Supplement II.B: Learning C++ Effectively with Visual C++

For Introduction to C++ Programming
By Y. Daniel Liang

0 Introduction

Supplement II.A, "Visual C++ Tutorial," gives a brief tutorial on how to use Visual C++. VC++ is not only a powerful and productive Java program development tool, but it is also a valuable pedagogical tool for teaching and learning Java programming. This supplement shows how to use VC++ effectively with the text.

The supplement is written for instructors, but it is also useful to students.

1 Important Tips

The objective of the course is to teach C++, not Visual C++. VC++ is a complex and powerful tool. All you need for this course, however, is a small and simple set of features that enable students to create, compile, run, and debug programs. So students should avoid exploring unnecessary features.

If your students follow the instructions in Supplement II.A, "Visual C++ Tutorial," or the instructions from you, students can master all essential skills in sixty minutes. It is important that your students adhere to the instructions to avoid frustrating mistakes. If a mistake is made, simply read the instructions and restart from scratch.

2 VC++ as a Valuable Pedagogical Tool

The following sections demonstrate how to utilize VC++ in the first seven chapters.

2.1 Using VC++ in Chapter 1

After Listing 1.1, you can start to cover how to create, compile, and run a program in VC++.

2.2 Using VC++ in Chapter 2

You may start to introduce debugging when you cover variables. You can use debug to show the value of a variable in the memory and show the change of the value during execution. Figure 1 shows a simple test program with variable *i*.

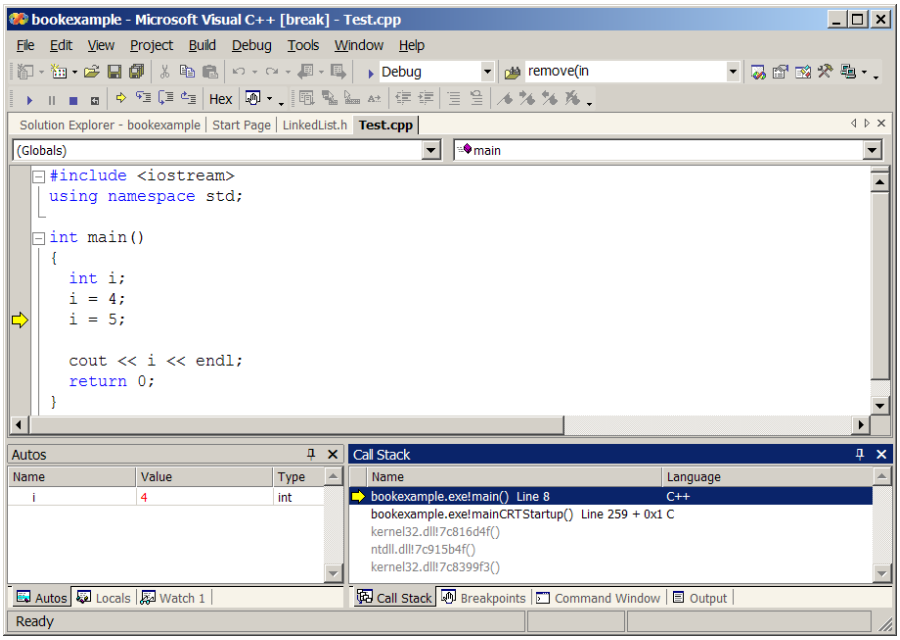

Figure 1

Displaying values of variables in VC++ debugger.

2.3 Using VC++ in Chapter 3

Use the debugger to trace the if statements in Section 3.7, "Nested if Statements," as shown in Figure 2.

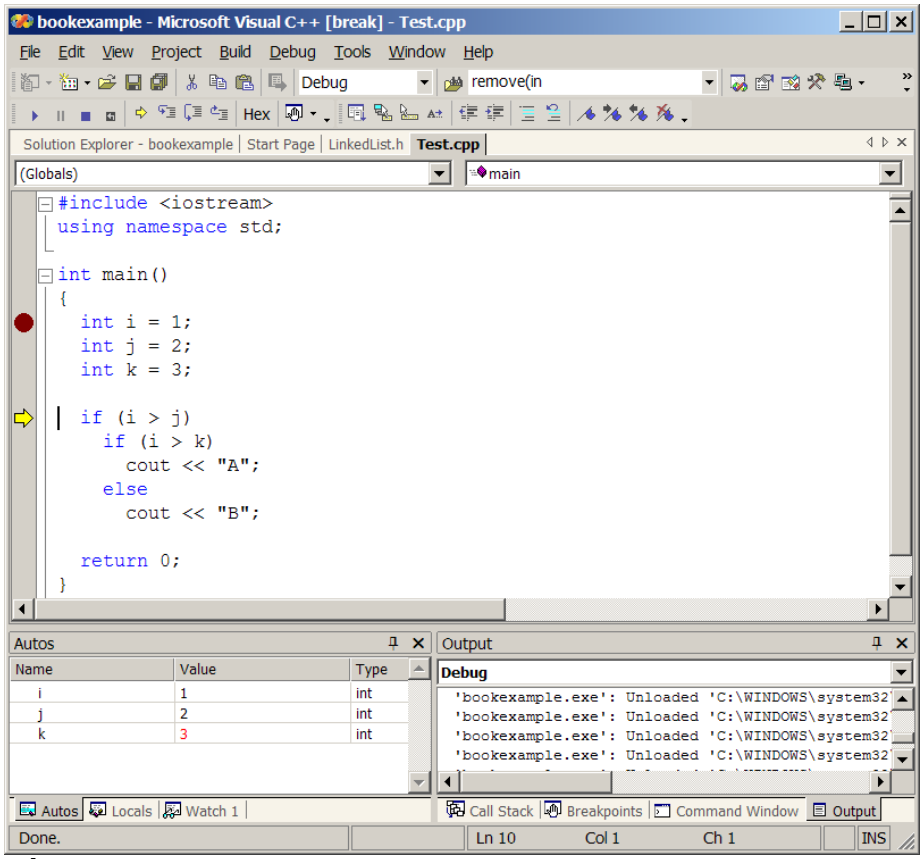

Figure 2

Trace the execution of an if statement.

2.4 Using VC++ in Chapter 4

Use the debugger to trace the while loop in Section 4.2, "The while Loop," as shown in Figure 3.

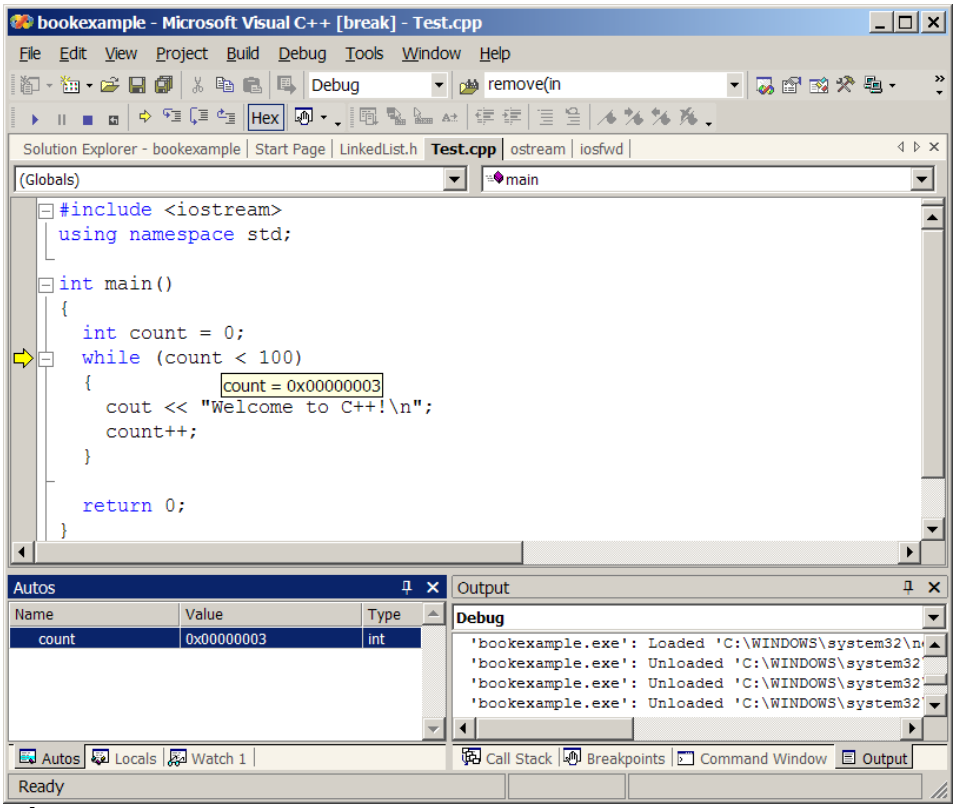

Figure 3

Trace the execution of a loop statement.

2.5 Using VC++ in Chapter 5

You can use the debugger to show the call stack, which is very effective to help understand function invocation. Let us use Listing 5.1 to demonstrate function invocation. Set a breakpoint at line 6. Start debugger, and the debugger pauses at Line 6. Choose Step into to step into the max function, as shown in Figure 4. Now in the Message pane, you will see the arguments are passed to the function.

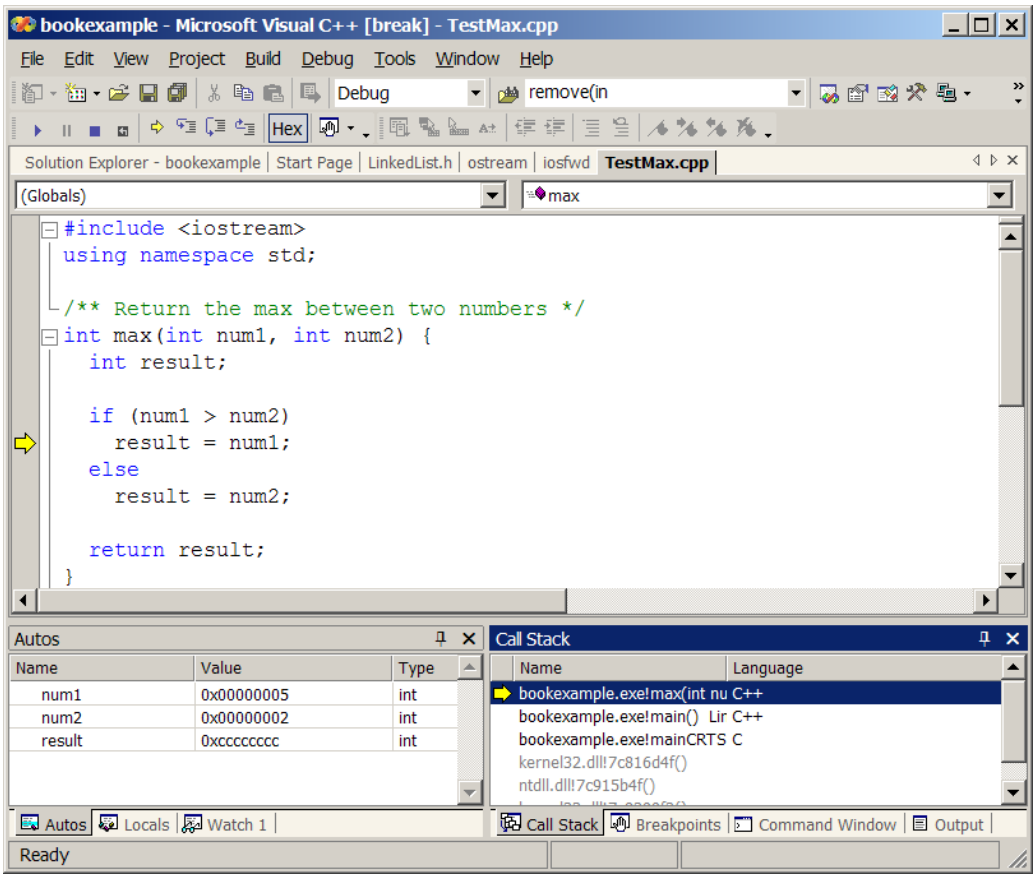

Figure 4

Trace function invocation.

2.6 Using VC++ in Chapter 6

You can use the debugger to show the values of all the elements in an array. Figure 5 shows debugging TestArray.cpp in Listing 6.1.

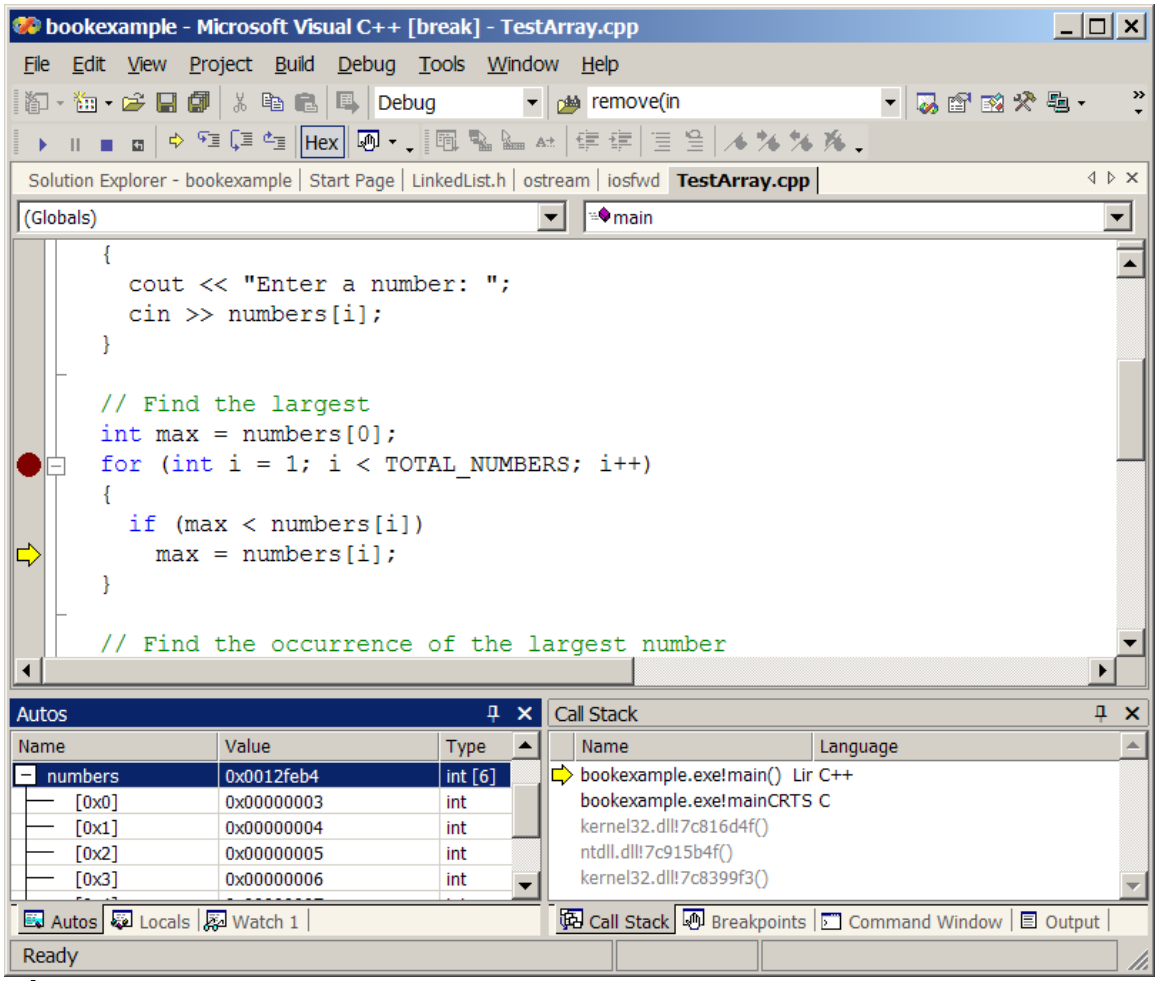

Figure 5

You can see the change of values in an array.

2.7 Using VC++ in Chapter 7

You can use the debugger to show the relationship between a variable and its pointer, as shown in Figure 6.

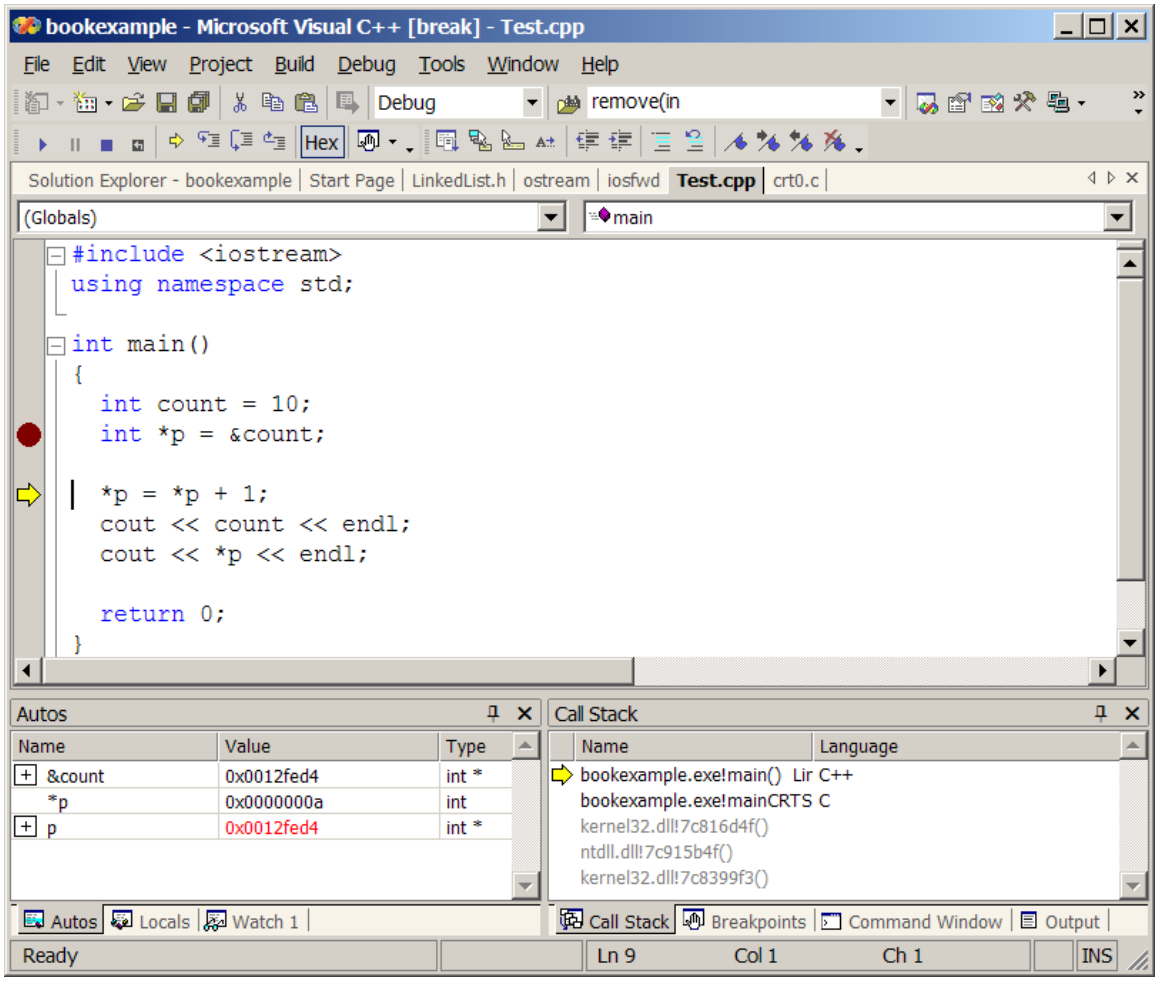

Figure 6

You can see the relationship between variable count and its pointer variable p.

2.7 Using VC++ in Chapter 8

Figure 7 shows tracing recursive execution of the factorial function.

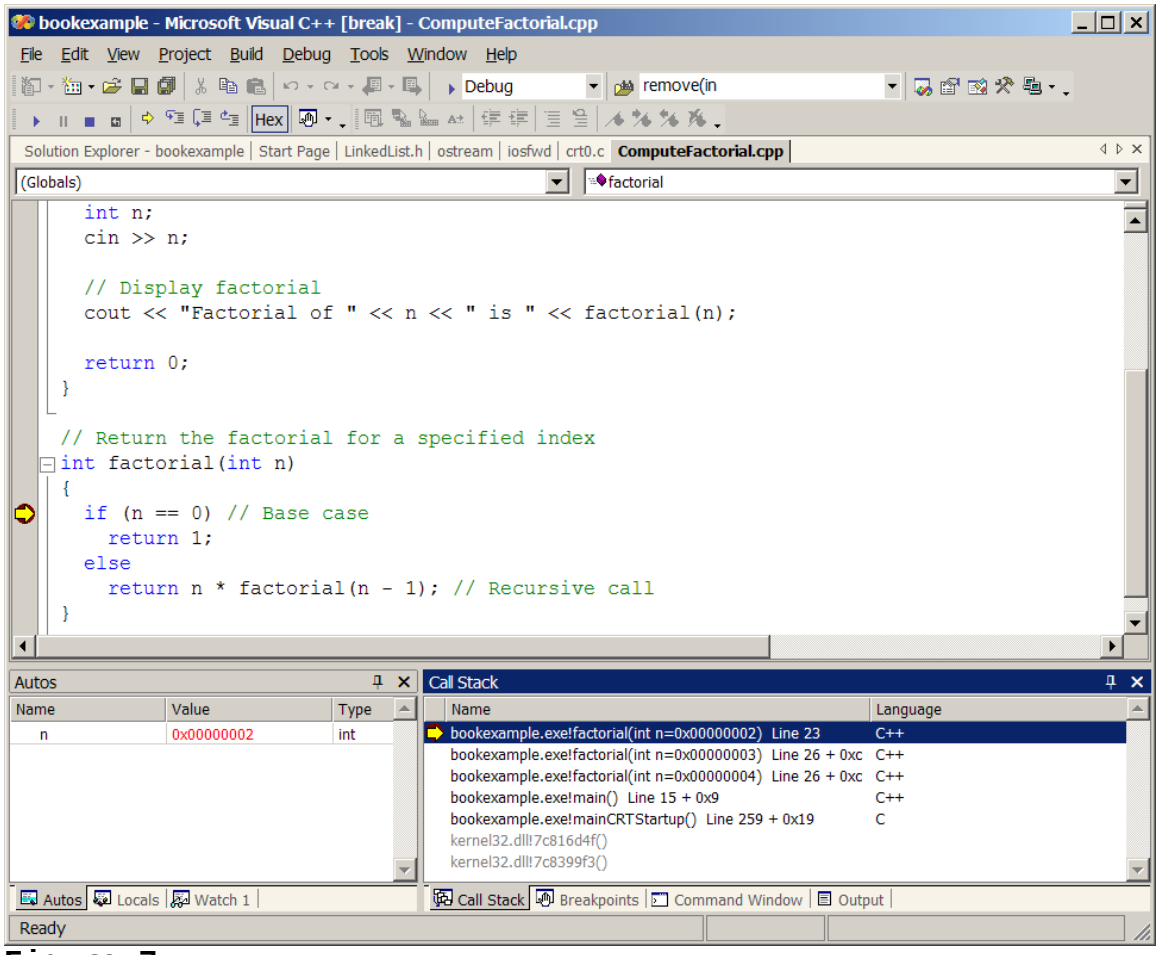

Figure 7

Trace a recursive function invocation.

2.8 Using VC++ in Chapter 9

You can use the debugger to show the contents of an object. Figure 8 shows debugging TestCircle.java in Listing 9.1.

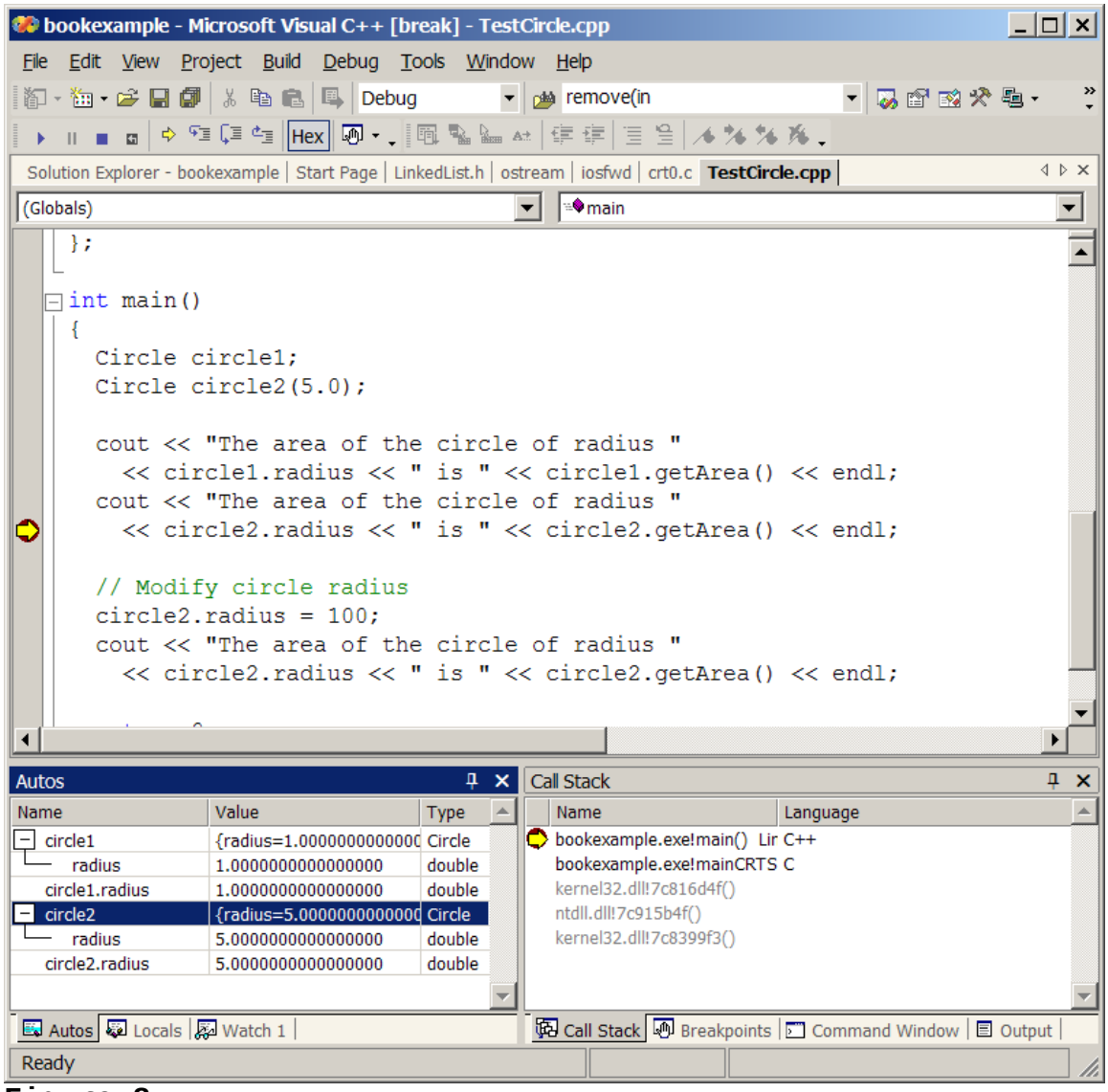

Figure 8

You can see the change of values in an object.

You can use the debugger to demonstrate how arguments are passed and to see the differences between passing primitive type values and objects.