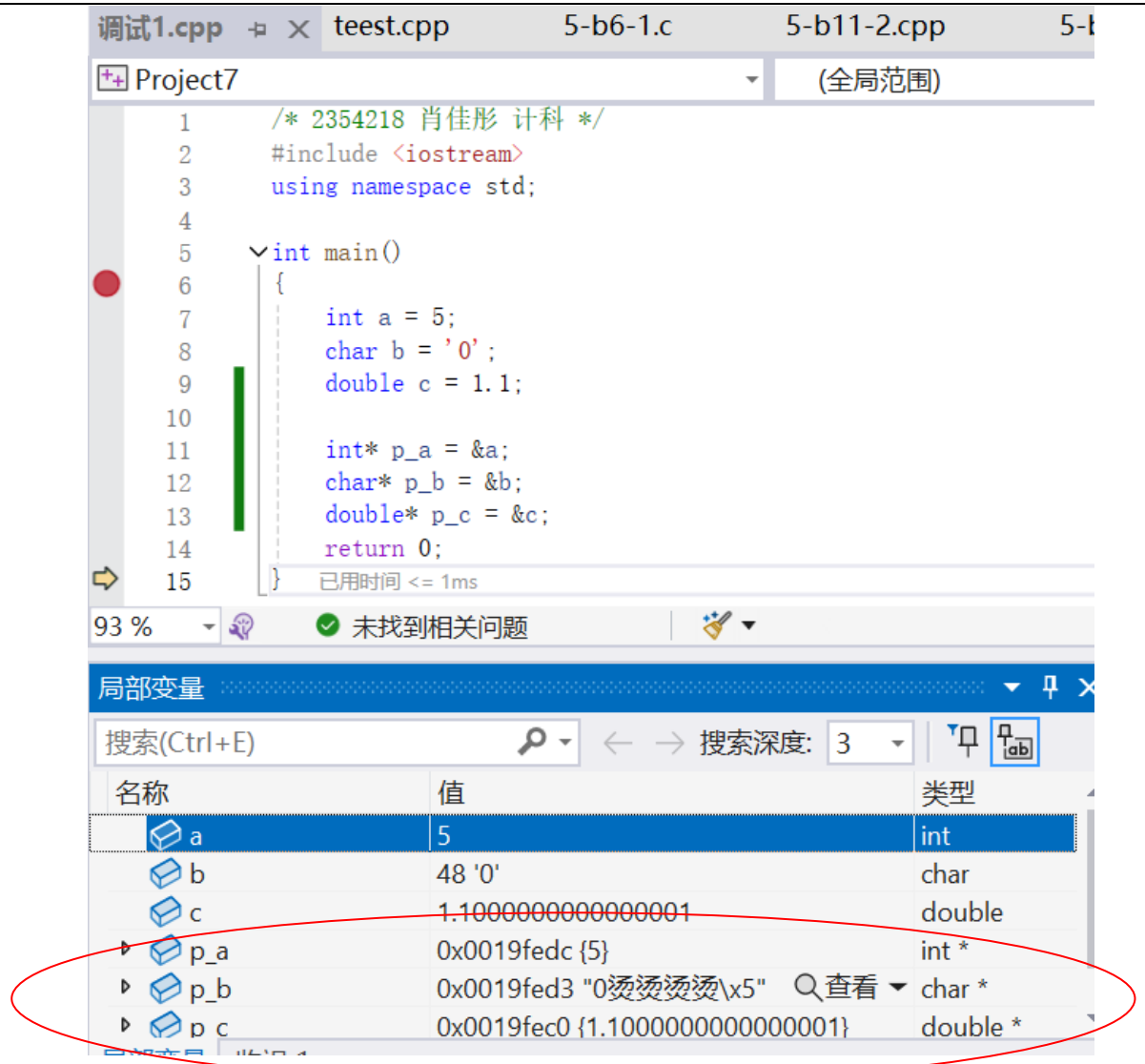
3.1 简单变量

对于**简单类型**的变量，在下方的窗口内直接查看其显示的值。



3.2 指向简单变量的指针

对于指向简单类型的变量的指针，在下方的窗口内查看地址。{}内显示的是其所指向地址位置的值；
如果还未访问到或者越界访问，则{}内为???，
如果访问的是char型变量，则遇到尾零截止。

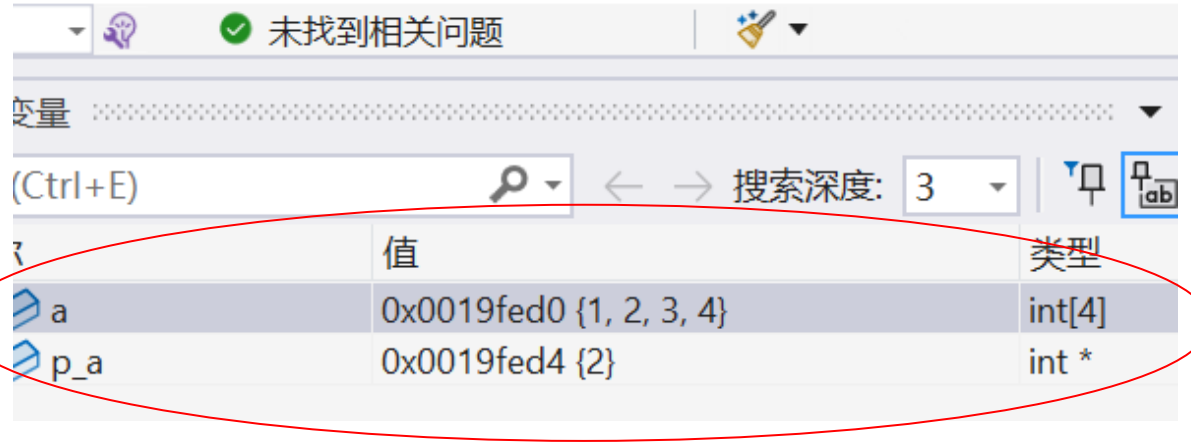


3.3,3.4 一维数组及其指针

对于一维数组,下方的窗口内会显示其首地址值， {}内显示一维数组内的所有值

对于指向一维数组的的指针，窗口内显示该数组的首地址和数组内的第一个元素值；通过对指针变量进行操作，可以观测到指针指向的数组内的不同元素。
如果访问越界，则其显示的值不可信。

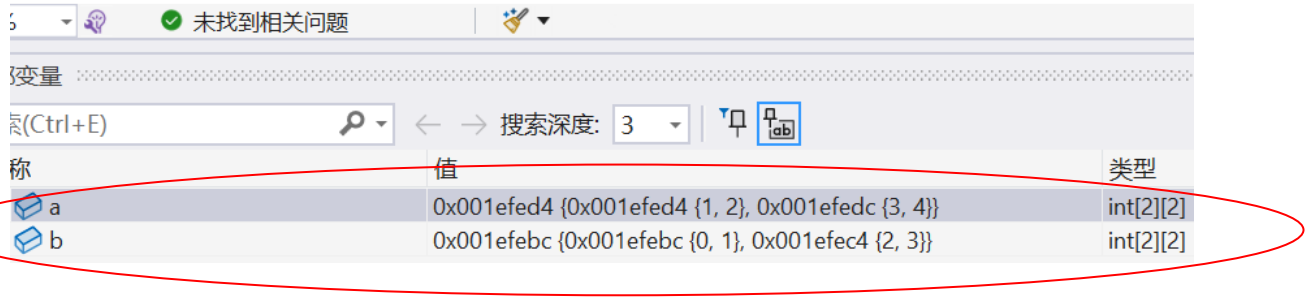
```
1   /* 2354218 肖佳彤 计科 */
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       int a[] = {1, 2, 3, 4};
8       int* p_a = a;
9       p_a++;
10      return 0;
11  }   已用时间 <= 1ms
```



3.5 二维数组

对于**二维数组**，窗口内会显示其首地址和值，
并且在随后的{}内按照前文一维数组所示的格式显示所有的一维数组的首地址和值。

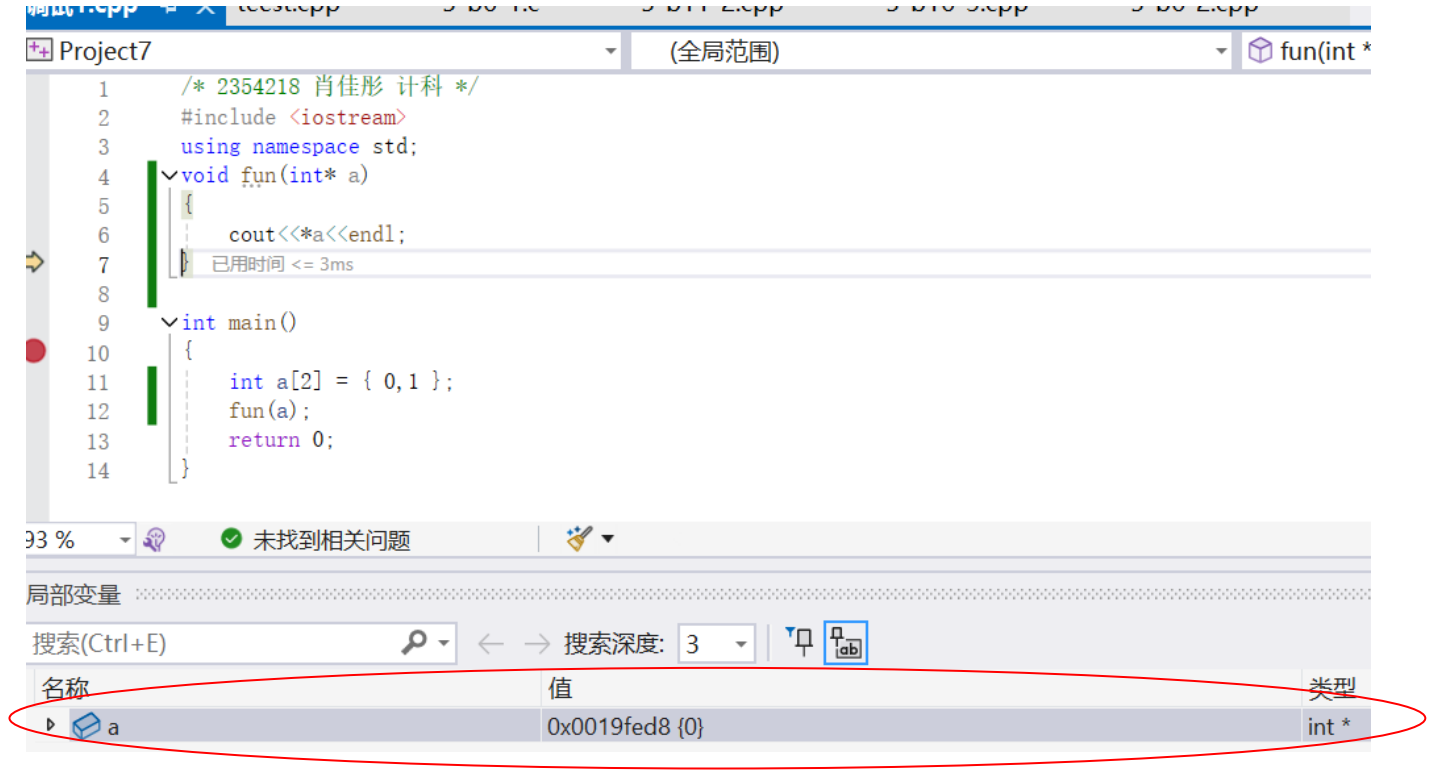
```
1   /* 2354218 肖佳彤 计科 */
2   #include <iostream>
3   using namespace std;
4
5   int main()
6   {
7       int a[2][2] = { {1,2},{3,4} };
8       int b[][2] = { 0,1,2,3 };
9       return 0;
10  }   已用时间 <= 1ms
```



3.6实参是一维数组名，形参是指针

实参是一维数组名，形参是指针

在下方的窗口内显示数组首元素的地址值和首元素的值。



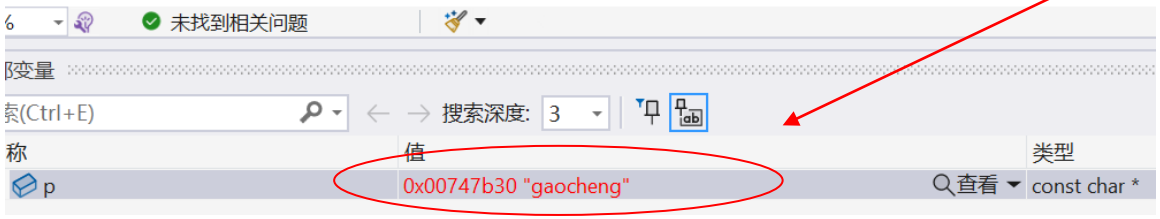
3.7 指向字符串常量的指针变量

```
1  /* 2354218 肖佳彤 计科 */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      char str[] = "gaocheng";
7      char* p=str;
8      return 0;  已用时间 <= 1ms
9  }
```

窗口中显示其字符串常量的首地址，fed后显示整个字符串常量的具体的值。


```
1  /* 2354218 肖佳彤 计科 */
2  #include <iostream>
3  using namespace std;
4  int main()
5  {
6      const char* p= "gaocheng";
7
8      return 0;  已用时间 <= 1ms
9  }
```

能看到无名字符串常量的值



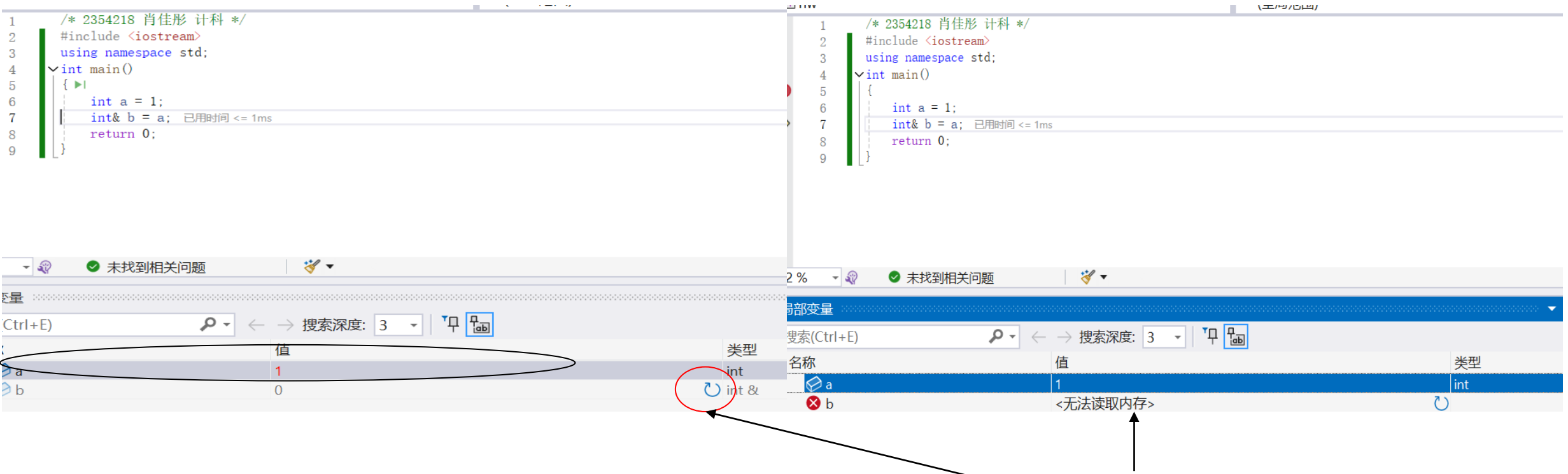
3.8 引用

```
/* 2354218 肖佳彤 计科 */
#include <iostream>
using namespace std;
int main()
{
    int a = 1;
    int& b = a;  已用时间 <= 1ms
    return 0;
}
```

未找到相关问题

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
a	1	int
b	0	int &



对引用来说，当被引用变量被声明后，
引用显示为0

点击立即计算后显示<无法读取内存>

3.8 引用

执行引用赋值语句后，引用的值直接赋被引用变量的值

```
1   /* 2354218 肖佳彤 计科 */
2   #include <iostream>
3   using namespace std;
4   int main()
5   {
6       int a = 1;
7       int& b = a;
8       return 0;  已用时间 <= 1ms
9   }
```

未找到相关问题

变量

(Ctrl+E) 搜索深度: 3

称	值	类型
a	1	int
b	1	int &

引用是所用的赋值变量的别名，而指针的值是地址。
因此引用调试时不显示地址，赋值前无法读取内存，指针显示地址和指向的值

3.9 指针越界访问

指针发生越界访问，显示的值不可信