

Visual Studio的程序调试

目录

- 为什么要程序调试？
- 通过printf做程序调试
- Visual Studio的断点调试
- Visual Studio的逐语句/逐过程调试

为什么要程序调试

- **语法错误**：程序编译时通常能检测出各种语法错误，这时编程者需要根据出错信息，找出程序中的语法错误，修改程序并消除语法错误，重新编译，如果操作，直到编译成功，生成可执行程序（.exe）。
- **逻辑错误**：即使程序能够运行，也并不意味着它就是正确的，我们需要仔细分析程序运行结果，看是否符合预期。如果不符合预期，很有可能是程序中还含有逻辑错误，需要修改程序并消除其中的逻辑错误。

为什么要程序调试

- 编译时可以检查出源代码中的**语法错误**，但是检查不出源代码中的**逻辑错误**。

```
1  /*****
2  程序名：对从1到100的整数求和
3  说明：本程序有错误，供调试练习用
4  *****/
5  #include <stdio.h>
6
7  int main() {
8      int sum = 0;
9      int k;
10     for (k = 1; k < 100; k++) {
11         sum += k;
12         //printf("k = %d\tsum = %d\n", k, sum);
13     }
14     printf("sum = %d\n", sum);
15     return 0;
16 }
```

求1~100之间所有整数的和

Microsoft Visual Studio 调试控制台

sum = 4950

E:\VS Projects\OJtest\Debug\OJtest.exe (进程 31368)已退出，代码为 0。
按任意键关闭此窗口...

为什么要程序调试

- **程序调试 (debug)**：要想清除程序中的逻辑错误，就必须学会一些调试方法，通过观察**变量的数值变化**，加上自己的逻辑思考，从而找出程序中的错误所在并修改清除之。

目录

- 为什么要程序调试？
- 通过printf做程序调试
- Visual Studio的断点调试
- Visual Studio的逐语句/逐过程调试

通过printf做程序调试

- 一种较简单的方法，是在程序中添加输出语句printf，输出运行过程中的变量的中间值。

```
1  /******  
2  程序名：对从1到100的整数求和  
3  说明：本程序有错误，供调试练习用  
4  *****/  
5  #include <stdio.h>  
6  
7  int main() {  
8      int sum = 0;  
9      int k;  
10     for (k = 1; k < 100; k++) {  
11         sum += k;  
12         //printf("k = %d\tsum = %d\n", k, sum);  
13     }  
14     printf("sum = %d\n", sum);  
15     return 0;  
16 }
```

去掉注释，输出变量 k 和 sum 在循环过程中的值

通过printf做程序调试

- 一种较简单的方法，是在程序中添加输出语句printf，输出运行过程中的变量的中间值。

```
Microsoft Visual Studio 调试控制台
k = 74    sum = 2775
k = 75    sum = 2850
k = 76    sum = 2926
k = 77    sum = 3003
k = 78    sum = 3081
k = 79    sum = 3160
k = 80    sum = 3240
k = 81    sum = 3321
k = 82    sum = 3403
k = 83    sum = 3486
k = 84    sum = 3570
k = 85    sum = 3655
k = 86    sum = 3741
k = 87    sum = 3828
k = 88    sum = 3916
k = 89    sum = 4005
k = 90    sum = 4095
k = 91    sum = 4186
k = 92    sum = 4278
k = 93    sum = 4371
k = 94    sum = 4465
k = 95    sum = 4560
k = 96    sum = 4656
k = 97    sum = 4753
k = 98    sum = 4851
k = 99    sum = 4950
sum = 4950
```

要解决程序中的错误，
就需要把 for 循环中的
“k < 100” 改为
“k <= 100”

通过printf做程序调试

- 小结：即使程序能运行，也需要仔细查看运行结果，判断运行结果**是否符合预期**。如果不符合预期，有可能是程序中含有逻辑错误；可以在程序中添加输出语句，输出运行过程中的变量的中间值。**观察变量的值的变化情况**，判断程序错误的原因，然后修改程序消除错误。

目录

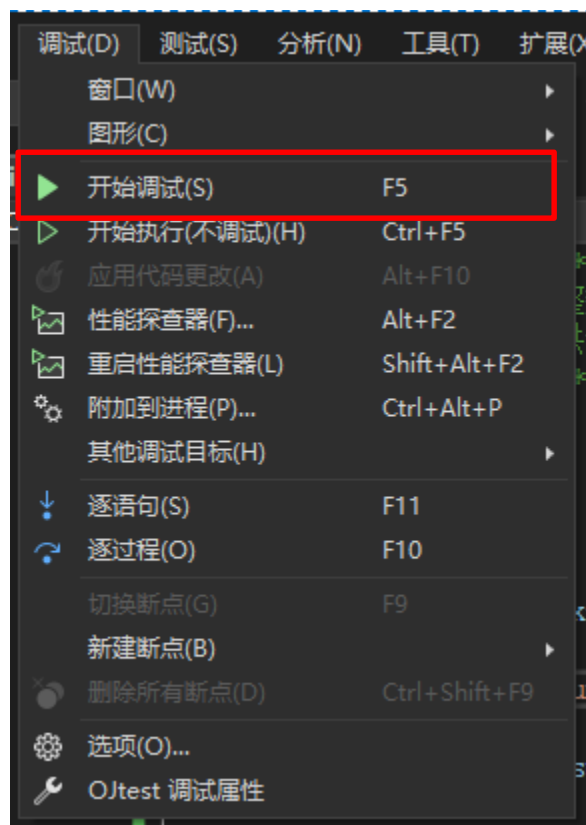
- 为什么要程序调试？
- 通过printf做程序调试
- **Visual Studio的断点调试**
- Visual Studio的逐语句/逐过程调试

Visual Studio的断点调试

- 上面程序通过printf运行过程中的变量的中间值可以找出问题并排错，但是有些程序的错误可能比较隐藏比较深，这时就需要更复杂一点的方法，也就是使用Visual Studio所提供的排错和调试工具，**包括两大类：断点调试、逐语句/过程调试。**

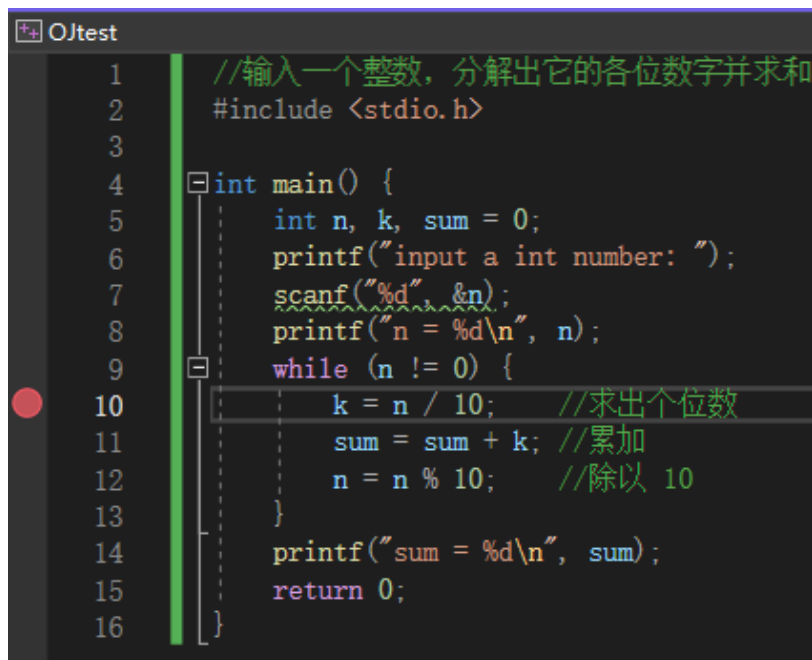
Visual Studio的断点调试

- 设置断点并调试：使用“设置断点”（F9），“开始调试”（F5）、“停止调试”（Shift+F5）这三个功能。



Visual Studio的断点调试

- 如何设置断点，有两种方式
 - 光标移动到你觉得可能出差那一行，并按F9
 - 在你觉得可能出错那一行的左侧，单击鼠标

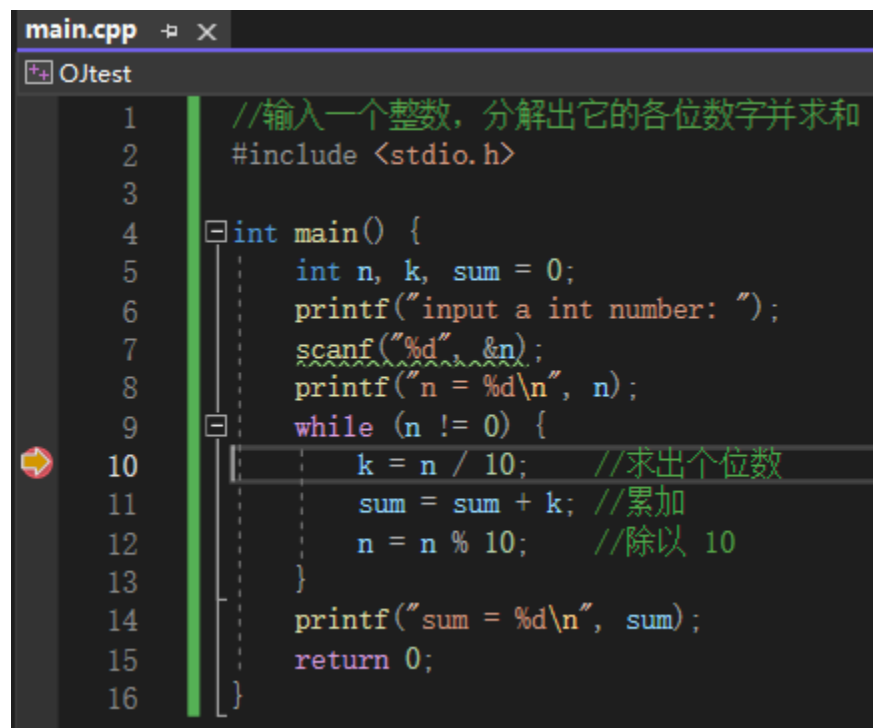


The screenshot shows the Visual Studio IDE with a C program open. The program is designed to calculate the sum of the digits of an integer. A breakpoint, represented by a red dot, is set on the left margin of line 10. The code is as follows:

```
OJtest
1 //输入一个整数，分解出它的各位数字并求和
2 #include <stdio.h>
3
4 int main() {
5     int n, k, sum = 0;
6     printf("input a int number: ");
7     scanf("%d", &n);
8     printf("n = %d\n", n);
9     while (n != 0) {
10        k = n / 10; //求出个位数
11        sum = sum + k; //累加
12        n = n % 10; //除以 10
13    }
14    printf("sum = %d\n", sum);
15    return 0;
16 }
```

Visual Studio的断点调试

- 设置断点后，按F5，程序会直接跳到设置断点那一行，并在红点上出现一个黄色的小箭头



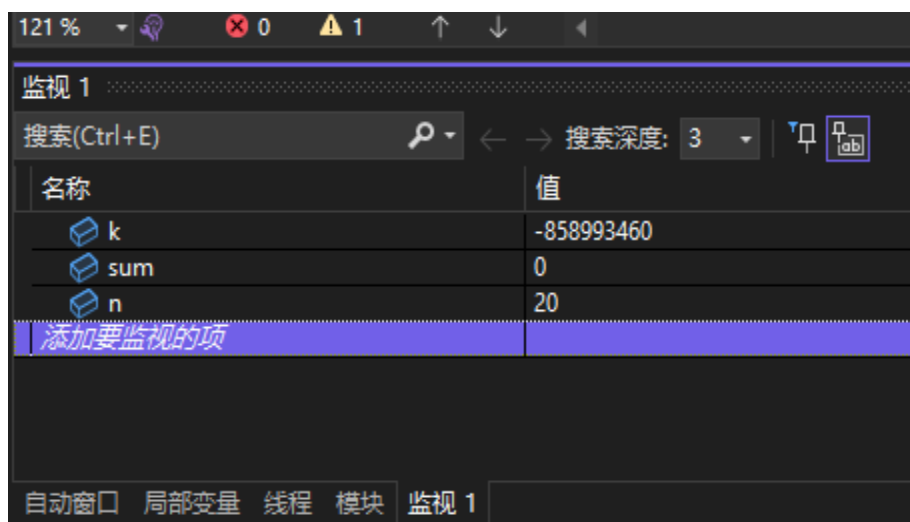
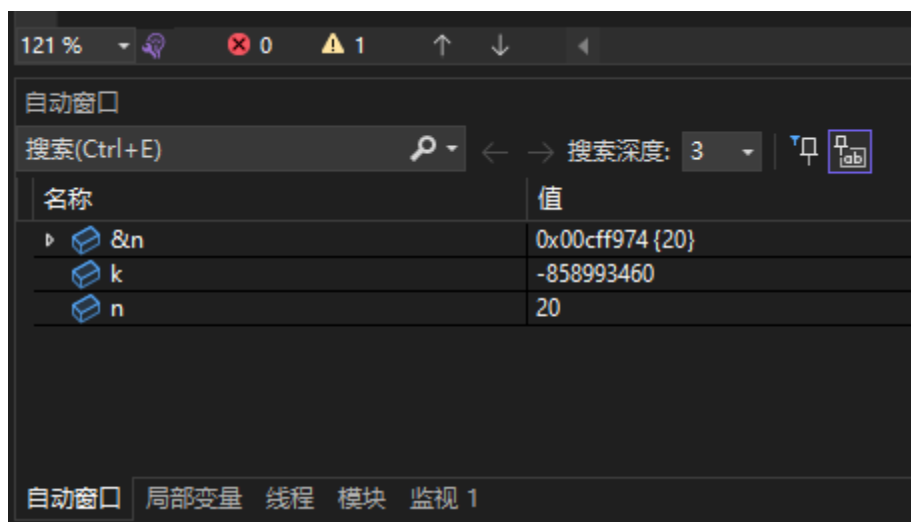
The screenshot shows the Visual Studio IDE with a C++ file named `main.cpp` open. The code is as follows:

```
1 //输入一个整数，分解出它的各位数字并求和 (
2 #include <stdio.h>
3
4 int main() {
5     int n, k, sum = 0;
6     printf("input a int number: ");
7     scanf("%d", &n);
8     printf("n = %d\n", n);
9     while (n != 0) {
10         k = n / 10; //求出个位数
11         sum = sum + k; //累加
12         n = n % 10; //除以 10
13     }
14     printf("sum = %d\n", sum);
15     return 0;
16 }
```

A breakpoint is set at line 10, indicated by a red dot in the left margin. A yellow arrow points to this red dot, signifying that the program has paused at the breakpoint.

Visual Studio的断点调试

- 这时，可以通过左下方的窗口（自动窗口、监视），查看变量的值。在监视窗口中，可以自己输入想要查看的值。



- 找到错误后，按Shift+F5退出调试模式

目录

- 为什么要程序调试？
- 通过printf做程序调试
- Visual Studio的断点调试
- **Visual Studio的逐语句/逐过程调试**

Visual Studio逐语句/逐过程调试

- 有时候不确定那句可能出错，就要从头到尾一条一条第查，就需要用到逐语句/逐过程调试
- 逐语句调试：按F11，从main函数第一条语句开始执行，每按一次F11，**向后执行一条语句**；如果遇到函数调用，则**进入函数中的语句**；可以**随时停下来查看变量值**。
- 逐过程调试：按F10，从main函数第一条语句开始执行，每按一次F10，**向后执行一条语句**；如果遇到函数调用，**不进入函数中的语句**；可以随时停下来查看变量值。

Visual Studio逐语句/逐过程调试

- 逐语句调试：按F11

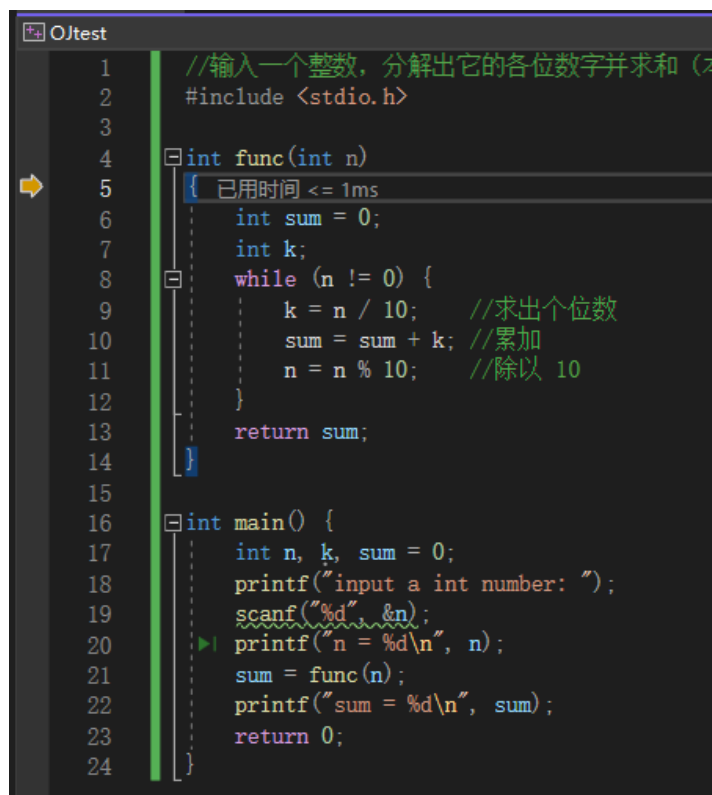
```
1 //输入一个整数，分解出它的各位数字并求和 (
2 #include <stdio.h>
3
4 int func(int n)
5 {
6     int sum = 0;
7     int k;
8     while (n != 0) {
9         k = n / 10; //求出个位数
10        sum = sum + k; //累加
11        n = n % 10; //除以 10
12    }
13    return sum;
14 }
15
16 int main() {
17     int n, k, sum = 0;
18     printf("input a int number: ");
19     scanf("%d", &n);
20     printf("n = %d\n", n);
21     sum = func(n); 已用时间 <= 1ms
22     printf("sum = %d\n", sum);
23     return 0;
24 }
```



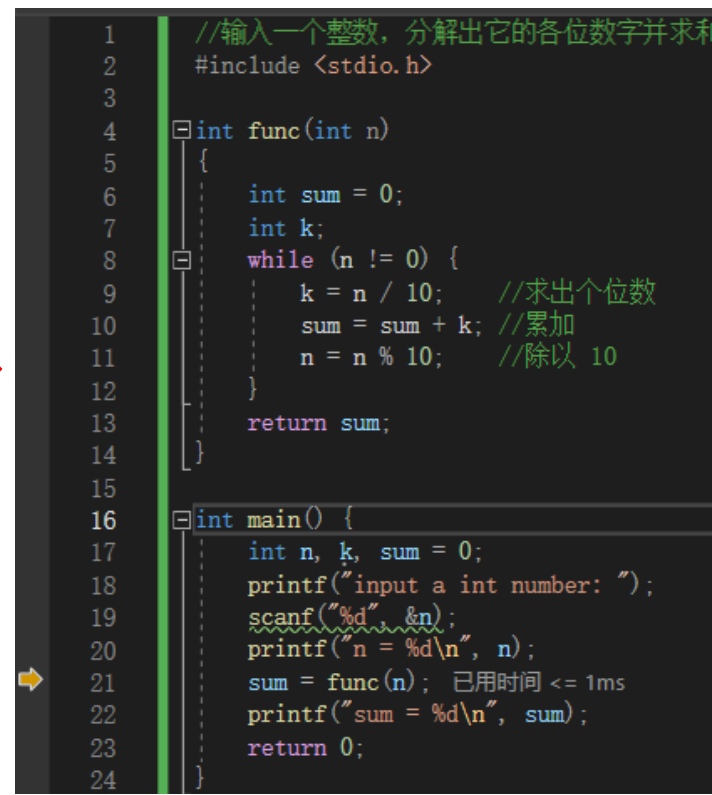
```
O/test
1 //输入一个整数，分解出它的各位数字并求和 (
2 #include <stdio.h>
3
4 int func(int n)
5 { 已用时间 <= 1ms
6     int sum = 0;
7     int k;
8     while (n != 0) {
9         k = n / 10; //求出个位数
10        sum = sum + k; //累加
11        n = n % 10; //除以 10
12    }
13    return sum;
14 }
15
16 int main() {
17     int n, k, sum = 0;
18     printf("input a int number: ");
19     scanf("%d", &n);
20     printf("n = %d\n", n);
21     sum = func(n);
22     printf("sum = %d\n", sum);
23     return 0;
24 }
```

Visual Studio逐语句/逐过程调试

- 逐语句调试：按Shift+F11跳出这个函数



```
1 //输入一个整数，分解出它的各位数字并求和 (2)
2 #include <stdio.h>
3
4 int func(int n)
5 { 已用时间 <= 1ms
6     int sum = 0;
7     int k;
8     while (n != 0) {
9         k = n / 10; //求出个位数
10        sum = sum + k; //累加
11        n = n % 10; //除以 10
12    }
13    return sum;
14 }
15
16 int main() {
17     int n, k, sum = 0;
18     printf("input a int number: ");
19     scanf("%d", &n);
20     printf("n = %d\n", n);
21     sum = func(n);
22     printf("sum = %d\n", sum);
23     return 0;
24 }
```



```
1 //输入一个整数，分解出它的各位数字并求和
2 #include <stdio.h>
3
4 int func(int n)
5 {
6     int sum = 0;
7     int k;
8     while (n != 0) {
9         k = n / 10; //求出个位数
10        sum = sum + k; //累加
11        n = n % 10; //除以 10
12    }
13    return sum;
14 }
15
16 int main() {
17     int n, k, sum = 0;
18     printf("input a int number: ");
19     scanf("%d", &n);
20     printf("n = %d\n", n);
21     sum = func(n); 已用时间 <= 1ms
22     printf("sum = %d\n", sum);
23     return 0;
24 }
```

Visual Studio逐语句/逐过程调试

- 逐过程调试：按F10，不进入函数，但是函数中的语句照样执行，只是我们不用跟踪进去

```
1 //输入一个整数，分解出它的各位数字并求和 (本  
2 #include <stdio.h>  
3  
4 int func(int n)  
5 {  
6     int sum = 0;  
7     int k;  
8     while (n != 0) {  
9         k = n / 10; //求出个位数  
10        sum = sum + k; //累加  
11        n = n % 10; //除以 10  
12    }  
13    return sum;  
14 }  
15  
16 int main() {  
17     int n, k, sum = 0;  
18     printf("input a int number: ");  
19     scanf("%d", &n);  
20     printf("n = %d\n", n);  
21     sum = func(n); 已用时间 <= 1ms  
22     printf("sum = %d\n", sum);  
23     return 0;  
24 }
```



```
1 //输入一个整数，分解出它的各位数字并求和 (本  
2 #include <stdio.h>  
3  
4 int func(int n)  
5 {  
6     int sum = 0;  
7     int k;  
8     while (n != 0) {  
9         k = n / 10; //求出个位数  
10        sum = sum + k; //累加  
11        n = n % 10; //除以 10  
12    }  
13    return sum;  
14 }  
15  
16 int main() {  
17     int n, k, sum = 0;  
18     printf("input a int number: ");  
19     scanf("%d", &n);  
20     printf("n = %d\n", n);  
21     sum = func(n);  
22     printf("sum = %d\n", sum); 已用时间 <= 1ms  
23     return 0;  
24 }
```

总结

● 小结：

- ✓ 在调试过程中，“逐过程”（F10）是指把当前语句作为一步执行完毕，而“逐语句”（F11）是指如果当前语句中含有函数调用则追踪进入到函数中去执行。
- ✓ 在调试过程中，断点调试、逐过程\逐语句调试通常可以混合使用，在该快进的地方选择断点调试、在该仔细查的地方逐过程\逐语句调试。
- ✓ 调试过程中的快捷键需要熟记，有利于加快操作速度。