



VS2022 调试工具的使用 实验报告

信06 2352018 刘彦

2024/5/23



1. VS2022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?

设置断点：在代码的左侧边栏点击，设置断点。断点是程序执行到该行时会暂停的地方。

启动调试：

点击工具栏上的“开始调试”按钮（通常是一个绿色的三角形图标），或使用快捷键**F5**。

停止调试：

点击工具栏上的“停止调试”按钮（通常是一个红色的正方形图标），或使用快捷键**Shift + F5**。



本图涉及知识点1.1



本图涉及知识点1.1



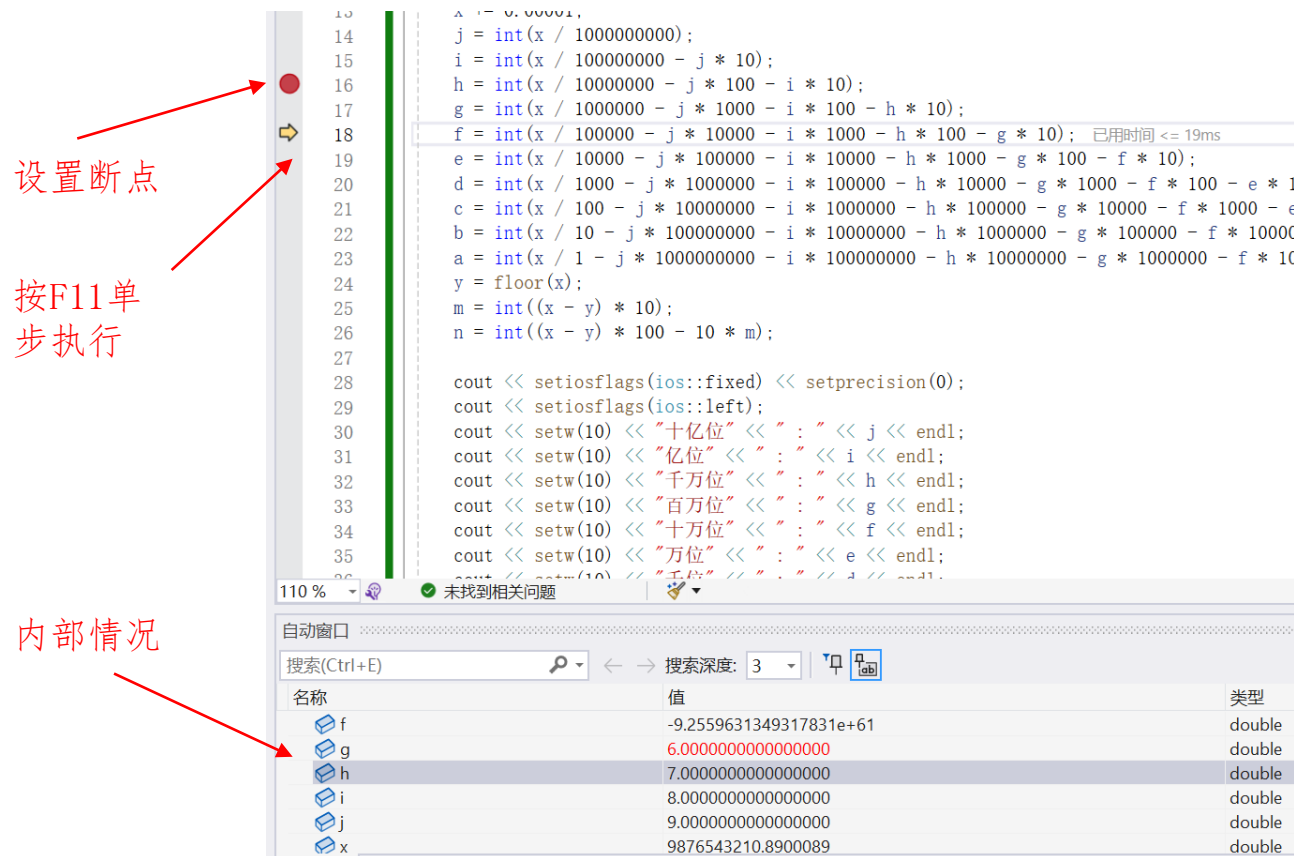
1. VS2022下调试工具的基本使用方法

1.2如何在一个函数中每个语句单步执行

调试过程中，你可以使用调试工具栏上的其他工具，比如单步执行（F10或F11），帮助更好地理解程序的执行流程和状态。

单步执行：

- 进入函数：当执行到函数调用时，你可以使用**F11**来进入函数内部进行单步调试。
- 步过函数：如果你不想进入函数内部，而是想执行完函数调用，可以使用**F10**来步过函数。
- 跳出函数：如果你已经在一个函数内部，并且想要返回到调用它的函数，可以使用**Shift + F11**。



本图涉及知识点1.2



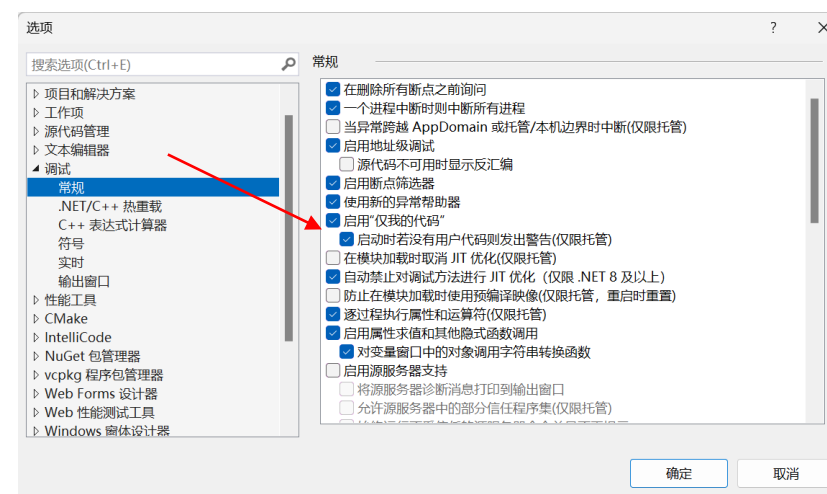
1. VS2022下调试工具的基本使用方法

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?

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?

在默认情况下, Visual Studio 的调试器不会进入系统函数或库函数的内部, 因为这些函数通常被认为是可信的. 若关闭了“仅我的代码”功能, 可以按F10, 会执行该函数, 但不会进入函数内部。

如果已经进入到cout/sqrt 等系统类/系统函数的内部, 使用快捷键**Shift + F11**可以跳出当前函数, 返回到调用它的函数的下一条可执行语句。



本图涉及知识点1.3



本图涉及知识点1.4

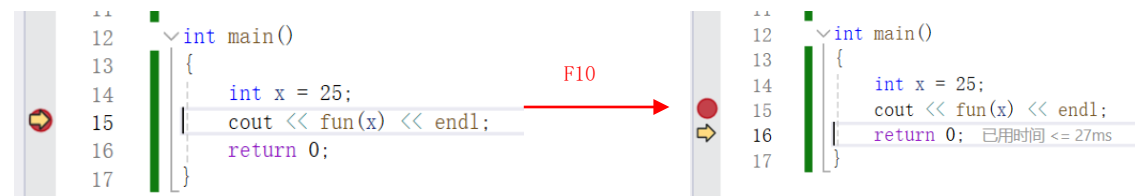


1. VS2022下调试工具的基本使用方法

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?

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?

可以按**F10**, 会执行该函数, 但不会进入函数内部。



如果已经进入到函数的内部, 使用快捷键**Shift + F11**可以跳出当前函数, 返回到调用它的函数的下一条可执行语句。

本图涉及知识点1.5



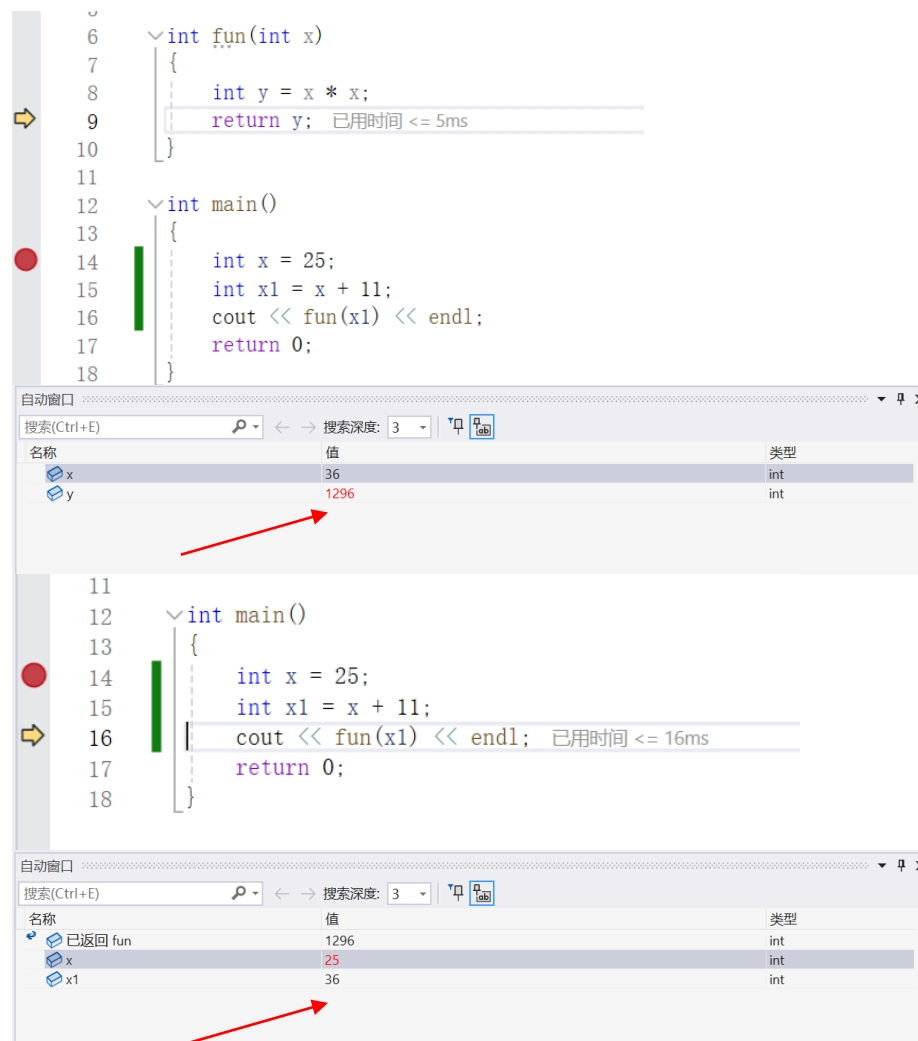
本图涉及知识点1.6



2. 用VS2022的调试工具查看各种生存期/作用域变量的方法

2.1 查看形参/自动变量的变化情况

“自动窗口”会自动显示当前执行行和上一行中使用的变量，包括形参和自动变量。



本图涉及知识点2.1



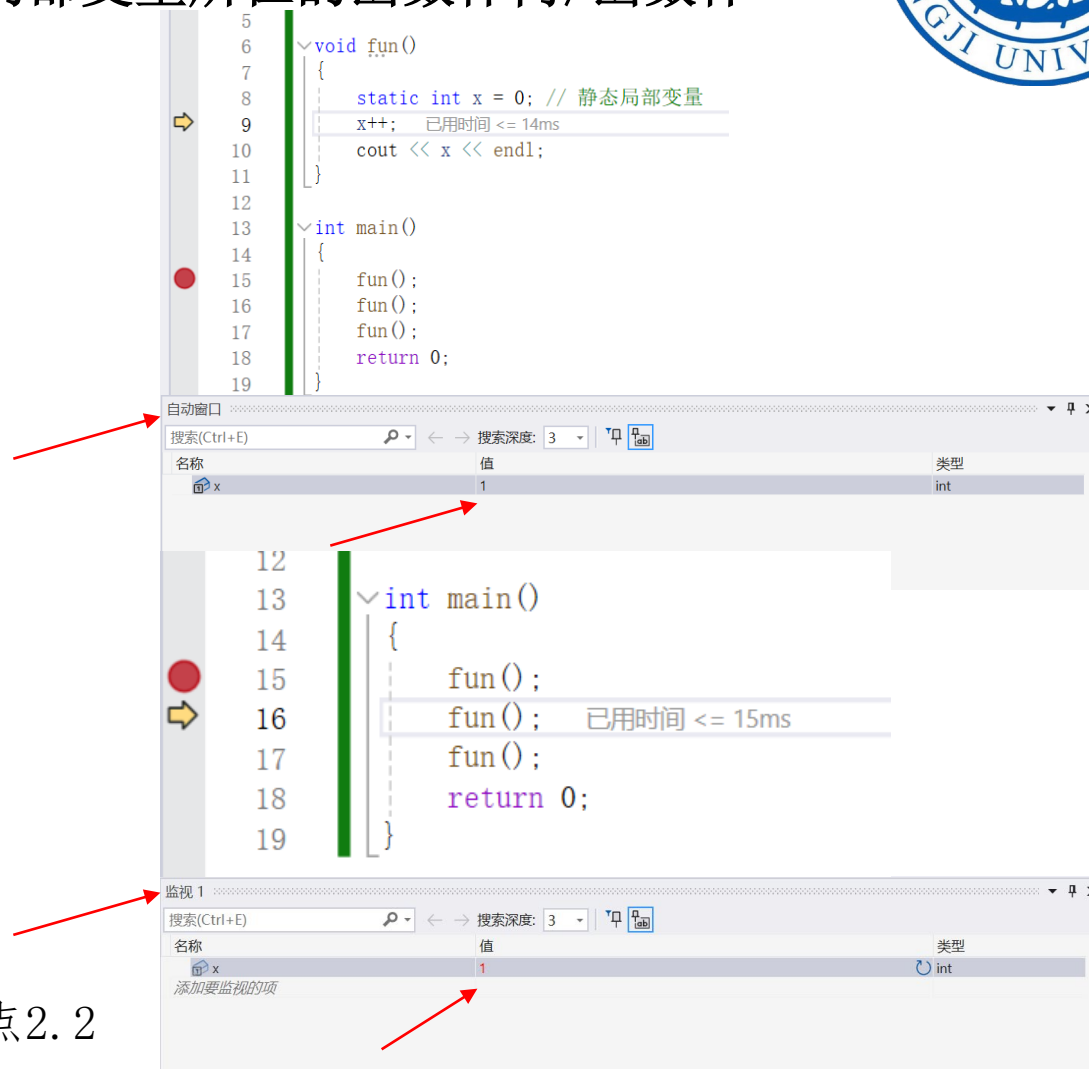
2. 用VS2022的调试工具查看各种生存期/作用域变量的方法

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)

在函数体内部时，“**自动窗口**”会自动显示当前执行行和上一行中使用的静态局部变量。

在函数体外部时，可以使用“**监视**”窗口查看该静态局部变量。

“**局部变量**”窗口也可以看到局部变量的变化情况



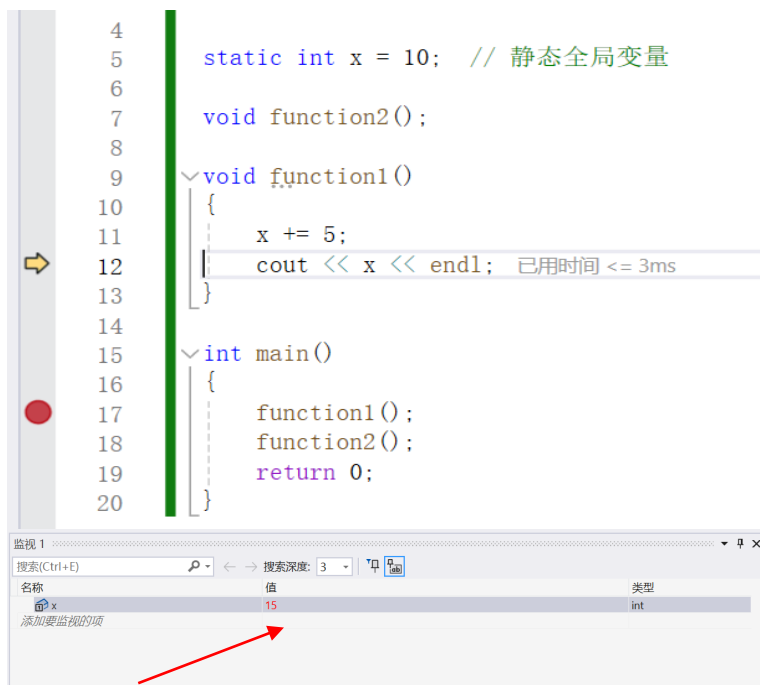
本图涉及知识点2.2



2. 用VS2022的调试工具查看各种生存期/作用域变量的方法

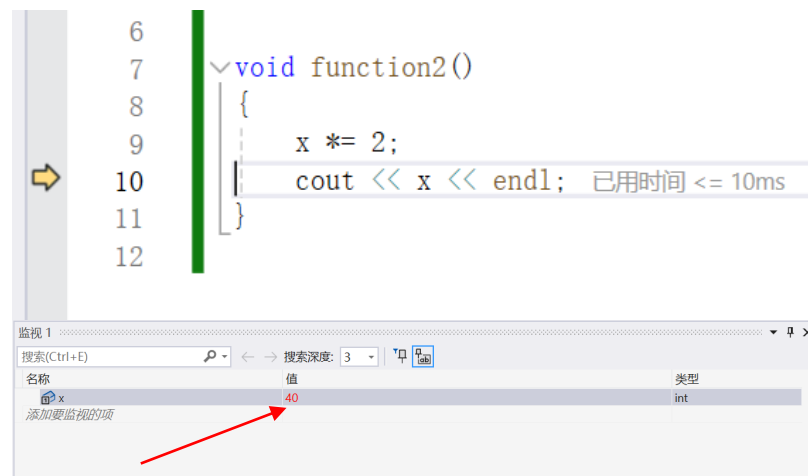
2.3 查看静态全局变量的变化情况(两个源程序文件, 有静态全局变量同名)

可以使用“**监视**”窗口查看该静态全局变量, “自动窗口”在进入函数体后才能显示该静态全局变量(故一般不用)。



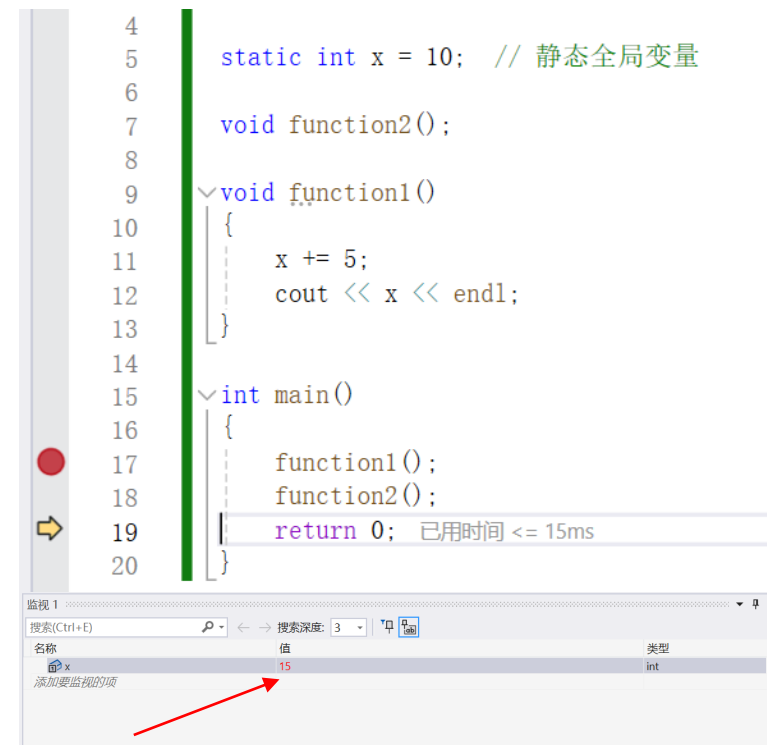
```
4 static int x = 10; // 静态全局变量
5
6 void function2();
7
8 void function1()
9 {
10     x += 5;
11     cout << x << endl; 已用时间 <= 3ms
12 }
13
14 int main()
15 {
16     function1();
17     function2();
18     return 0;
19 }
20
```

名称	值	类型
x	15	int



```
6 void function2()
7 {
8     x *= 2;
9     cout << x << endl; 已用时间 <= 10ms
10 }
11
12
```

名称	值	类型
x	40	int



```
4 static int x = 10; // 静态全局变量
5
6 void function2();
7
8 void function1()
9 {
10     x += 5;
11     cout << x << endl;
12 }
13
14 int main()
15 {
16     function1();
17     function2();
18     return 0; 已用时间 <= 15ms
19 }
20
```

名称	值	类型
x	15	int

本图涉及知识点2.3

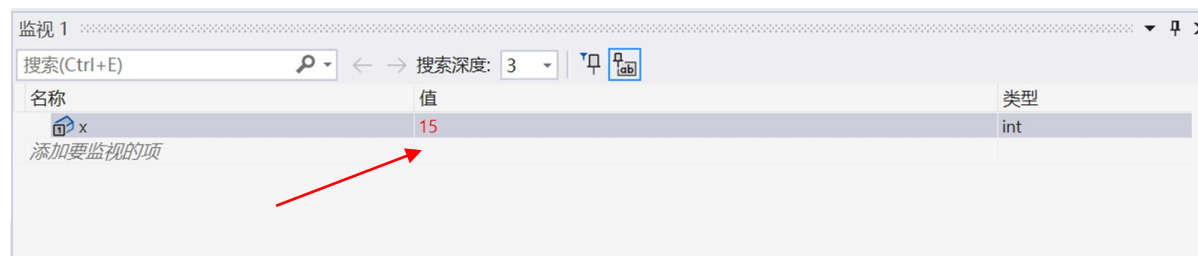


2. 用VS2022的调试工具查看各种生存期/作用域变量的方法

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)

可以使用“**监视**”窗口查看该外部全局变量, “自动窗口”在进入函数体后才能显示该外部全局变量(故一般不用)。

```
4 |
5 | int x = 10; // 外部全局变量
6 |
7 | void function2();
8 |
9 | void function1()
10 | {
11 |     x += 5;
12 |     cout << x << endl; 已用时间 <= 6ms
13 | }
14 |
15 | int main()
16 | {
17 |     function1();
18 |     function2();
19 |     return 0;
20 | }
```



本图涉及知识点2.4



3. 用VS2022的调试工具查看各种不同类型变量的方法

3.1 char/int/float 等简单变量

“自动窗口”会自动显示当前执行行和上一行中使用的简单变量。

```
4 int main()
5 {
6     char a = 'A';
7     int b = 1;
8     float c = 1.1F;
9
10    a++, b++, c++;
11    cout << a << " " << b << " " << c << endl;
12    return 0;
13
14
15
```

名称	值	类型
a	65 'A'	char
b	1	int
c	1.1000002	float

```
4 int main()
5 {
6     char a = 'A';
7     int b = 1;
8     float c = 1.1F;
9
10    a++, b++, c++;
11    cout << a << " " << b << " " << c << endl; 已用时间 <= 6ms
12    return 0;
13
14
15
```

名称	值	类型
a	66 'B'	char
b	2	int
c	2.09999990	float

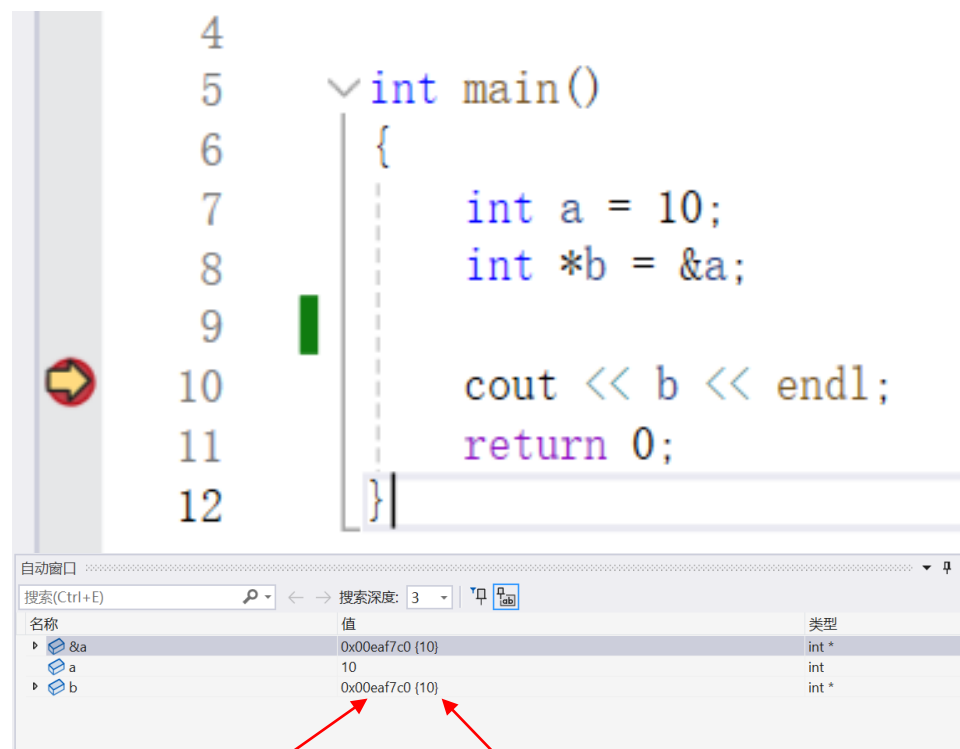
本图涉及知识点3.1



3. 用VS2022的调试工具查看各种不同类型变量的方法

3.2 指向简单变量的指针变量(如何查看地址、值?)

“自动窗口”会自动显示当前执行行和上一行中使用的指针变量, 前面是地址, 后面括号中是值。



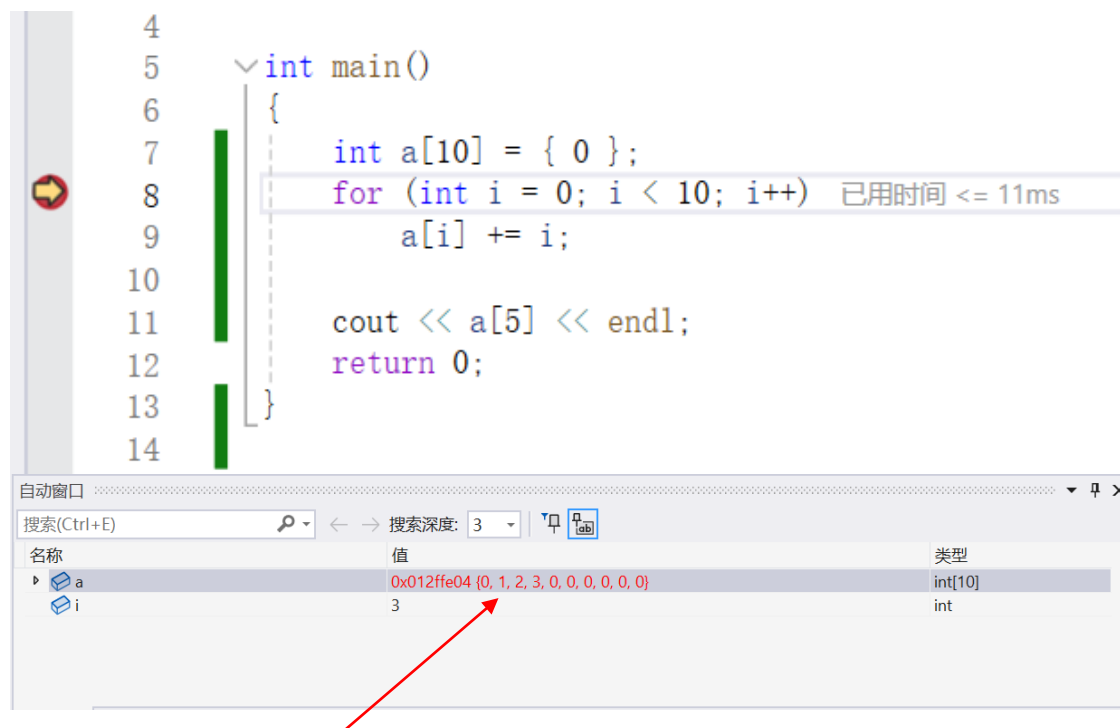
本图涉及知识点3.2



3. 用VS2022的调试工具查看各种不同类型变量的方法

3.3 一维数组

“自动窗口”会自动显示当前执行行和上一行中使用的一维数组，前面是地址，后面括号中是数组中的值。



本图涉及知识点3.3



3. 用VS2022的调试工具查看各种不同类型变量的方法

3.4 指向一维数组的指针变量(如何查看地址、值?)

“自动窗口”会自动显示当前执行行和上一行中使用的指向一维数组的指针变量, 前面是地址, 后面括号中是值。

```
4
5 int main()
6 {
7     int a[10] = { 0 };
8     int * b = a;
9     for (int i = 0; i < 10; i++)
10     {
11         a[i] += i;
12         b++; 已用时间 <= 7ms
13     }
14
15
16     cout << a[5] << endl;
17     return 0;
18 }
19
```

名称	值	类型
a	0x0098f780 {0, 1, 0, 0, 0, 0, 0, 0, 0, 0}	int[10]
a[i]	1	int
b	0x0098f784 {1}	int *
i	1	int

本图涉及知识点3.4



3. 用VS2022的调试工具查看各种不同类型变量的方法

3.5 二维数组(包括数组名仅带一个下标的情况)

“自动窗口”会自动显示当前执行行和上一行中使用的二维数组, 前面是地址, 后面括号中是数组中的值。

The screenshot shows the Visual Studio 2022 debugger interface. The code window displays the following code:

```
4
5  int main()
6  {
7      int a[10][10] = { 0 };
8      for (int i = 0; i < 10; i++)
9      {
10         a[0][i] += i;
11     } 已用时间 <= 15ms
12
13
14     cout << a[0][5] << endl;
15     return 0;
16 }
```

The Auto window at the bottom shows the following variables:

名称	值	类型
a	0x008ff574 {0x008ff574 {0, 1, 0, 0, 0, 0, 0, 0, 0, 0}, 0x008ff59c {0, 0, 0, 0, 0, 0, 0, 0, 0, 0}}	int[10][10]
a[0]	0x008ff574 {0, 1, 0, 0, 0, 0, 0, 0, 0, 0}	int[10]
a[0][i]	1	int
i	1	int

A red arrow points to the value of a[0][i] in the Auto window.

The screenshot shows the Visual Studio 2022 debugger interface. The code window displays the following code:

```
4
5  int main()
6  {
7      int a[][10] = { 0 };
8      for (int i = 0; i < 10; i++)
9      {
10         a[0][i] += i;
11     } 已用时间 <= 12ms
12
13
14     cout << a[0][5] << endl;
15     return 0;
16 }
```

The Auto window at the bottom shows the following variables:

名称	值	类型
a	0x00bbf704 {0x00bbf704 {0, 1, 2, 0, 0, 0, 0, 0, 0, 0}}	int[1][10]
a[0]	0x00bbf704 {0, 1, 2, 0, 0, 0, 0, 0, 0, 0}	int[10]
a[0][i]	2	int
i	2	int

A red arrow points to the value of a[0][i] in the Auto window.

本图涉及知识点3.5



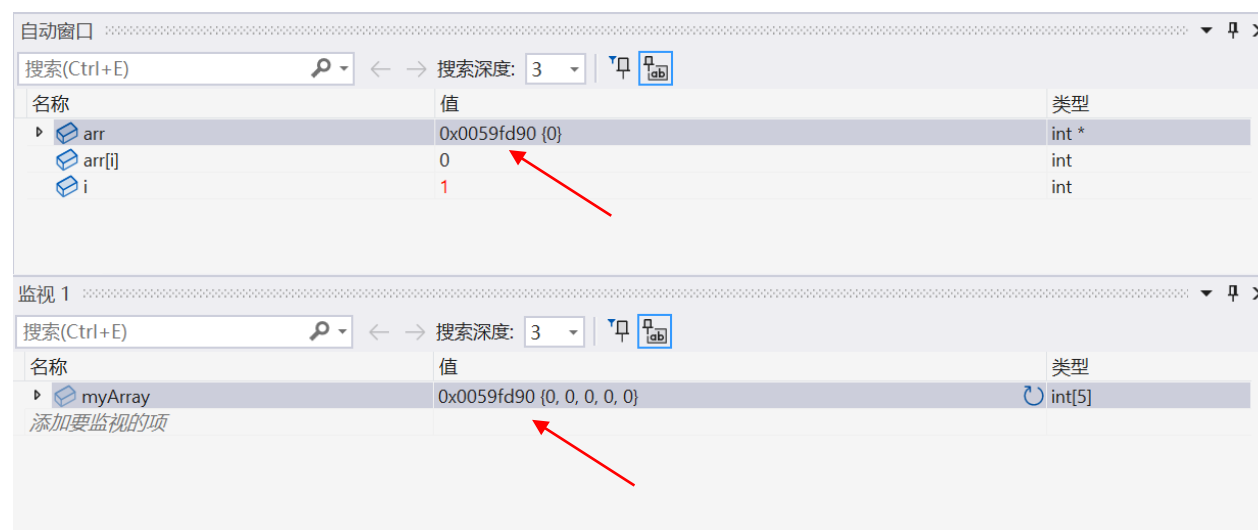
3. 用VS2022的调试工具查看各种不同类型变量的方法

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?

“自动窗口”会自动显示当前执行行和上一行中使用的指针, 可以在“自动窗口”通过指向数组的指针查看其地址。

可以使用“监视”窗口查看该数组的地址和值, 前面是地址, 后面括号中是值。

```
5
6 void fun(int * arr)
7 {
8     for (int i = 0; i < 5; ++i)
9     {
10        arr[i] += i; 已用时间 <= 9ms
11    }
12 }
13
14 int main()
15 {
16     int myArray[5] = { 0 };
17
18     fun(myArray);
19
20     cout << myArray[1];
21     return 0;
22 }
```



本图涉及知识点3.6



3. 用VS2022的调试工具查看各种不同类型变量的方法

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?)

“自动窗口”会自动显示当前执行行和上一行中使用的指针,可以在“自动窗口”通过指向数组的指针查看其地址。
可以使用“监视”窗口查看该字符串的地址, &myString[0] 会获取到这个字符串字面量的第一个字符的地址。

```
5
6 int main()
7 {
8     const char *myString = "Hello, World!";
9
10    return 0; 已用时间 <= 2ms
11 }
```

名称	值	类型
myString	0x00b69bd0 "Hello, World!"	const char *

名称	值	类型
&myString[0]	0x00b69bd0 "Hello, World!"	const char *

本图涉及知识点3.7

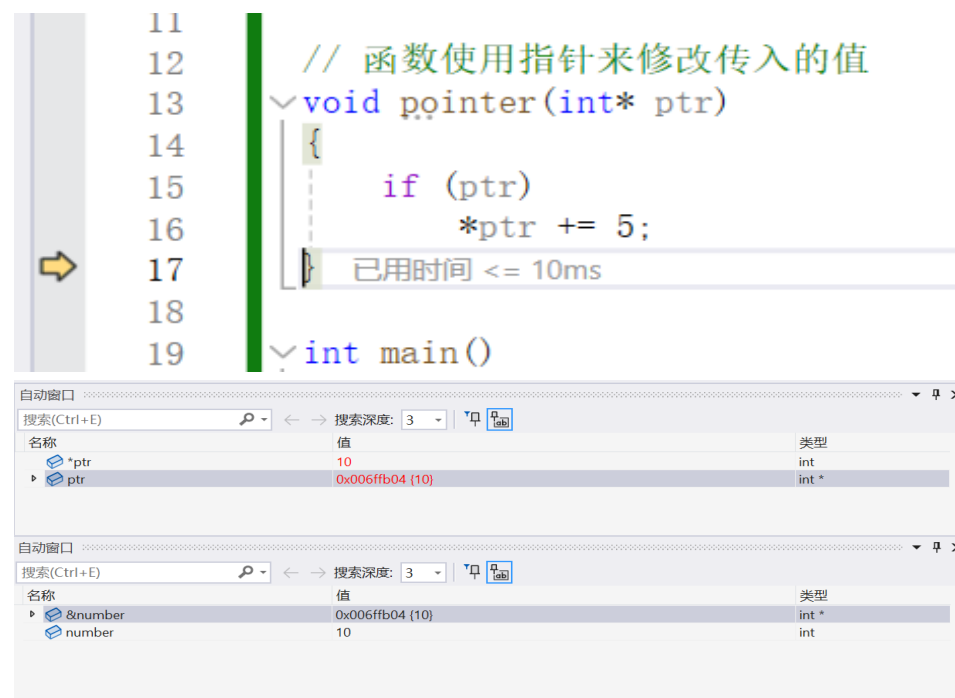
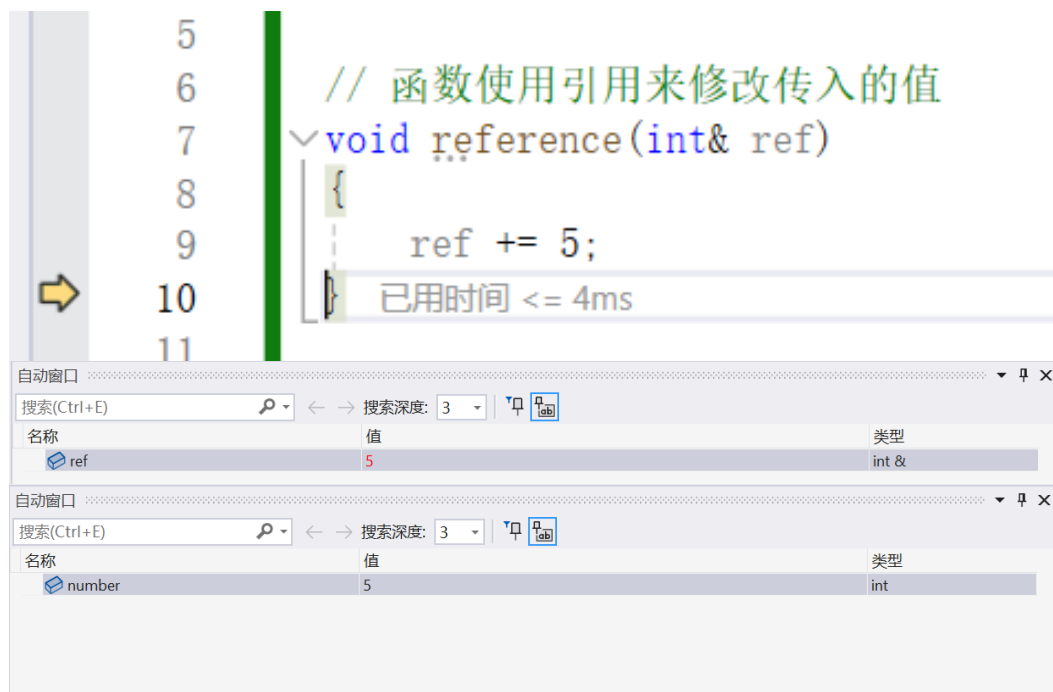


3. 用VS2022的调试工具查看各种不同类型变量的方法

3.8 引用(引用与指针是否有区别?有什么区别?)

“自动窗口”会自动显示当前执行行和上一行中使用的变量,可以在“自动窗口”查看引用的变量。由于引用是原始变量的别名,所以在调试时,实际上是在查看原始变量的值和地址。

引用相当于直接对变量进行改变,指针相当于改变指针变量,再将其地址赋给变量的地址。



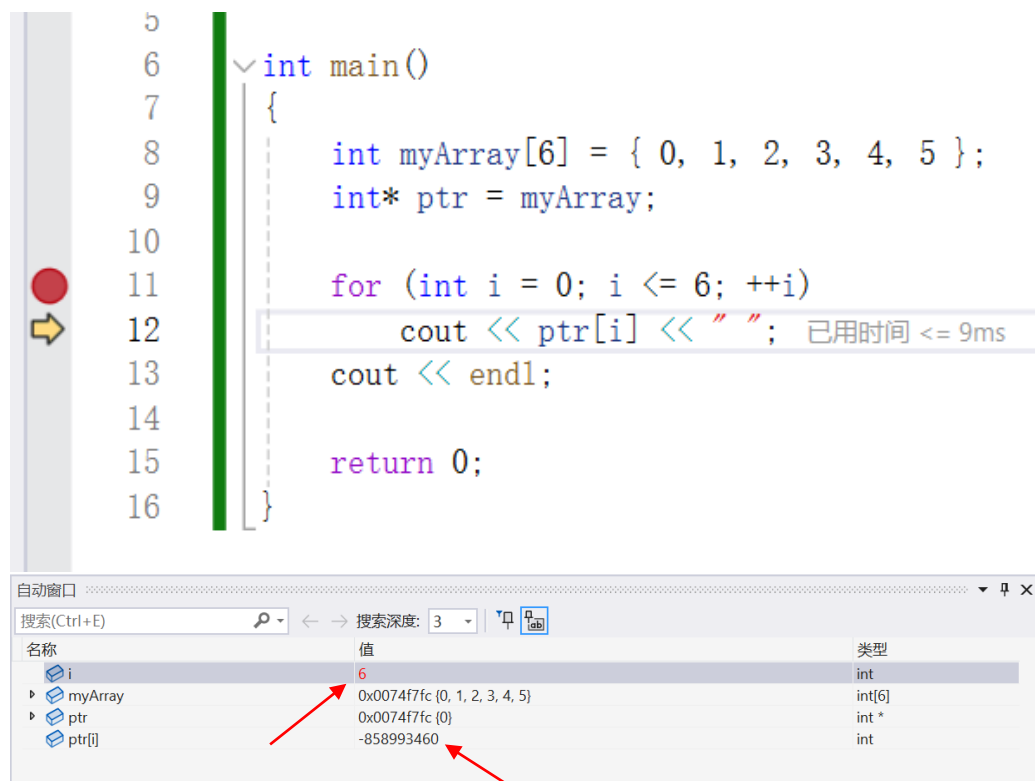
本图涉及知识点3.8



3. 用VS2022的调试工具查看各种不同类型变量的方法

3.9 使用指针时出现了越界访问

“自动窗口”会自动显示当前执行行和上一行中使用的变量,可以在“自动窗口”查看指针出现越界访问时所指向数组的值(这个值是非正常值)。



本图涉及知识点3.9