



## 高级语言程序设计

### VS2019 调试工具的使用

装

订

线

作 者 姓 名：\_\_\_\_\_王灏廷\_\_\_\_\_

学 号：\_\_\_\_\_1953609\_\_\_\_\_

学院、 专业：\_\_\_\_\_软件学院 软件工程\_\_\_\_\_

同济大学

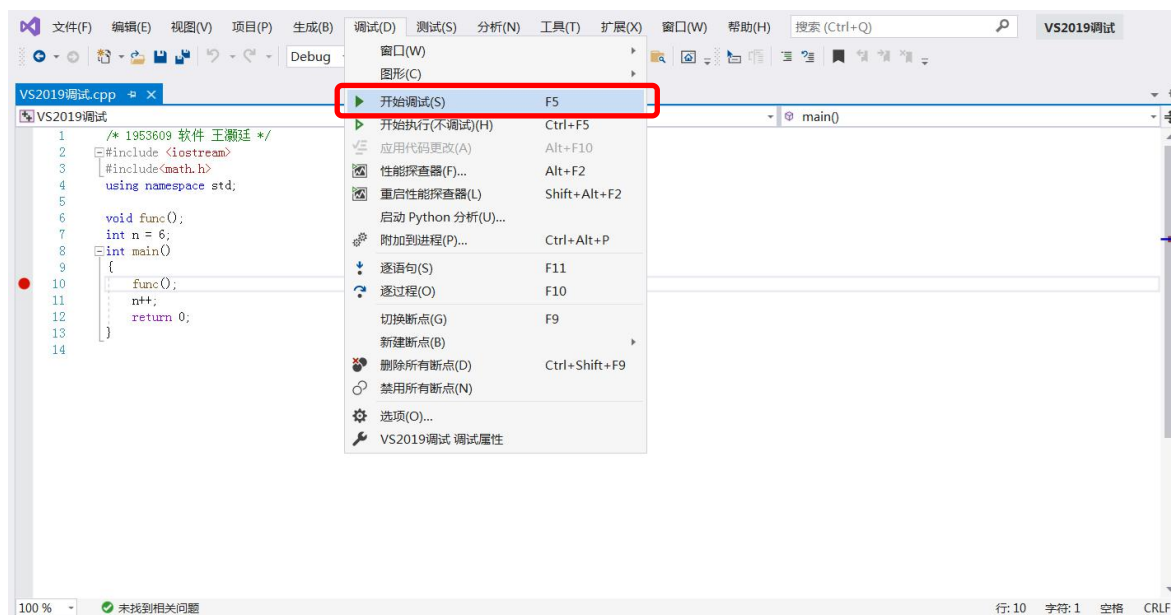
Tongji University

二〇二一年六月

## 1.VS2019 下调试工具的基本使用方法

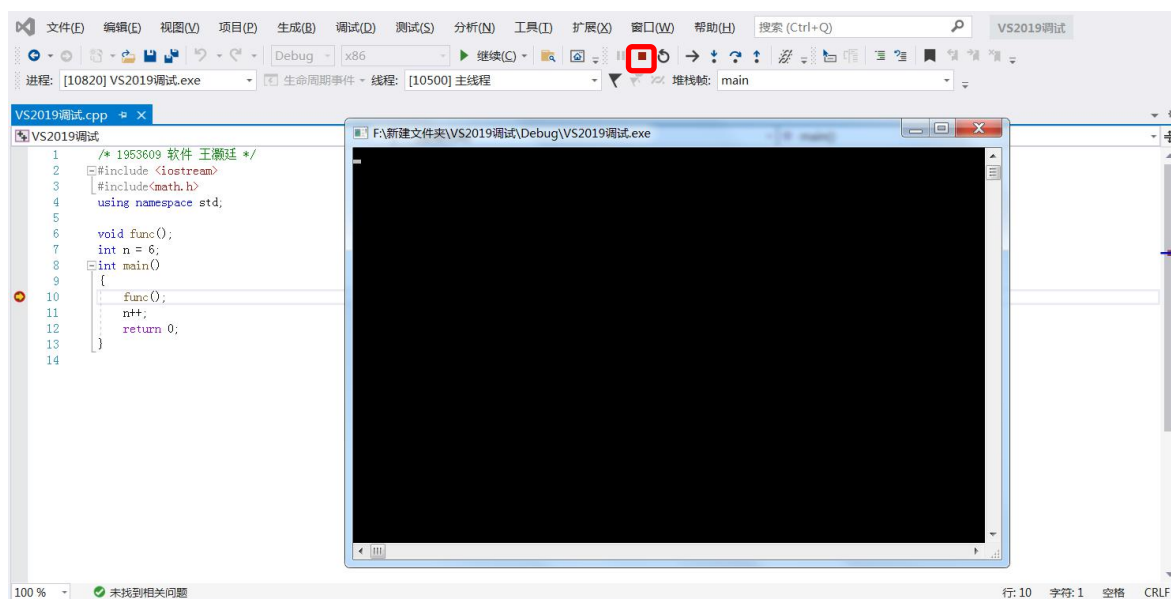
### 1.1 如何开始调试？如何结束调试？

#### 1.1.1 开始调试



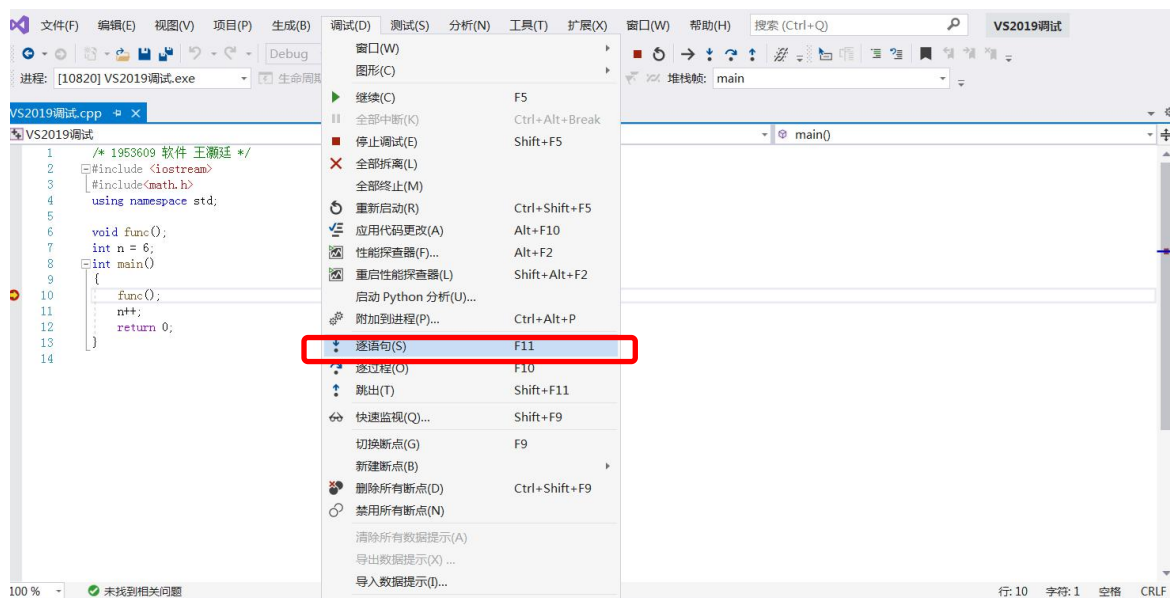
直接按 F5 键或者点击“调试”-“开始调试”进行调试。

#### 1.1.2 结束调试



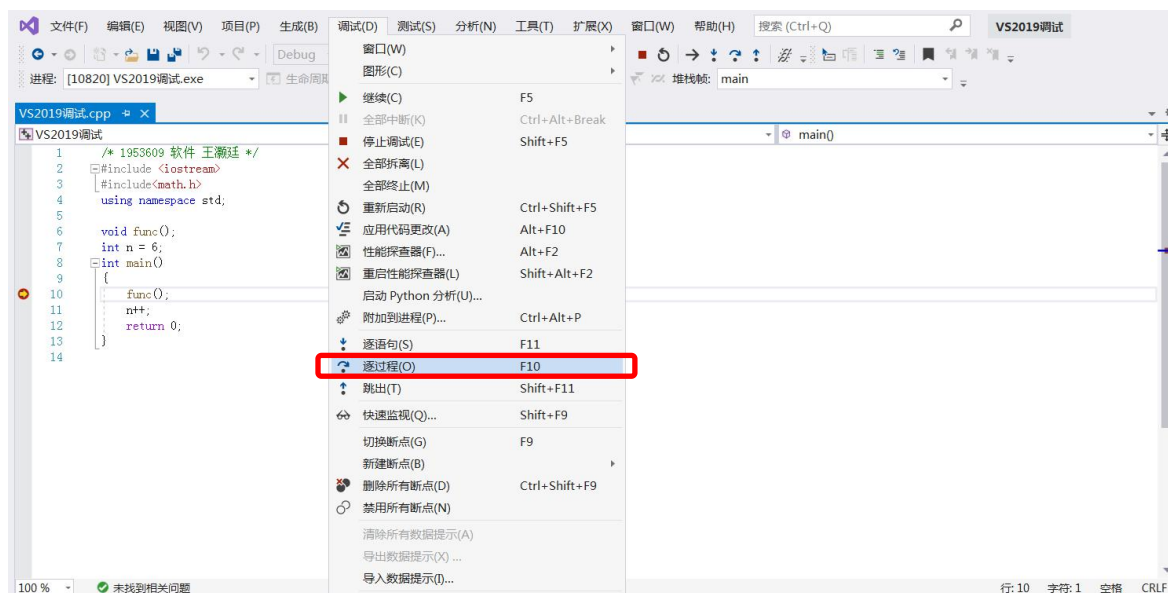
直接按 shift 和 F5 或者点击调试-停止调试 也可以按红色的方块

## 1.2 如何在一个函数中每个语句单步执行？



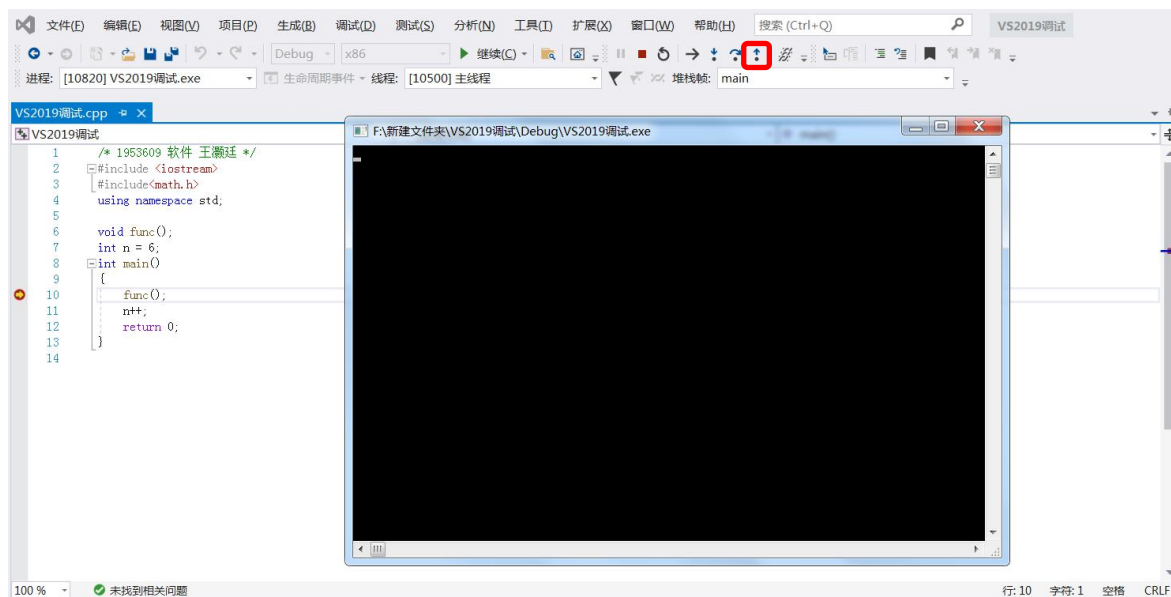
点击 调试 - 逐语句 或者 直接按“F11”

## 1.3 在碰到 cout/sqrt 等系统类/系统函数时，如何一步完成这些系统类/系统函数的执行而不要进入到这些系统类/函数的内部单步执行？



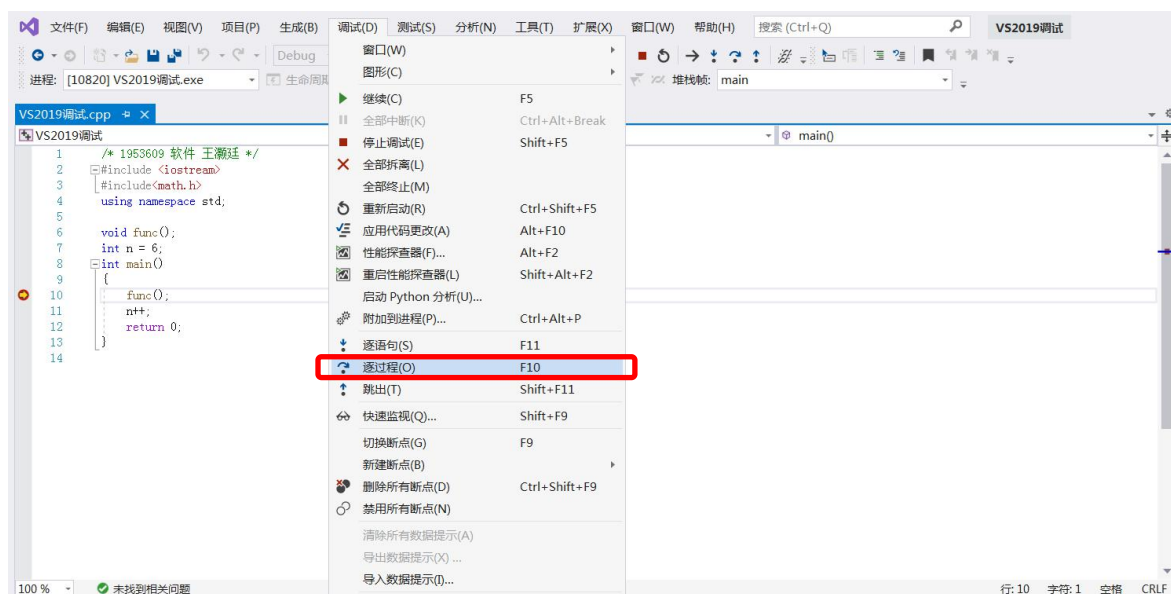
调试 - 逐过程 或者 直接按“F10”（一般情况下逐语句不会进入系统类/系统函数）

## 1.4 如果已经进入到 cout/sqrt 等系统类/系统函数的内部，如何跳出并返回自己的函数？



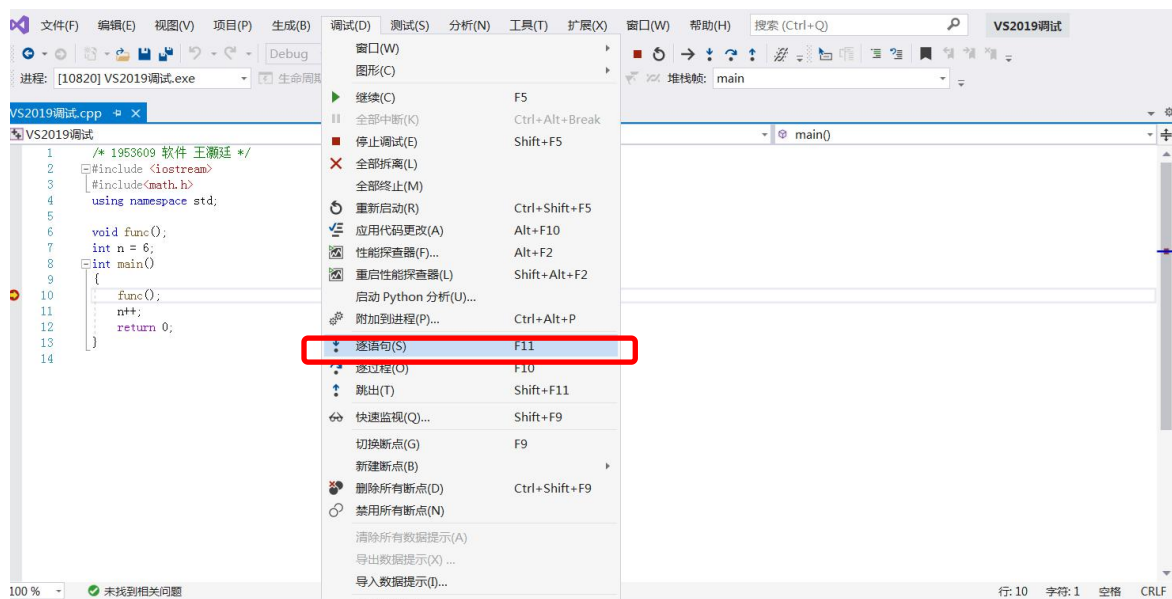
调试 - 跳出 或者 按 “Shift+F11”

1.5 在碰到自定义函数的调用语句（例如在 main 中调用自定义的 fun 函数）时，如何一步完成 自定义函数的执行而不要进入到这些自定义函数的内部单步执行？



按 F10 或者 点击 调试-逐过程。

1.6 在碰到自定义函数的调用语句（例如在 main 中调用自定义的 fun 函数）时，如何转到被调 用函数中单步执行？



点击 调试 - 逐语句 或者 直接按“F11”

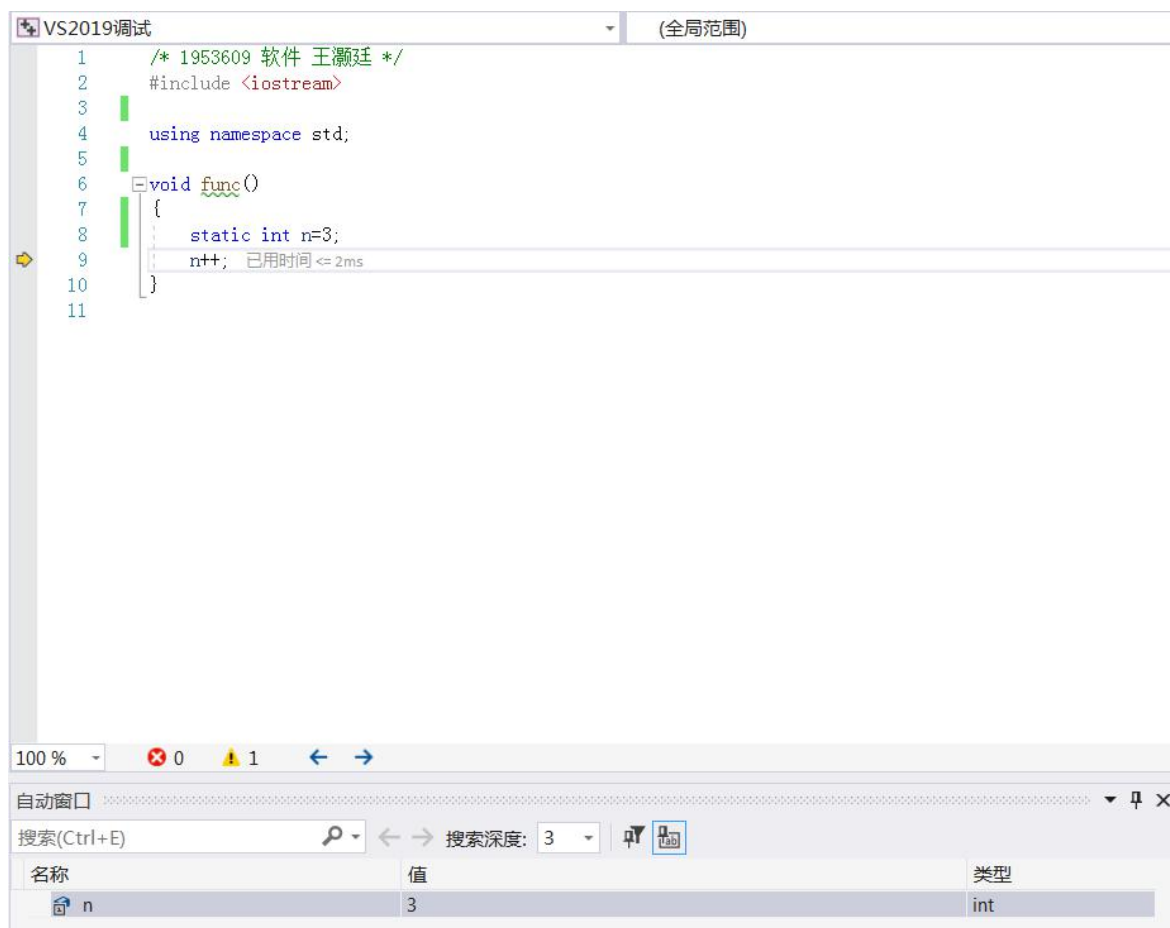
## 2. 使用VS2019的调试工具查看各种生存期/作用域变量

### 2.1 查看形参/自动变量的变化情况



在调试过程中，按 窗口 — 局部变量 在屏幕左下角会有一个窗口，显示自动变量的变化情况，同样当调试进入函数时，形参的变化情况也会显示。

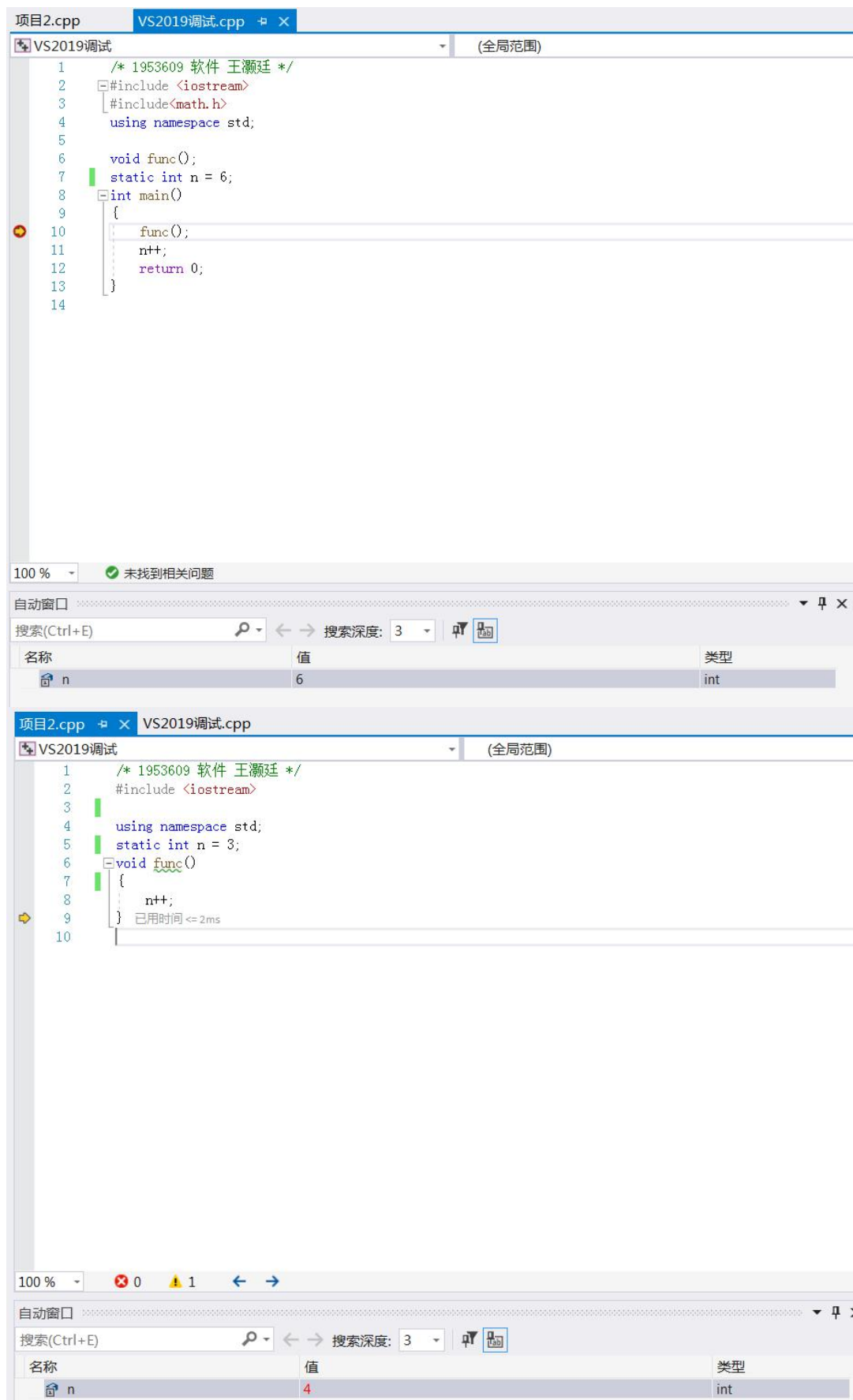
## 2.2 查看静态局部变量的变化情况（该静态局部变量所在的函数体内/函数体外）



静态局部变量同样在窗口中显示，但只有当调试在该静态局部变量所在的函数内进行时才能被查看到，在函数体外无法查看。

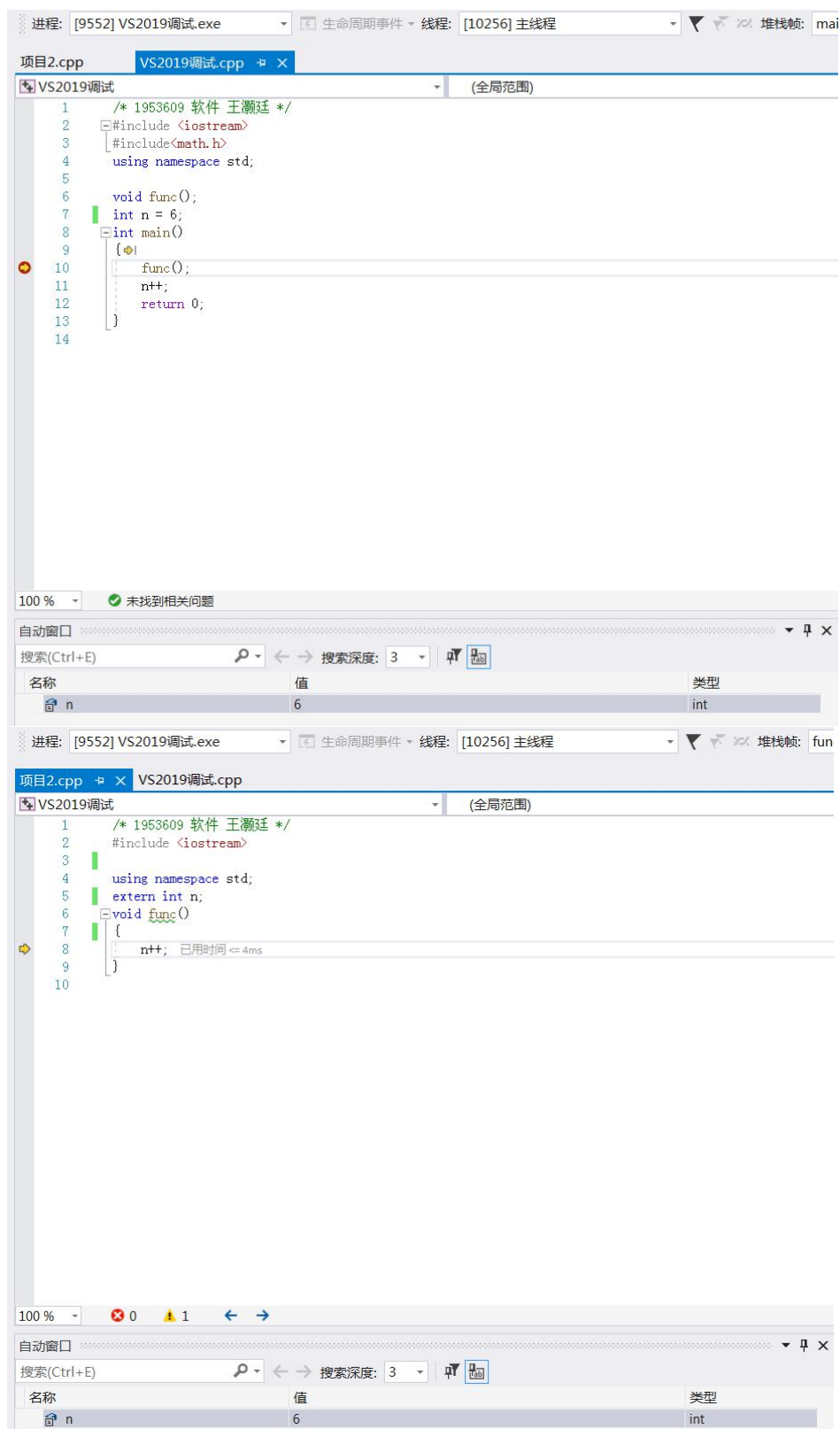
## 2.3 查看静态全局变量的变化情况（两个源程序文件，有静态全局变量同名）

通过左下角的监视 1 栏可以自己添加想要观察的变量，但是当两个源程序文件静态全局变量同名时，在哪个文件中，显示的就是哪个文件中的变量（但我认为这种情况应极力避免，在项目比较大的时候很难区分同名变量分别代表的含义）





## 2.4 查看外部全局变量的变化情况（两个源程序文件，一个定义，另一个有 extern 说明）





通过左下角的监视栏可以自己添加想要观察的变量，而当两个源程序文件，一个定义，另一个有 `extern` 说明时，`n` 始终是一个，正常查看即可。

### 3. 使用VS2019的调试工具查看各种不同类型变量

#### 3.1 char/int/float 等简单变量

#### 3.2 指向简单变量的指针变量（如何查看地址、值？）

#### 3.3 一维数组

#### 3.4 指向一维数组的指针变量（如何查看地址、值？）

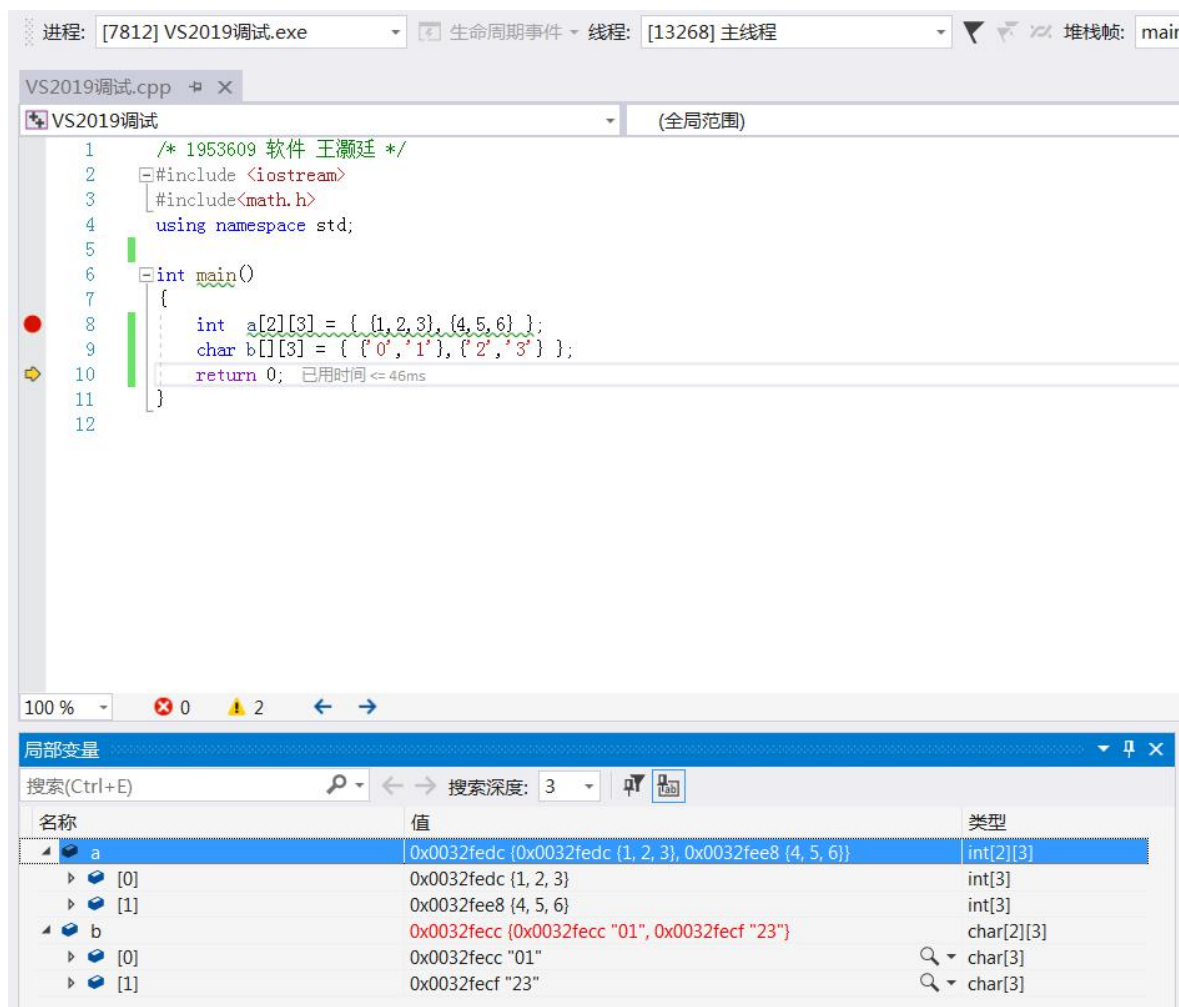
The screenshot shows the VS2019 IDE with a C++ program being debugged. The program defines variables `a` (char), `b` (int), `c` (float), `pa` (char\*), `pb` (int\*), `pc` (float\*), and `arr` (int array). The local variables window at the bottom displays the current state of these variables.

名称	值	类型
<code>a</code>	65 'A'	char
<code>arr</code>	0x0020f6ec {1, 2, 3}	int[3]
<code>b</code>	1	int
<code>c</code>	2.50000000	float
<code>p_arr</code>	0x0020f6ec {1}	int *
<code>pa</code>	0x0020f73f <字符串中的字符无效。>	char *
<code>pb</code>	0x0020f730 {1}	int *
<code>pc</code>	0x0020f724 {2.50000000}	float *

（同第二部分）局部变量通过局部变量栏查看，全局变量通过在监视1中输入变量名来查看  
 指针:地址在值一栏显示，而后面的大括号内则是具体的值（指向一维数组的指针变量的值为数组首元素）  
 但 `char*` 貌似无法显示字符值，只能显示目标地址，但输出值是正确的。  
 如果 `pa` 是指向字符的指针，`cout << pa;` 显示的是 `pa` 指向的字符串，而不是 `pa` 变量的值，这是指向字符的指针比较特殊的地方。

之所以这么做，可能是因为通过指针输出字符串比输出指针的值更为常用。

## 3.5 二维数组（包括数组名仅带一个下标的情况）



直接显示的是全部二维数组的地址值，点击二维数组名，会出现各个一维数组的地址和值。在数组名仅带一个下标时，编译器会自动补全，同样显示的是全部二维数组的地址值，点击二维数组名，会出现各个一维数组的地址和值。

## 3.6 实参是一维数组名，形参是指针的情况，如何在函数中查看实参数组的地址、值？

实参是一维数组名，形参是指针时，数组退化成指针，函数中无法直接查看实参数组的地址和值，只有数组的首地址和首地址值，但是可以通过监视指针来间接达到（见下页）

进程: [10192] VS2019调试.exe 生命周期事件 线程: [12808] 主线程 堆栈帧: swa

VS2019调试.cpp

VS2019调试 (全局范围)

```

1  /* 1953609 软件 王灏廷 */
2  #include <iostream>
3  #include<math.h>
4  using namespace std;
5  void swap(int* a)
6  { 已用时间 <= 3ms
7      int tmp;
8      tmp = a[0];
9      a[0] = a[1];
10     a[1] = tmp;
11 }
12 int main()
13 {
14     int a[2] = { 1,2 };
15     swap(a);
16     return 0;
17 }

```

100 % 0 6

局部变量

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
a	0x003dfeec {1}	int *
tmp	2130567168	int

进程: [10192] VS2019调试.exe 生命周期事件 线程: [12808] 主线程 堆栈帧: swap

VS2019调试.cpp

VS2019调试 (全局范围)

```

1  /* 1953609 软件 王灏廷 */
2  #include <iostream>
3  #include<math.h>
4  using namespace std;
5  void swap(int* a)
6  {
7      int tmp;
8      tmp = a[0];
9      a[0] = a[1];
10     a[1] = tmp; 已用时间 <= 13ms
11 }
12 int main()
13 {
14     int a[2] = { 1,2 };
15     swap(a);
16     return 0;
17 }

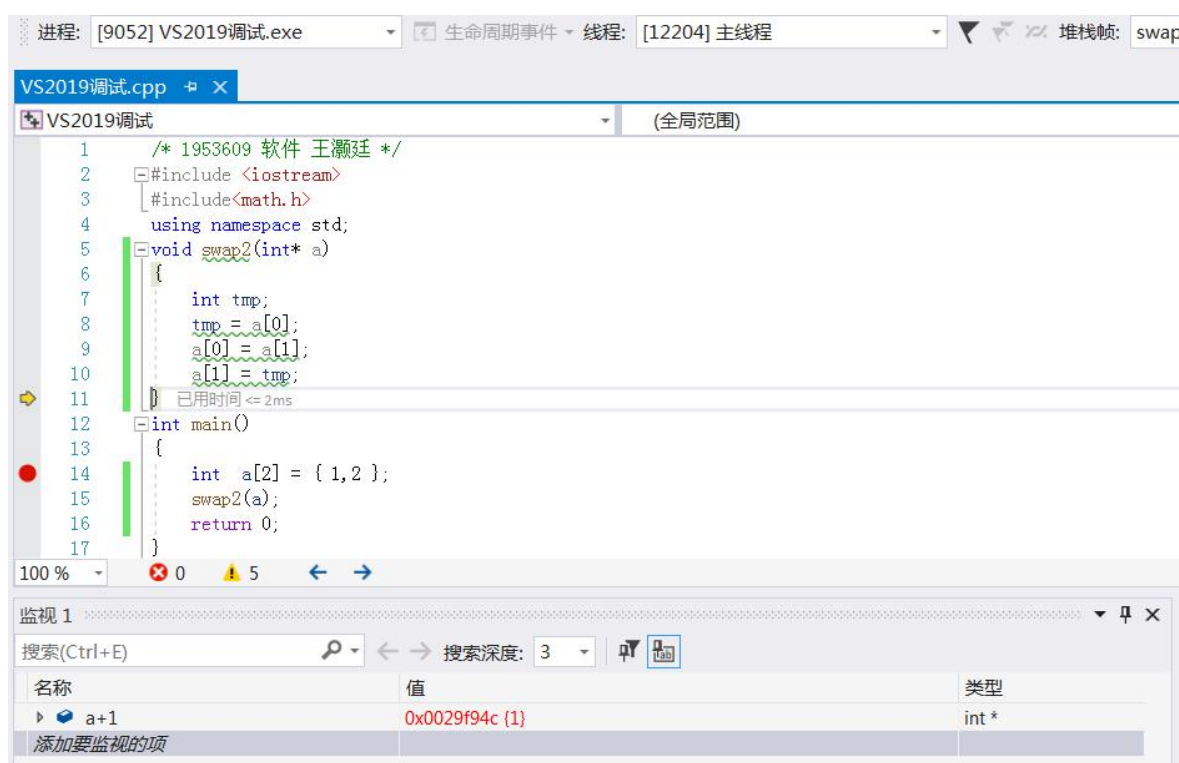
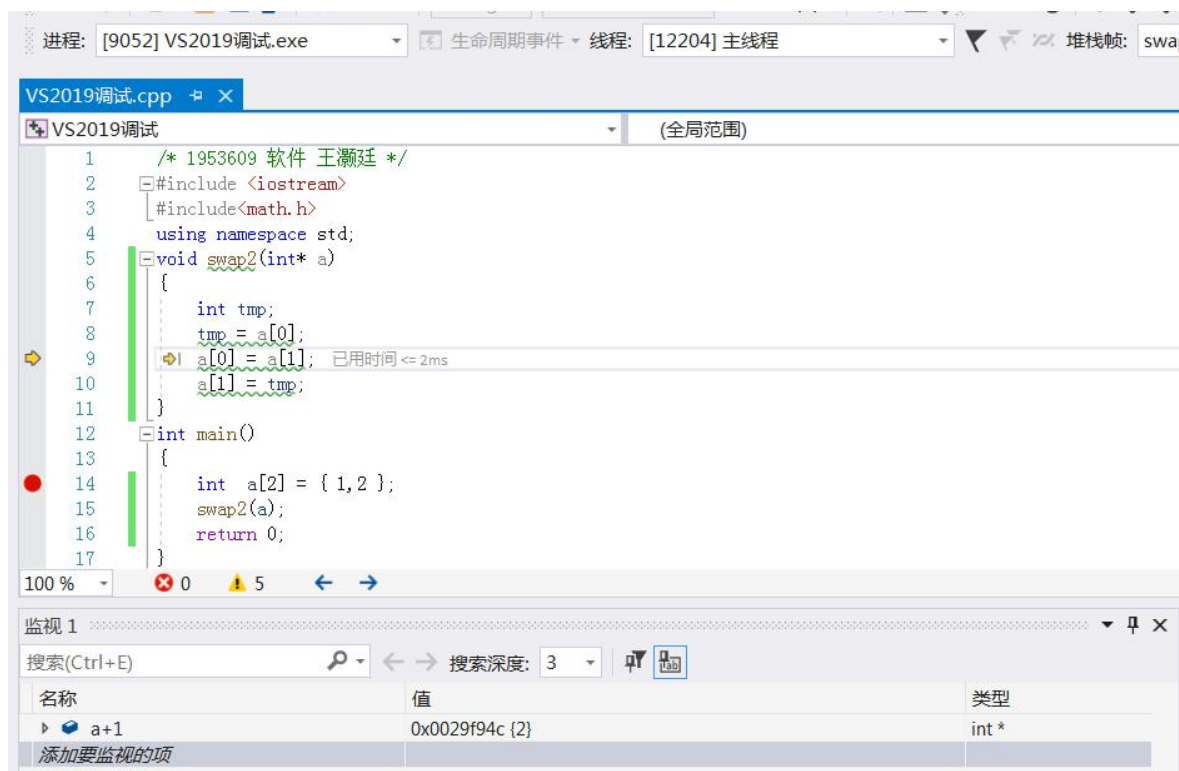
```

100 % 0 6

局部变量

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
a	0x003dfeec {2}	int *
tmp	1	int



3.7 指向字符串常量的指针变量（能否看到无名字符串常量的地址？）

```

1  /* 1953609 软件 王灏廷 */
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  int main()
7  {
8      const char* p = "abcde";
9      return 0; 已用时间 <= 1ms
10 }
11

```

自动窗口

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
p	0x01219bd0 "abcde"	const char *

可以看到无名字符串常量的地址

### 3.8 引用（引用与指针是否有区别？有什么区别？）

```

1  /* 1953609 软件 王灏廷 */
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  void swap2(int (&a)[2])
7  {
8      int tmp;
9      tmp = a[0];
10     a[0] = a[1];
11     a[1] = tmp;
12 }
13
14 void func(int& c)
15 {
16     c++;
17 }
18
19 int main()
20 {
21     int p[2] = { 1, 2 };
22     int a = 3;
23     int& b = a;
24     int* pp = p;
25     func(a); 已用时间 <= 1ms
26     swap2(p);
27     return 0;
28 }
29

```

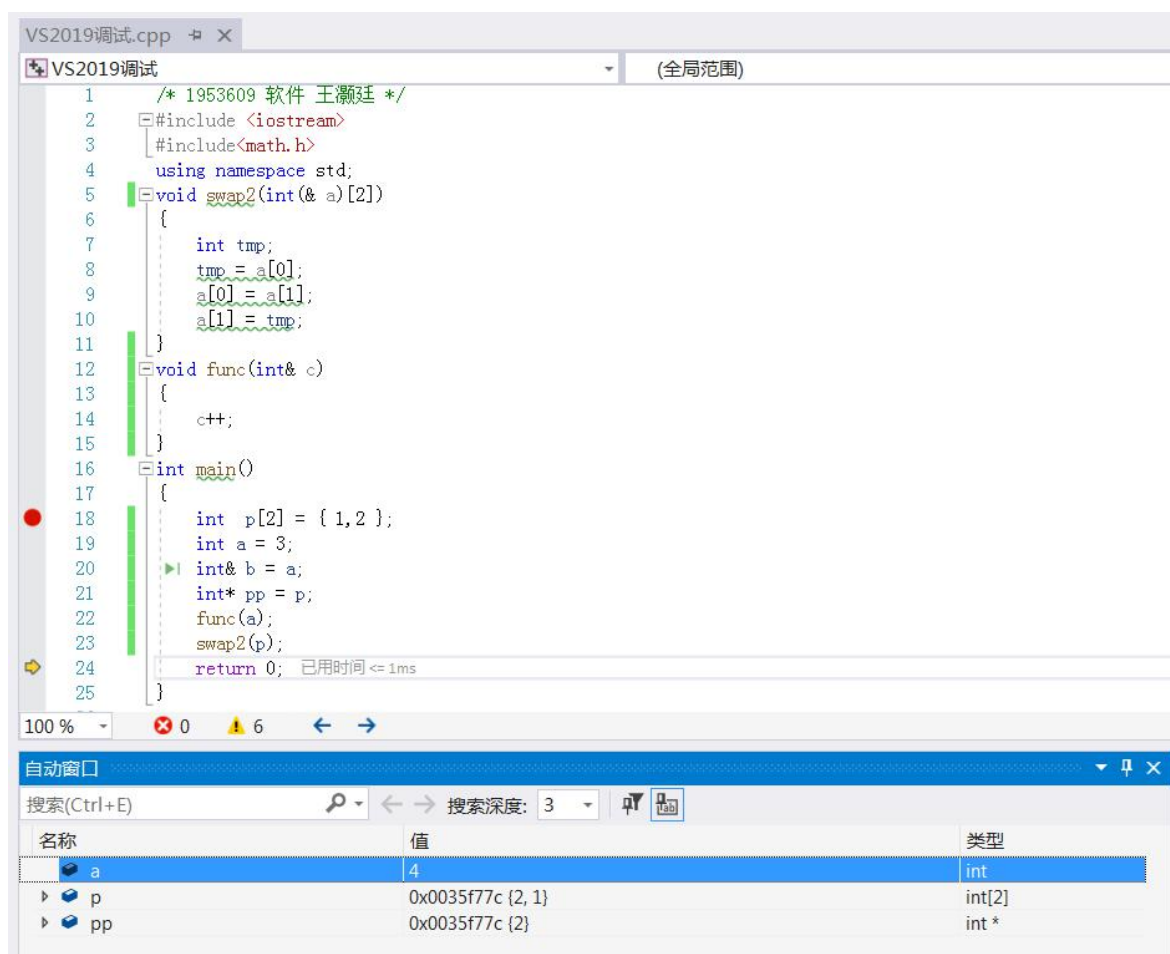
自动窗口

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
a	3	int
b	3	int &
p	0x0035f77c (1, 2)	int[2]
pp	0x0035f77c (1)	int *



由调试可知引用是变量的别名，而指针是存放地址的变量



```

1  /* 1953609 软件 王灏廷 */
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5  void swap2(int(& a)[2])
6  {
7      int tmp;
8      tmp = a[0];
9      a[0] = a[1];
10     a[1] = tmp;
11 }
12 void func(int& c)
13 {
14     c++;
15 }
16 int main()
17 {
18     int p[2] = { 1, 2 };
19     int a = 3;
20     int& b = a;
21     int* pp = p;
22     func(a);
23     swap2(p);
24     return 0; 已用时间 <= 1ms
25 }
    
```

名称	值	类型
a	4	int
p	0x0035f77c {2, 1}	int[2]
pp	0x0035f77c {2}	int *

在这里看到数组也同样成立，在函数中使用引用或指针都可以改变实参。

### 3.9 使用指针时出现了越界访问



无可来源 VS2019调试.cpp

### 帧不在模块中

在加载的模块中未找到当前堆栈帧。无法显示此位置的源代码。

您可以在“反汇编”窗口中 [查看反汇编](#)。若要总是通过查看反汇编来确定缺少的源文件，请更改“选项”对话框中的设置。

已引发异常

0x00000005 处(位于 VS2019调试.exe 中)引发的异常: 0xC0000005: 执行位置 0x00000005 时发生访问冲突。

[复制详细信息](#) | [启动 Live Share 会话...](#)

异常设置

☒ 引发此异常类型时中断

[打开异常设置](#) | [编辑条件](#)

VS2019调试

```

1  /* 1953609 软件 王灏廷 */
2  #include <iostream>
3  #include <math.h>
4  using namespace std;
5
6  int main()
7  {
8      int p[2] = { 1, 2 };
9      int* pp = p;
10     *(pp + 5) = 5;
11     return 0;
12 }
13

```

100 % 0 2

自动窗口

搜索(Ctrl+E) 搜索深度: 3

名称	值	类型
p	0x001efeb8 {1, 2}	int[2]
pp	0x001efeb8 {1}	int *

在调试时可以看到越界赋值成功，但是会引发异常，因为访问了非法地址。