
《高级程序语言设计》

实验报告

**实 验 名 称:** VS2022调试工具的使用与总结

**班　　级:** 数据科学与大数据技术

**学　　号:** 2351495

**姓　　名:** 闫业豪

**完 成 时 间:** 2025年6月3日

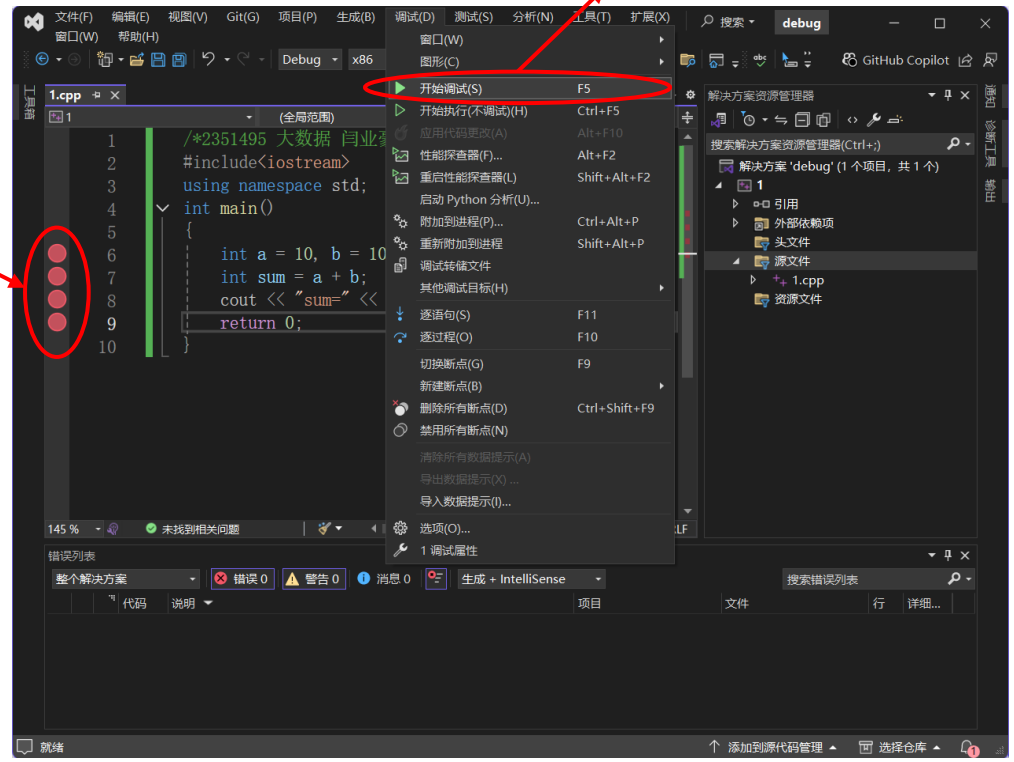
# 1.VS2022调试工具的基本使用方法

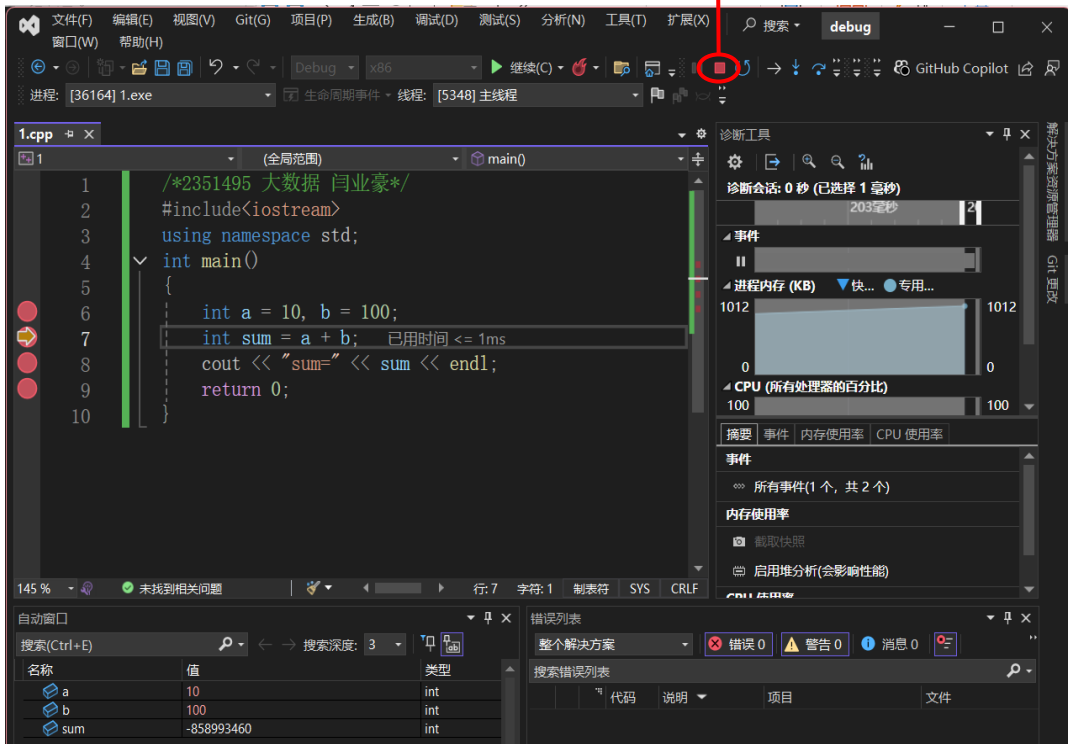
## 1.1开始/结束调试

按此键或F5进行调试

按此键或shift+F5结束调试

在程序左侧
打断点



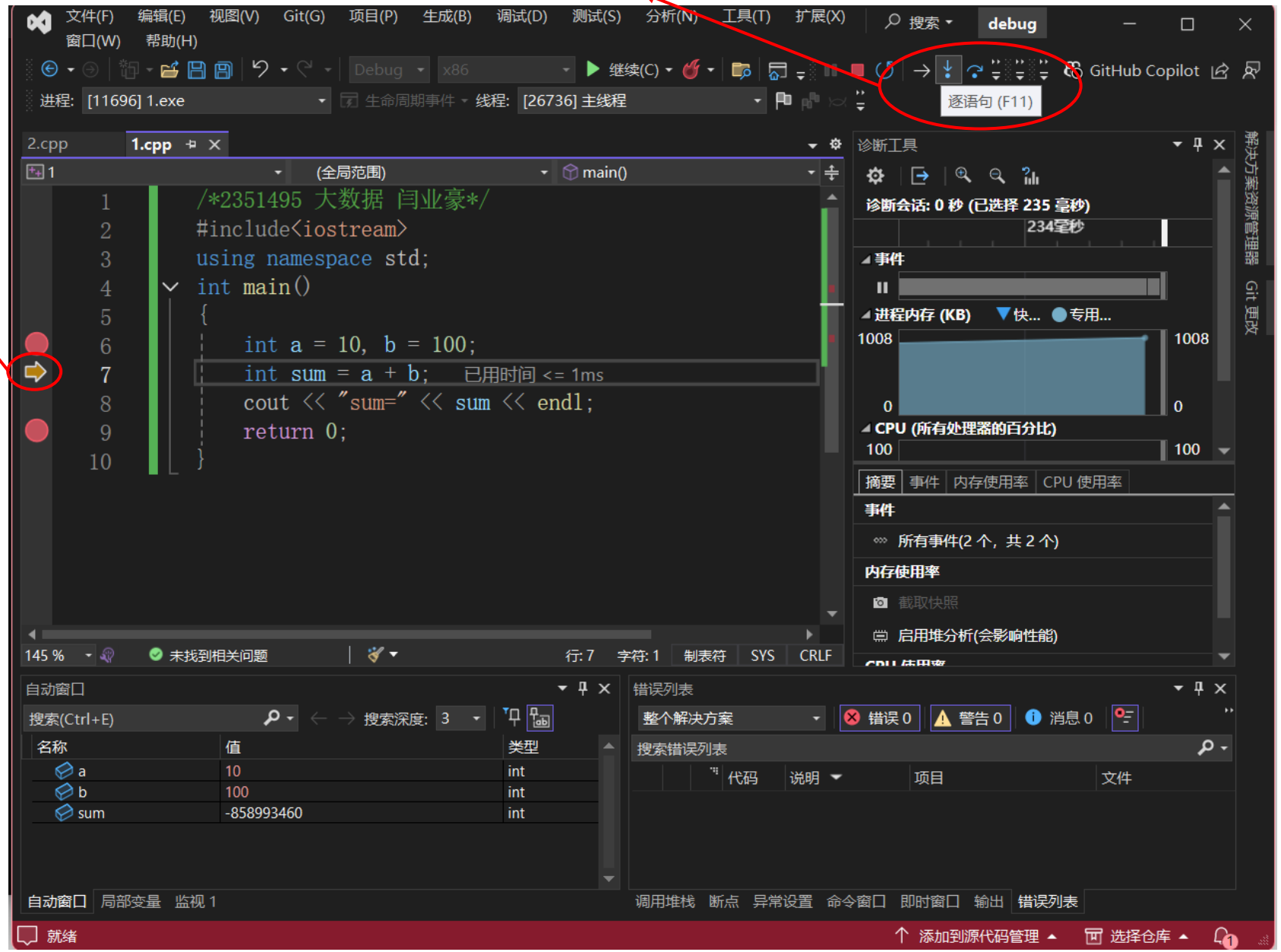




通过添加断点可以在指定位置使程序停止执行

本页涉及知识点1.1

# 1.2单步执行

按此键或F11即可单步执行




黄色箭头指示当前要执行的指令

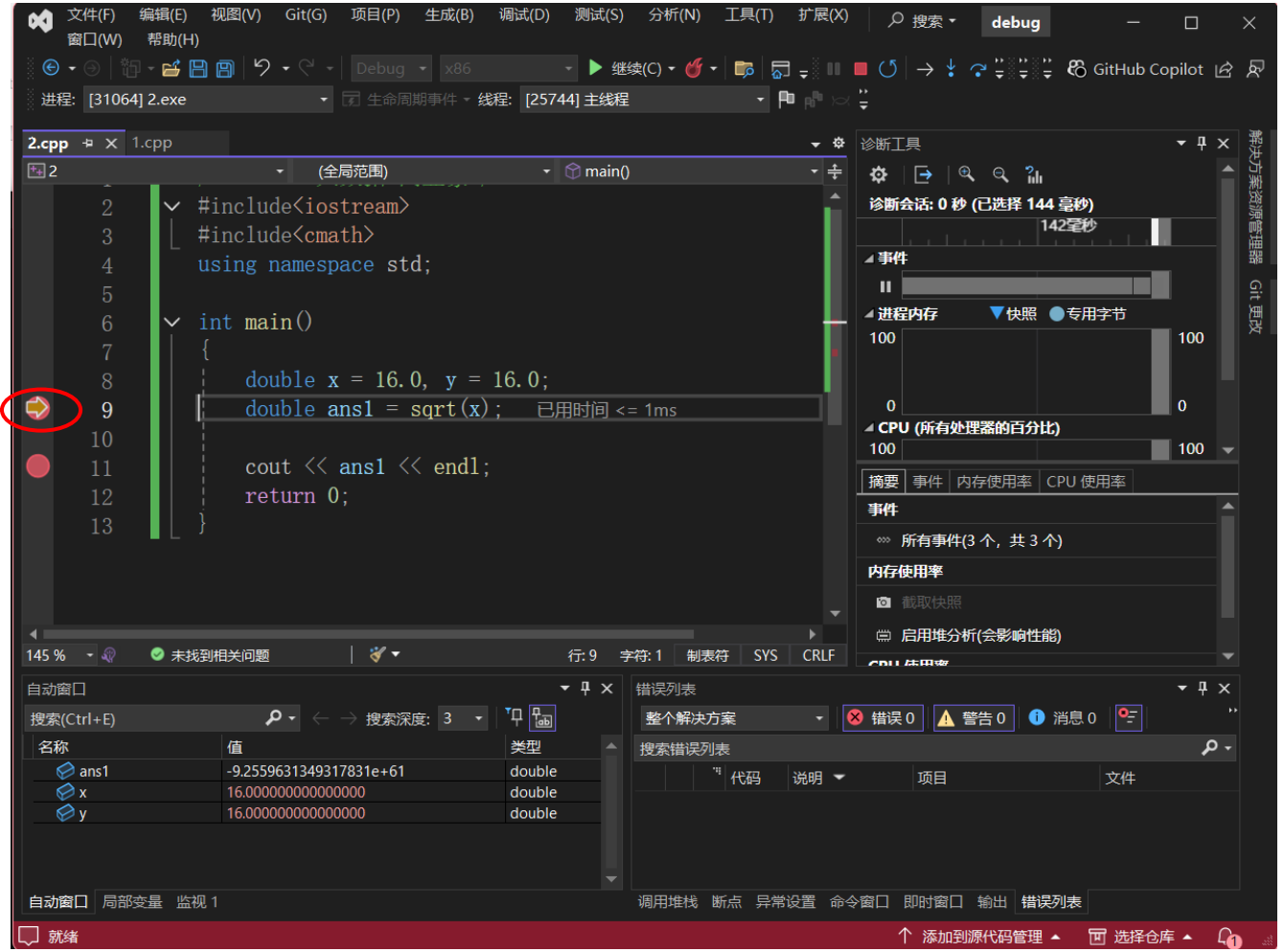
本页涉及知识点1.2

## 1.3跳过系统函数

## 1.4跳出系统函数

当前要执行的语句为标准库函数sqrt，视为完整的语句，单步执行不会进入其内部。

按F11逐语句执行即可



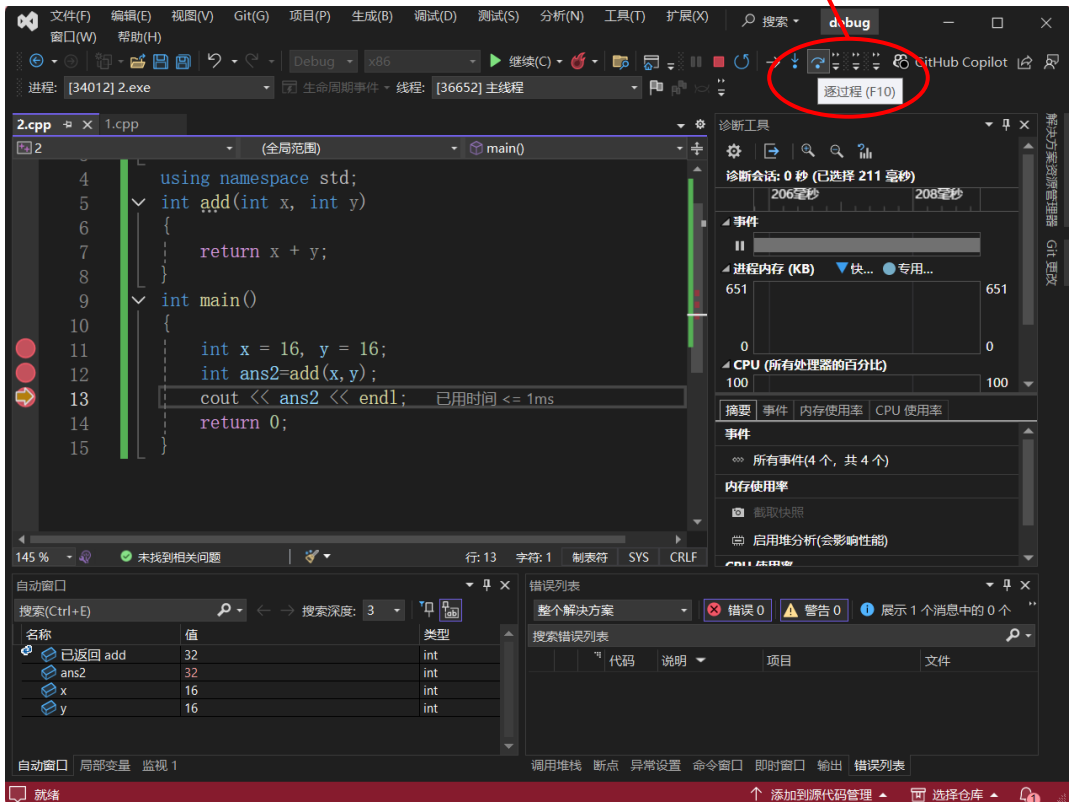

系统函数/系统类视为一个完整的语句，单步执行不会进入其内部,该语句执行完后便会跳出

本页涉及知识点1.3、1.4

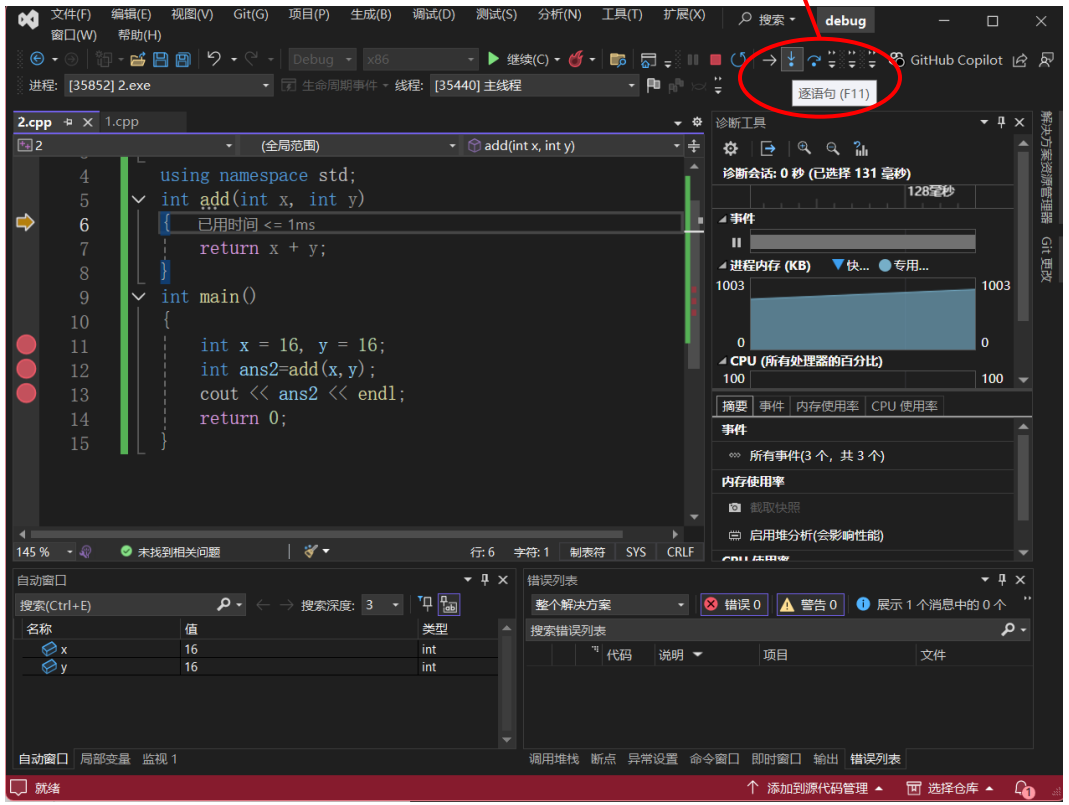
## 1.5跳过自定义函数

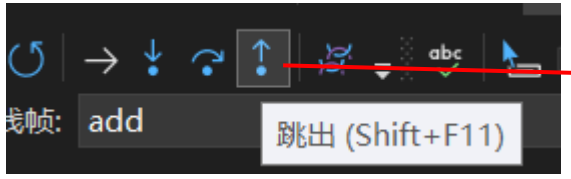
## 1.6进入自定义函数

逐过程(F10)可以将函数调用视为单个语句一次性执行，不会停留在被调用的函数内




逐语句(F11)会跳转入被调用函数中执行单步执行




Shift+F11可以完成当前函数执行，跳出函数

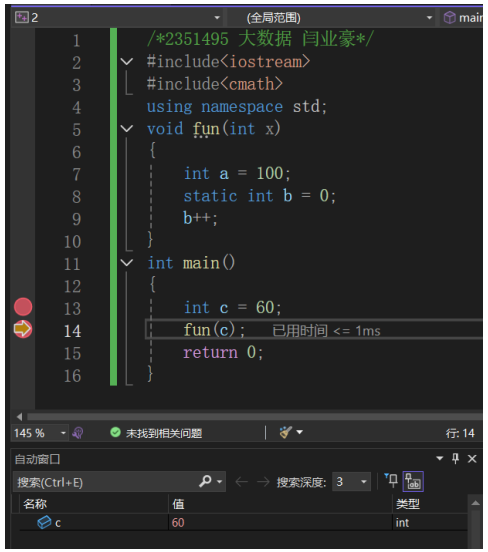
本页涉及知识点1.5、1.6

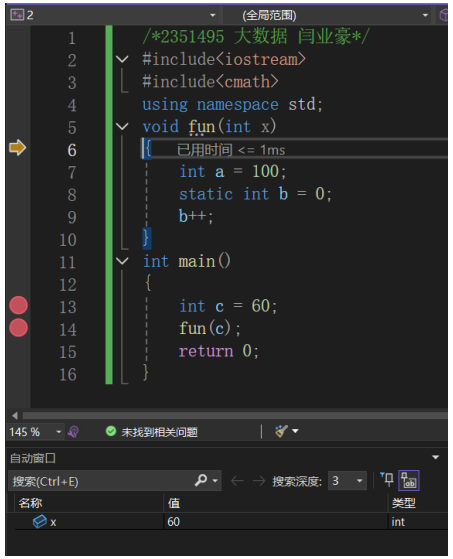
# 2.查看不同生存期/作用域变量

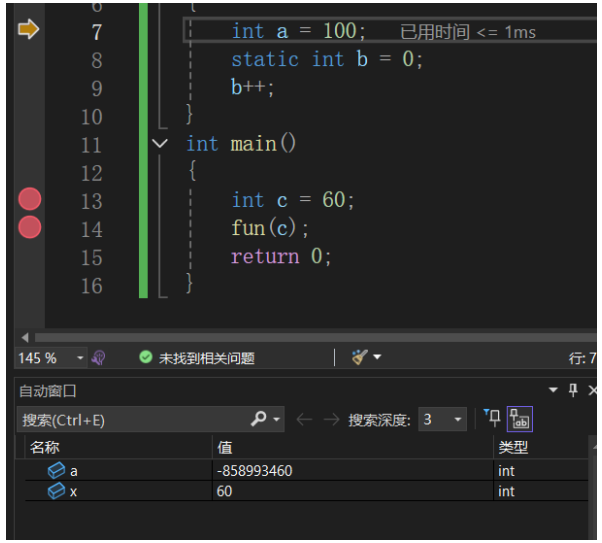
## 2.1形参 / 自动变量

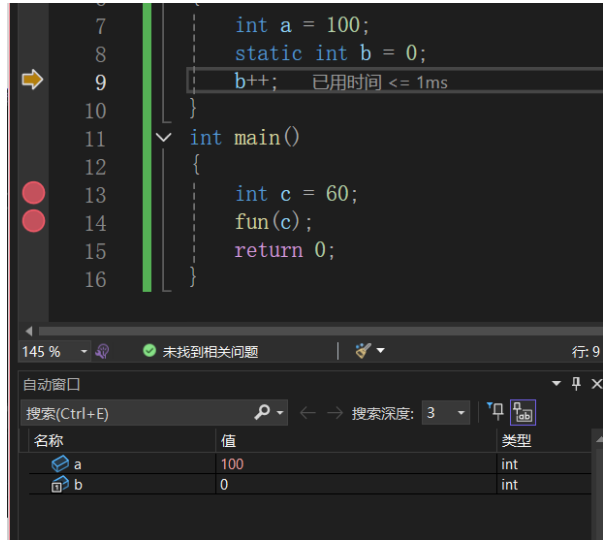
## 2.2静态局部变量

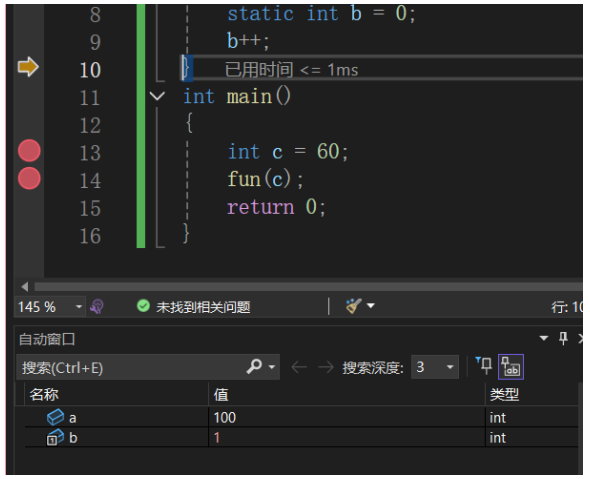
注意黄色箭头和窗口内的变量名称即可




在调试此程序时，局部变量窗口会根据程序执行的当前位置显示不同作用域内的变量。当程序执行在main函数中时，窗口显示局部变量c（值为60）；当程序执行进入fun函数内部时，窗口会显示该函数的参数x（值为60，由c传入）、局部变量a（值为100）以及静态局部变量b（初值为0，每次函数调用后递增1）。需要注意的是，静态局部变量b只有在程序执行位置处于fun函数内部时才能在局部变量窗口中观察到，当程序执行返回到main函数或其他位置时，该静态变量将不再显示在窗口中，尽管它在内存中依然存在并保持其值

本页涉及知识点2.1、2.2

# 2.3静态全局变量

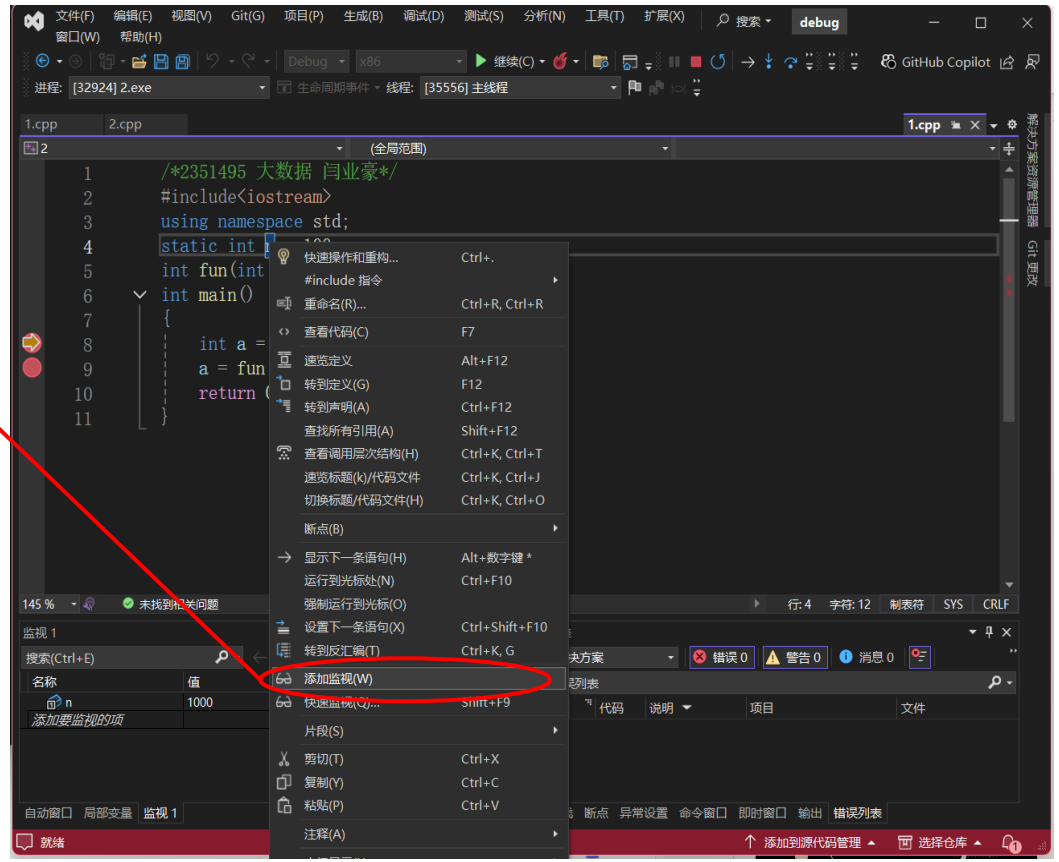
1.cpp

```
/*2351495 大数据 闫业豪*/
#include<iostream>
using namespace std;
static int n = 100;
int fun(int x);
int main()
{
int a = 10;
a = fun(a);
return 0;
}
```

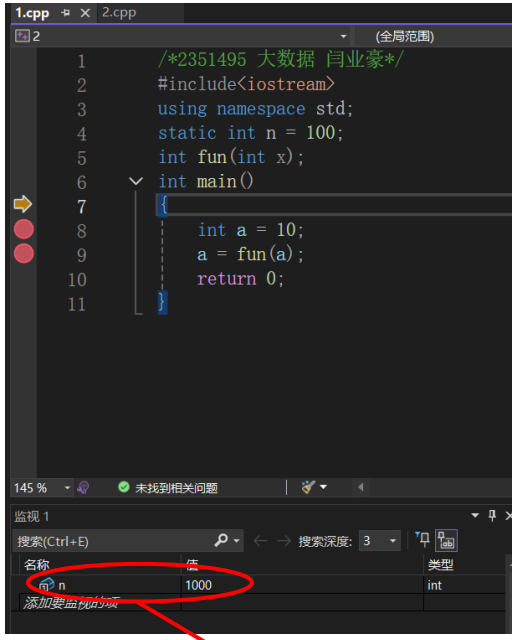
2.cpp

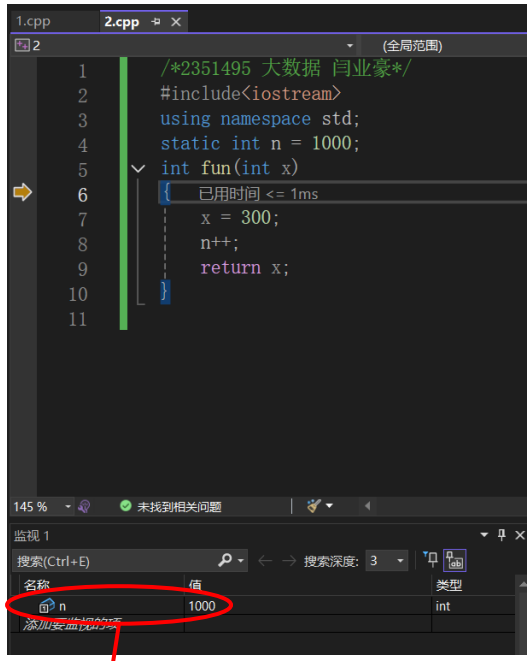
```
/*2351495 大数据 闫业豪*/
#include<iostream>
using namespace std;
static int n = 1000;
int fun(int x)
{
x = 300;
n++;
return x;
}
```

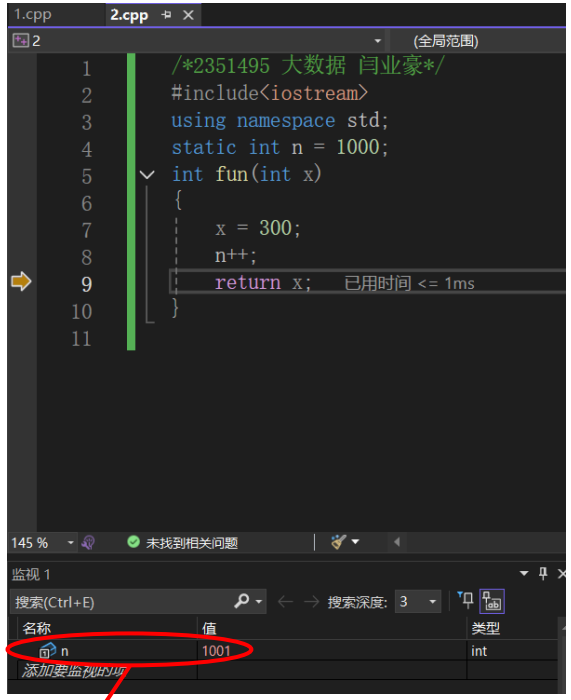
先将全局变量手动添加监视，添加后即可在监视窗口观察

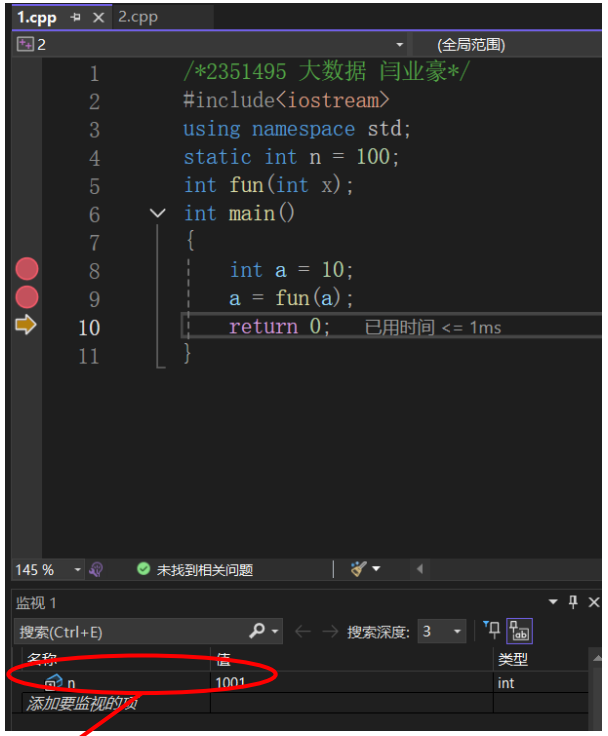



本页涉及知识点2.3



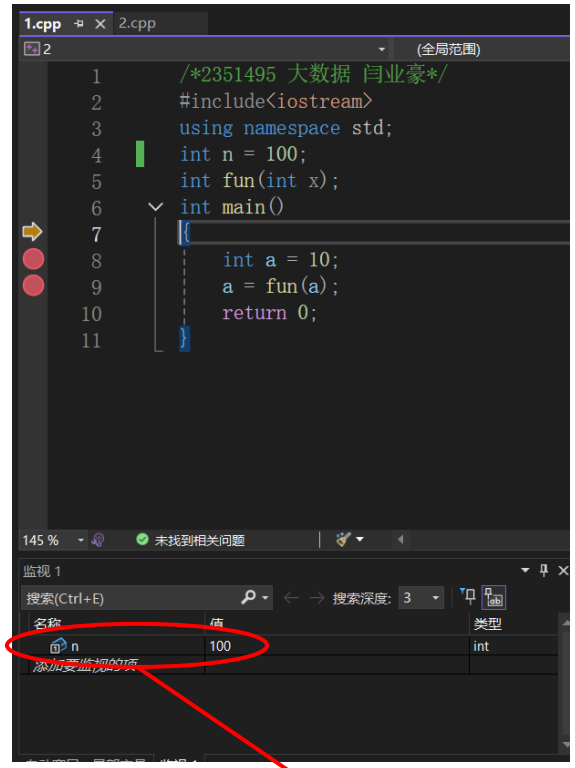



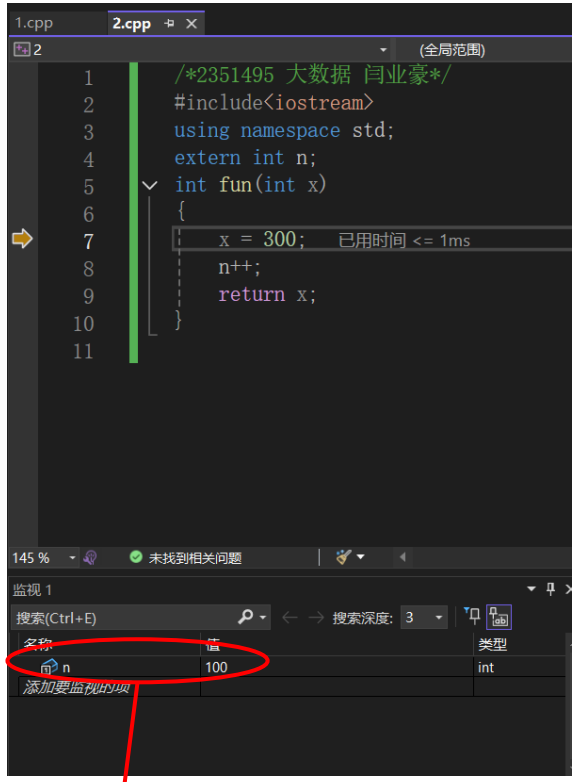



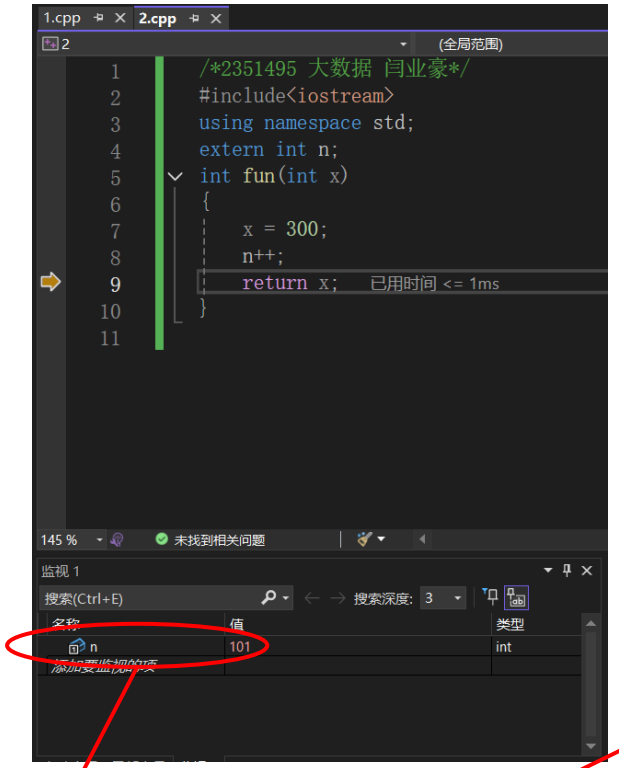

当2个源文件中都定义了同名的静态全局变量n时，VS2022的监视窗口会出现显示混淆的问题。实际调试过程中发现，无论程序执行在哪个文件中，监视窗口显示的都是同一个n变量的值（2.cpp），这使得开发者无法准确区分和跟踪两个文件中各自的静态全局变量状态。这种情况在调试时极容易造成混淆，建议在实际开发中避免在不同源文件中使用相同名称的静态全局变量，或采用更具区分性的命名方式来避免此类调试困

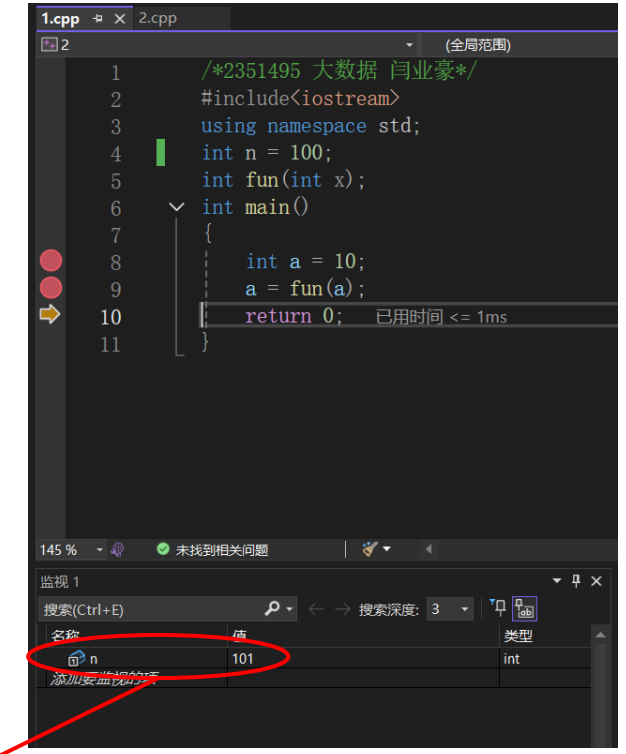
本页涉及知识点2.3

# 2.4外部全局变量



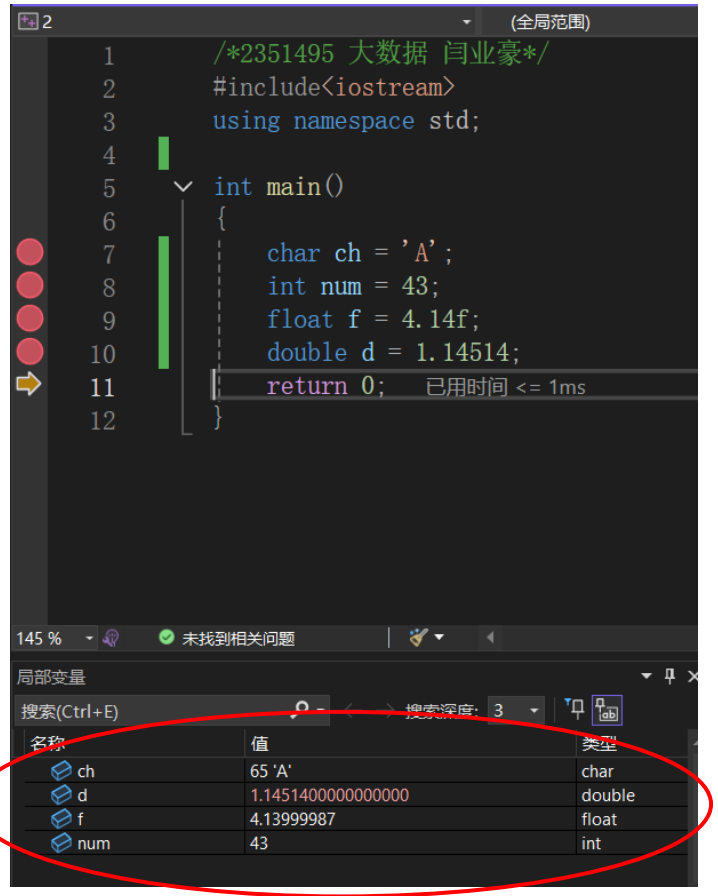




当一个cpp文件中定义了全局变量n，而另一个cpp文件中使用extern声明引用该变量时，两个文件实际上共享的是同一个全局变量。在VS2022的监视窗口中，无论程序执行在哪个文件中，显示的都是同一个n变量的值，这是正确的行为表现。通过extern声明，第二个文件获得了对第一个文件中定义变量的访问权限，因此在调试过程中观察到的变量值变化是统一且一致的，这与静态全局变量的情况不同，不会产生混淆问题，可以正常进行调试观察

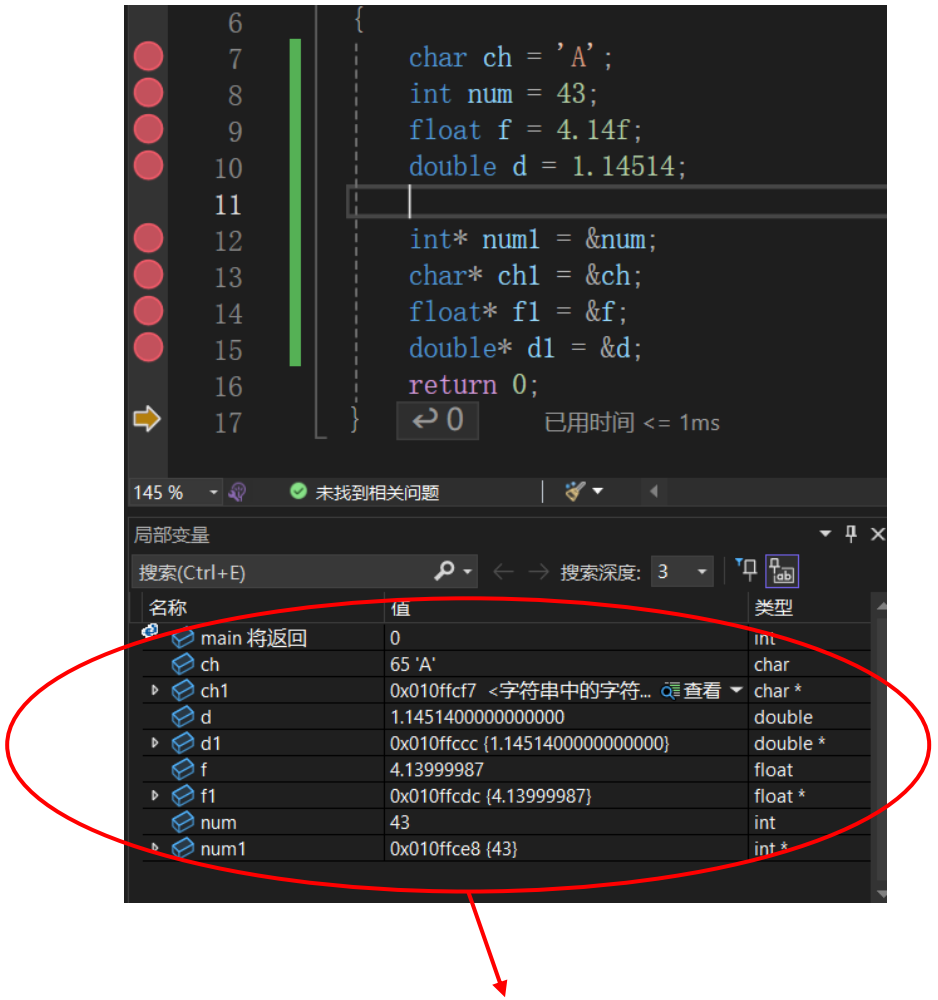
本页涉及知识点2.4

# 3.查看不同类型变量

## 3.1简单变量
## 3.2指向简单变量的指针




对于基本数据类型变量可以直接在监视窗口、局部变量窗口或自动窗口中查看其当前值。这些简单类型变量的值会以直观的数值形式显示，无需进行额外的展开或特殊操作，调试器会自动将变量的内存内容转换为可读的数值格式进行展示。

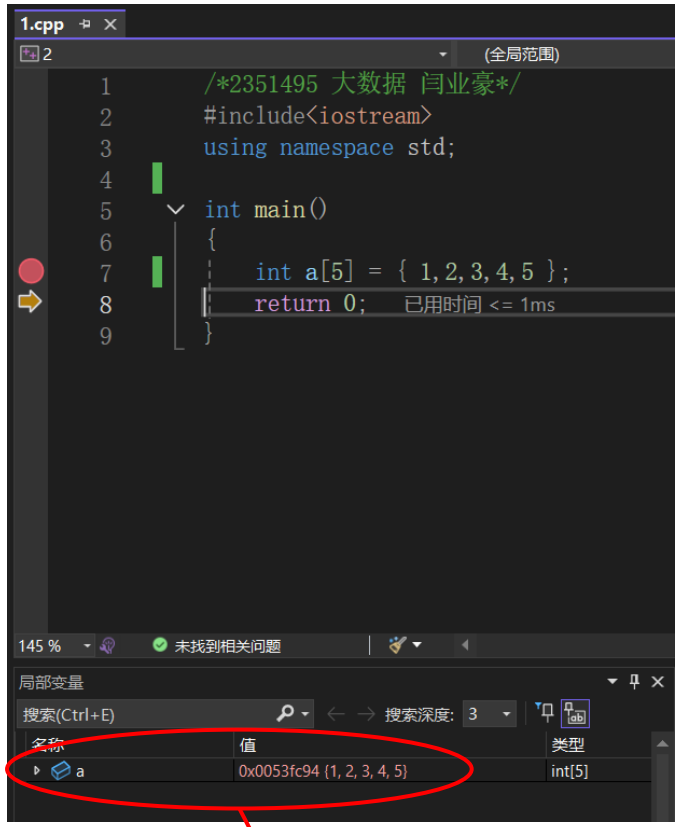



指针地址后紧跟大括号{}，括号内显示该指针所指向内存地址中存储的实际数值。当指针指向有效的内存区域时，大括号内会正常显示目标变量的值；但如果指针尚未初始化、指向无效地址或发生越界访问时，大括号内则会显示"???"，这是调试器提示该指针指向的内存区域无法正常访问或读取的标志

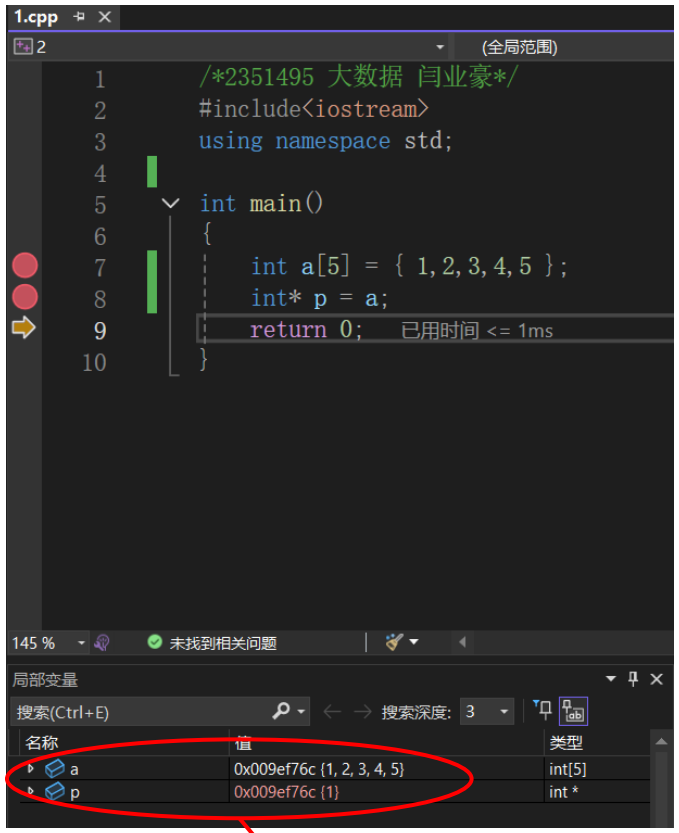
本页涉及知识点3.1、3.2

# 3.3一维数组

# 3.4指向一维数组的指针




一维数组变量会显示其首元素的内存地址值，同时在紧随其后的大括号{}内完整展示数组中所有元素的值。调试器会自动遍历整个数组并将每个元素的值按顺序列出

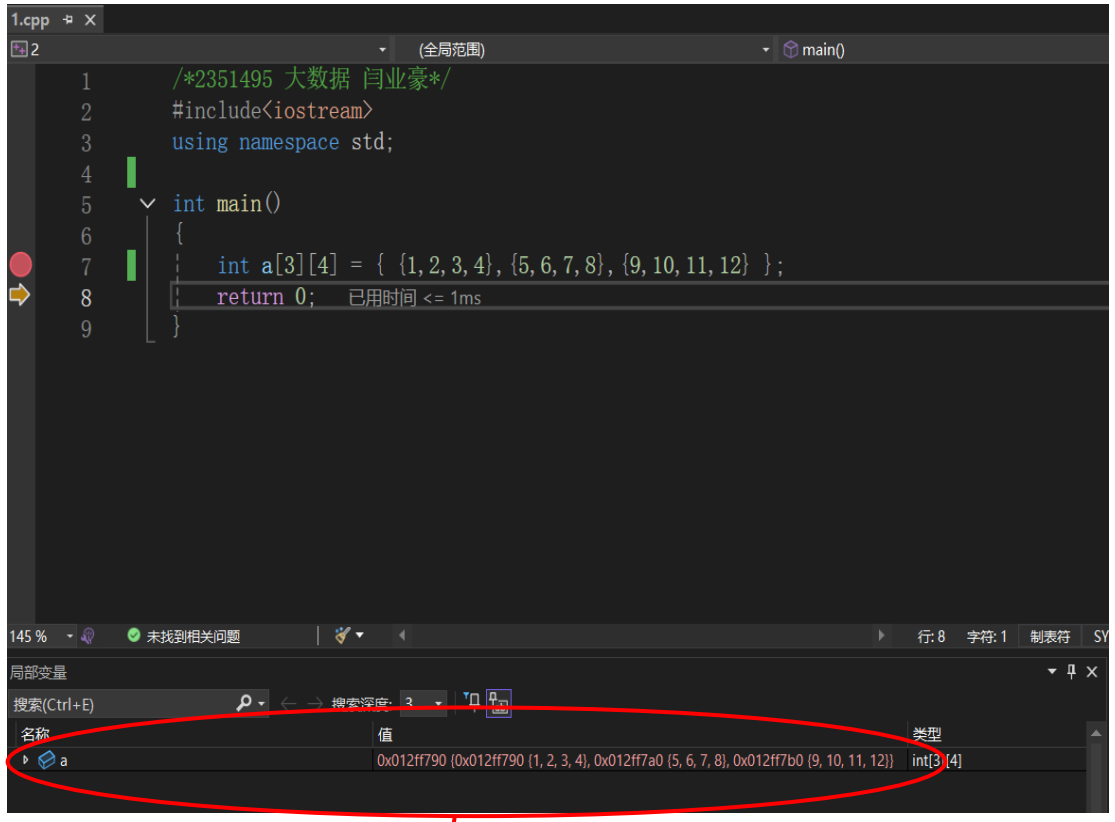



对于指向一维数组的指针变量，调试窗口会显示该指针所指向数组的首地址，并在大括号{}内显示数组的第一个元素值。与直接的数组变量不同，指针只会显示第一个元素而非整个数组内容。需要特别注意的是，如果指针发生越界访问或指向无效内存区域，大括号内显示的数值将不可信

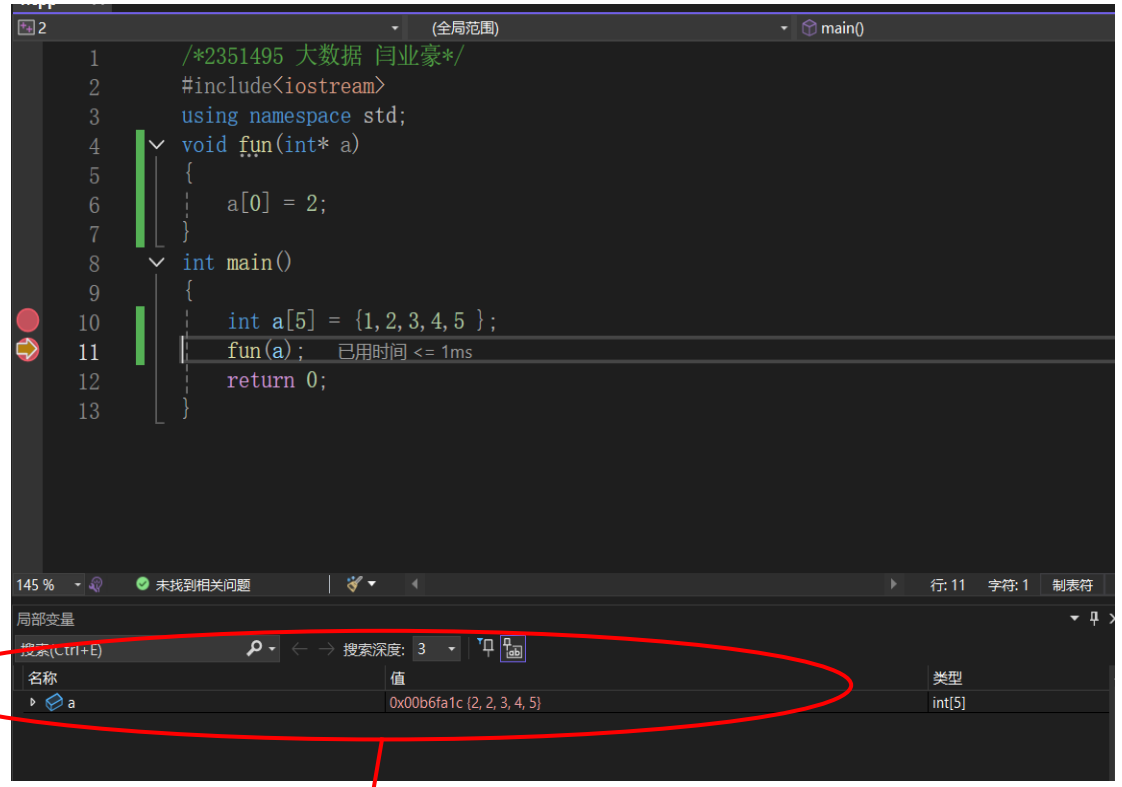
本页涉及知识点3.3、3.4

# 3.5二维数组

# 3.6数组作为参数传递




二维数组变量会显示其在内存中的首地址，并在后面的括号内完整展示整个数组的所有元素值。调试器会按照内存中的存储顺序，将二维数组的所有元素以一维的形式连续显示出来

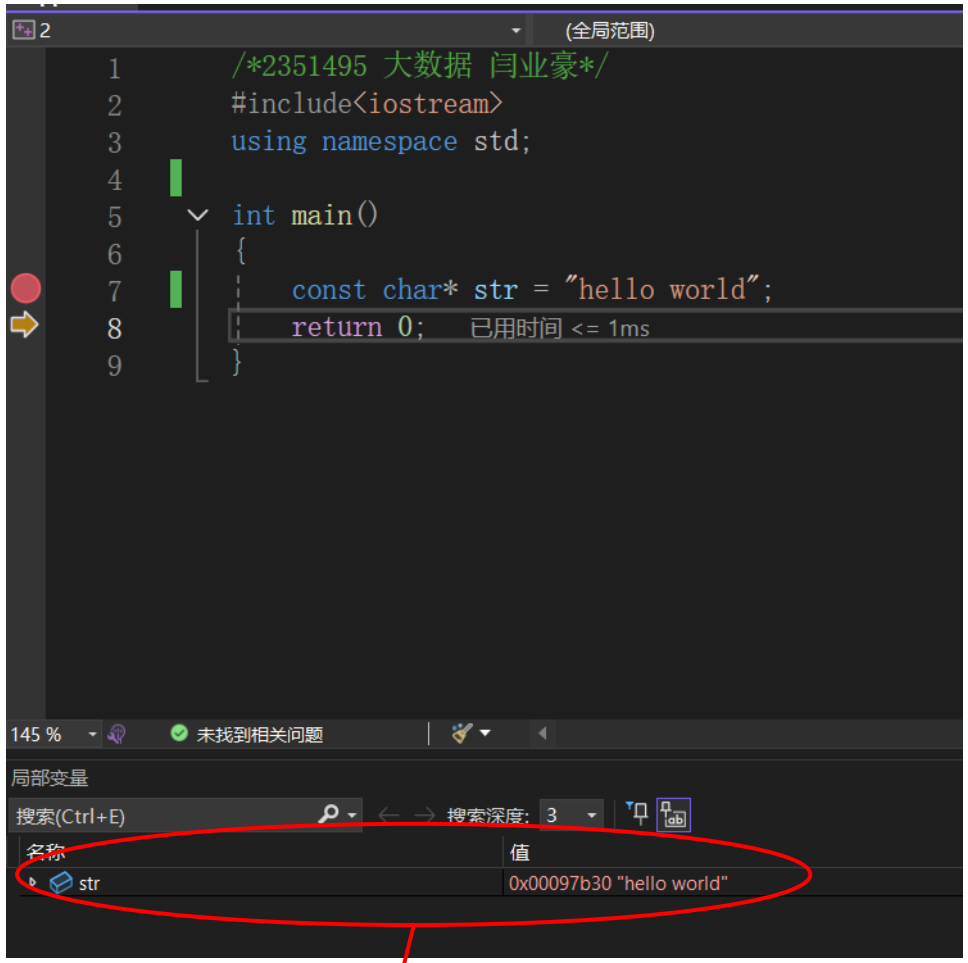



当一维数组作为指针类型的实参传递给函数时，其效果等同于将数组首元素的地址直接赋值给形参指针。在调试窗口内，该形参指针会显示数组首元素的内存地址值，并在紧随的大括号{}内显示首元素的具体数值

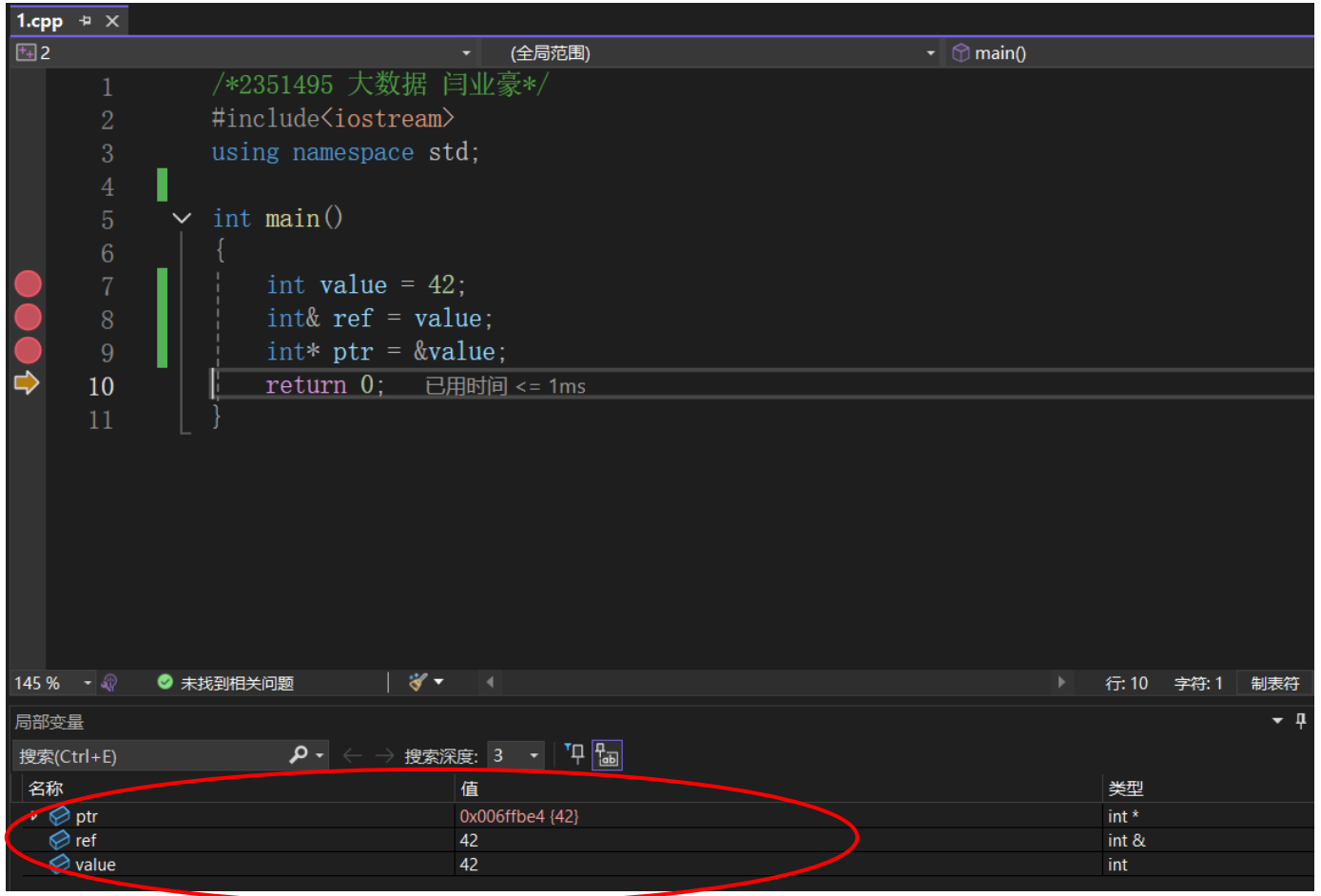
本页涉及知识点3.5、3.6

## 3.7指向字符串常量的指针

## 3.8引用类型



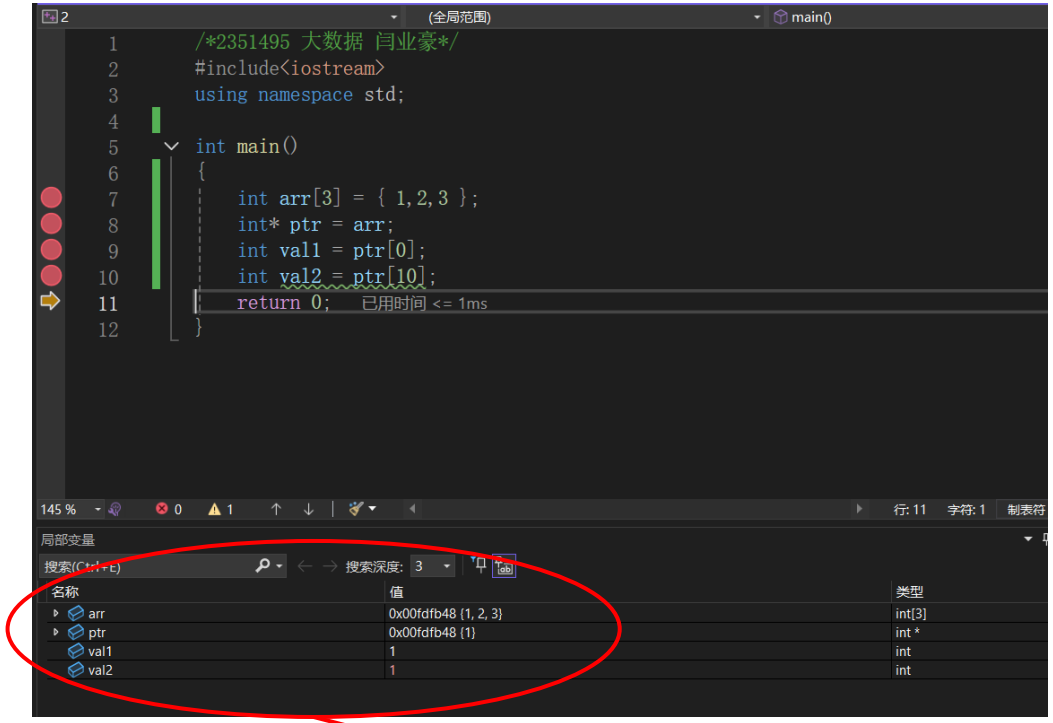

指向字符串常量的指针变量，在窗口内会显示其字符串常量的首地址，后面会跟上整个字符串常量

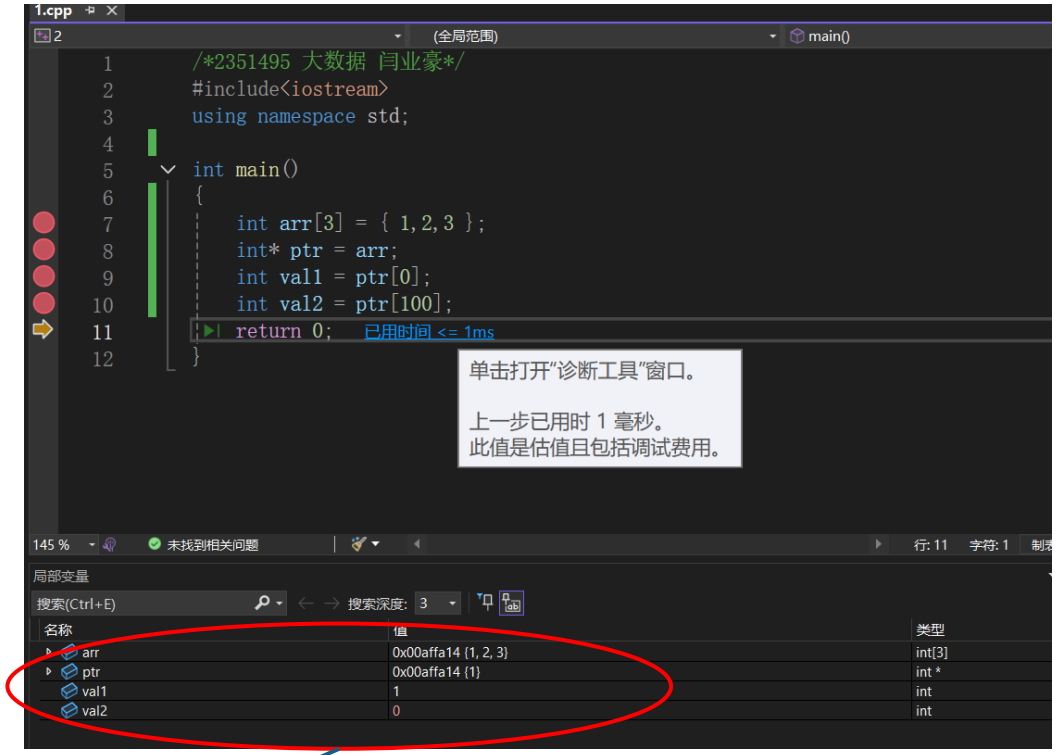



引用类型变量会显示其所引用对象的内存地址值，并在紧随的大括号{}内显示被引用变量的实际数值。由于引用本质上是目标变量的别名，调试器会将引用变量与其所引用的原始变量视为同一实体，因此在窗口中观察到的地址和数值变化与原始变量完全一致

本页涉及知识点3.7、3.8

# 3.9指针越界访问




当指针进行越界访问时，调试器仍会显示该越界位置的内存地址和相应的数值，但这些数值是不可信的。正常访问会在大括号{}内显示正确的数组元素值，而越界访问的位置可能显示随机的垃圾数据、未初始化的内存内容或其他程序数据

本页涉及知识点3.9