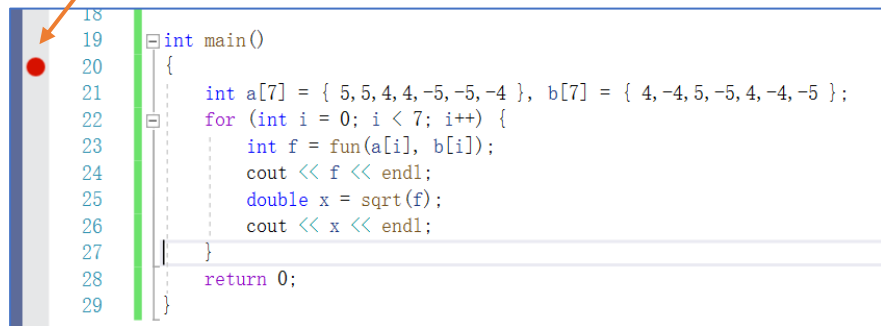


# VS2019调试工具使用报告

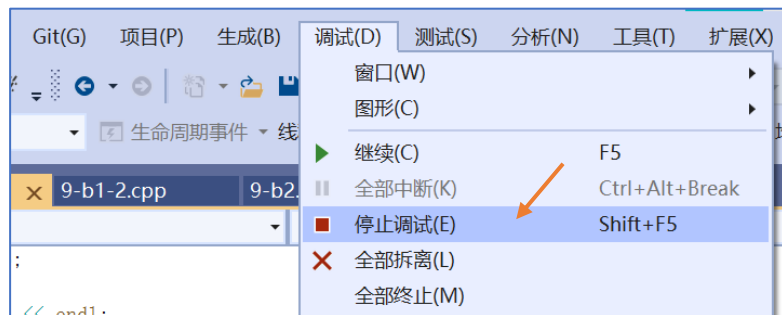
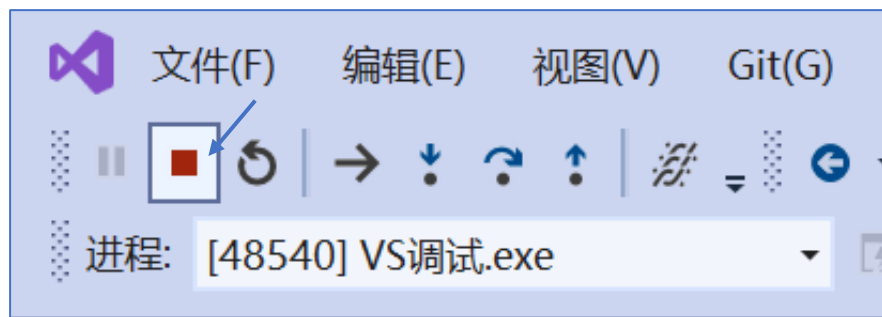
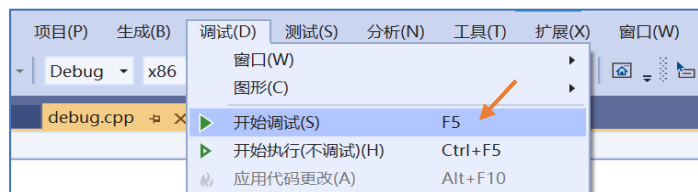
2151140 信07 王谦

2021.12.28

## 1.1 开始结束调试



```
18
19 int main()
20 {
21     int a[7] = { 5, 5, 4, 4, -5, -5, -4 }, b[7] = { 4, -4, 5, -5, 4, -4, -5 };
22     for (int i = 0; i < 7; i++) {
23         int f = fun(a[i], b[i]);
24         cout << f << endl;
25         double x = sqrt(f);
26         cout << x << endl;
27     }
28     return 0;
29 }
```



开始:

- 1.鼠标左键双击此位置设置断点 (F9)
- 2.调试菜单中点击开始调试(F5)

结束:

点击上栏中如图所示图标

或者调试菜单中点击停止调试(shift+F5)

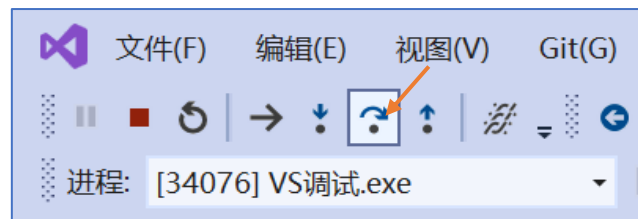
## 1.2 每个语句单步执行



点击上栏中如图所示图标

或者点击调试菜单项中的逐语句 (F11)

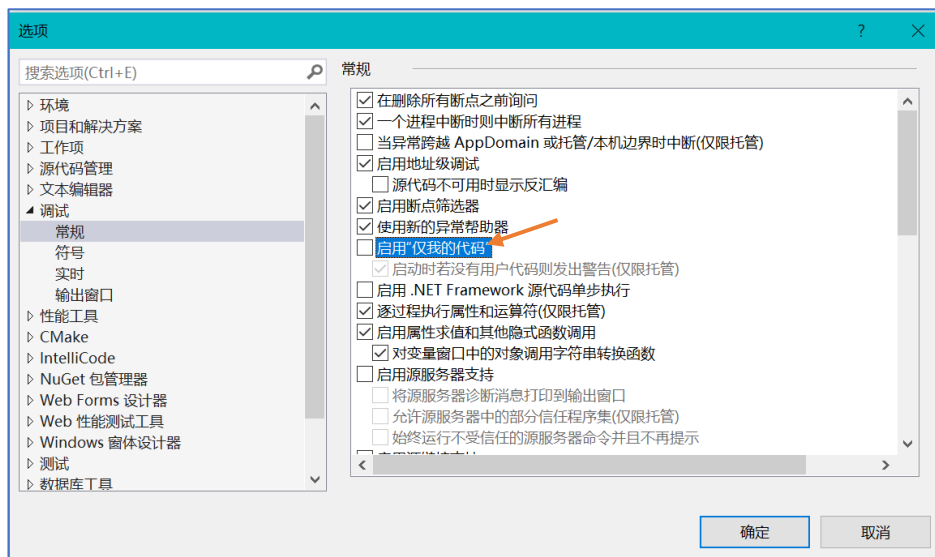
## 1.5 不进入自定义函数内部



点击上栏中如图所示图标

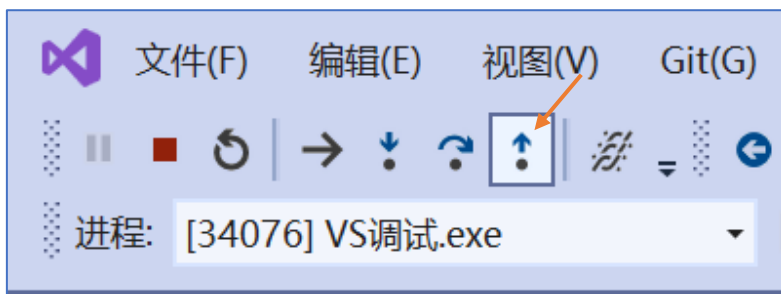
或者点击调试菜单项中的逐过程 (F10)

## 1.3/1.4 不进入/跳出 系统函数内部



不进入：

在选项中取消 启用“仅我的代码”



跳出：

点击上栏中如图所示图标

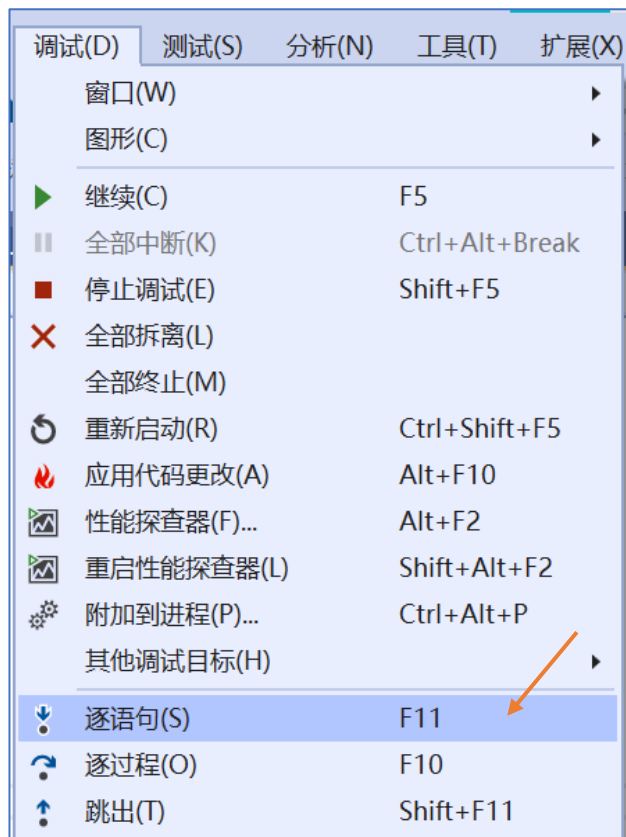
或者调试菜单中点击跳出(shift+F11)

## 1.6 转入自定义函数单步执行



点击上栏中如图所示图标

或者点击调试菜单项中的逐语句 (F11)



- 2.1 查看形参、自动变量
- 2.2 查看静态局部变量
- 2.3 查看静态全局变量
- 2.4 查看外部全局变量

监视 1

搜索(Ctrl+E)    搜索深度: 3

名称	值	类型
a[i]	未定义标识符 "i"	
f	未定义标识符 "f"	
b[i]	未定义标识符 "i"	
i	未定义标识符 "i"	
x	未定义标识符 "x"	
c	未定义标识符 "c"	
d	未定义标识符 "d"	
s	未定义标识符 "s"	
W	20	int
Z	0	int

添加要监视的项

a[i]	4	int
f	9	int
b[i]	5	int
i	2	int
x	2.23606801	float
c	9	int
d	4	int
s	1	int
W	100	int
Z	10	int

添加要监视的项

```
#include <iostream>
#include <cmath>
using namespace std;

static int Z;
extern int W;

int fun(int d, int b)
{
    static int s = 0;
    int c = d + b;
    if (c > d) {
        cout << c << endl;
        s = 1;
        Z = 10;
        W = 100;
        return c;
    }
}
```

```
int main()
{
    int a[7] = { 5, 5, 4, 4, -5, -5, -4 }, b[7];
    for (int i = 0; i < 7; i++) {
        int f = fun(a[i], b[i]);
        cout << f << endl; 已用时间 <= 1ms
        float x = sqrt(f);
        cout << x << endl;
    }

    return 0;
}
```

debug.cpp\* 2.cpp

VS调试

debug.cpp 2.cpp 9-b

VS调试

1	#include <iostream>
2	#include <cmath>
3	using namespace std;
4	
5	int W = 20;
6	static int Z = 10;

添加相应的变量在箭头所示位置便可监视，出现叉号表示变量不在生存期，符号颜色黯淡表示不在作用域，颜色深色则处于作用域且处于生存期。

- 1.形参和自动变量生存期和作用域都限于所在的{}内（本函数内）
- 2.静态局部变量生存期为自其出现直到程序结束，但作用域只在所处的{}内（本函数内）
- 3.静态全局变量生存期为自其出现直到程序结束，作用域为本源程序文件
- 4.外部全局变量生存期为自其出现直到程序结束，作用域为全部源程序文件

- 3.1 char/int/float
- 3.2 指向简单变量的指针
- 3.3 一维数组
- 3.4 指向一维数组的指针
- 3.5 二维数组
- 3.6 实参数组，形参指针
- 3.7 指向字符串常量的指针
- 3.8 引用
- 3.9指针越界

```
#include <iostream>
#include <cmath>
using namespace std;

void fun(char* p_ch)
{
    cout << *(p_ch + 2) << endl;
}
```

```
int main()
{
    int i, *p_i = &i;
    float f, *p_f = &f;
    char ch[5], *p_ch = ch;
    int j[5] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    int& t = i;

    i = 2;
    *p_i = 3;
    t = 4;
    f = 2.1;
    *p_f = 2.3;
```

```
for (int k = 0; k < 5; k++) {
    ch[k] = k + 65;
}
for (int k = 0; k < 5; k++) {
    *(p_ch + k) = k + 97;
}
char c = *(p_ch + 5);
cout << c << endl;
fun(ch); 已用时间 <= 1ms

return 0;
```

名称	值	类型
i	1	int
p_i	0x010ffae0 (17824492)	int *
f	17824492	int
p_f	0.00000000	float
ch	ucrtbased.dll!0x7751d033 (加载符号以获取其他信息) (4.03125...	float *
ch[0]	4.03125246e+18	float
ch[1]	0x010ffaa4 "\x2"	char[5]
ch[2]	2 '\x2'	char
ch[3]	0 '\0'	char
ch[4]	0 '\0'	char
ch[k]	32 ''	char
c	未定义标识符 "k"	char
t	119 'w'	char
p_ch	17824392	int &
	0x00000000 <NULL>	char *
	<无法读取内存>	char

名称	值	类型
i	4	int
p_i	0x010ffad8 {4}	int *
f	4	int
p_f	2.29999995	float
ch	0x010ffac0 (2.29999995)	float *
ch[0]	2.29999995	float
ch[1]	0x010ffaa4 "abcde...	char[5]
ch[2]	97 'a'	char
ch[3]	98 'b'	char
ch[4]	99 'c'	char
ch[k]	100 'd'	char
c	101 'e'	char
t	101 'e'	char
p_ch	-52 '?'	char
	4	int &
	0x010ffaa4 <字符串中的字符无效。>	char *
	97 'a'	char

1-5.监视列表中值的一栏可以看到当前的变量取值，类型栏可以看到类型，指针可以在值的一栏看到地址与值两行

6.形参指针指代实参数组的地址和值

7.指针能看到无名字符串常量的地址

8.指针有自己的独立空间，只是指代地址与值，与原变量不同；引用只不过是给变量加的一个新名字，两者本就是同一个量

越界访问的值会出现乱值

ch	241 '?'	unsigned char
ch[k]	101 'e'	char
c	-52 '?'	char
t	4	int &
p_ch	0x009cfc98 "abcde烫烫烫烫烫檀顯"	char *
	97 'a'	char