Debugging in Visual Studio

Learn how to use the powerful integrated debugging environment provided in Visual Studio 2003 and 2005

By Steve Jones
Game Institute

Microsoft® Visual Studio 2005

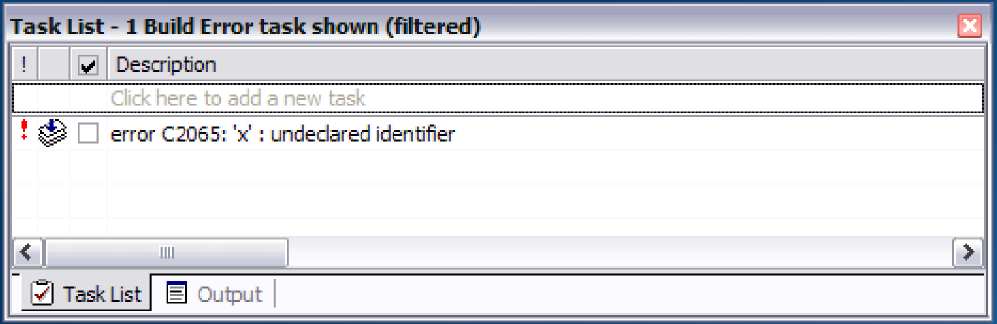
What we will cover

- •Debugging native, 32-bit console and win32 applications
- •Learn most common debugging tools
- •Visual Studio .NET 2003 and 2005 IDE environments
- •The most common techniques for debugging
- •Examples

Debugging

What are you trying to find and fix?

- Two main types of code errors
  - Syntax
    - Compiler catches most if not all of these for you.
  - Semantic or logical
    - Syntactically correct yet program may “crash and burn” at run-time!



Task List - 1 Build Error task shown (filtered)

!		✓	Description
			Click here to add a new task
!		☐	error C2065: 'x' : undeclared identifier

For example: Compiler will not catch an un-initialized pointer but you WILL get a run-time error if you try to use it!

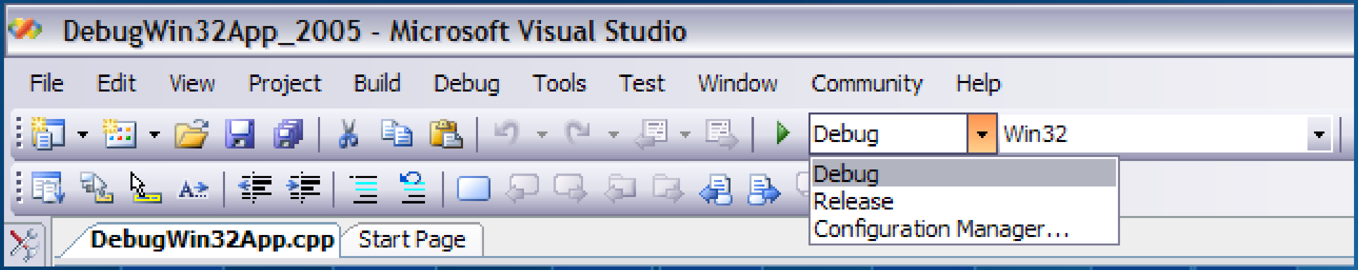
Why Should I Use Visual Studio to Debug my Program?

- Even most experienced coder creates errors or “bugs”
- Visual Studio debugger will provide two powerful run-time facilities:
  - Trace the program **Execution**
  - **Watch** variables during program execution
- These allow you to stop at procedure locations, inspect memory and register values, change variables, observe message traffic, and get a close look at what your code does.

Microsoft® Visual Studio 2005

Project Configuration Settings

- Debug vs. Release Configurations
  - The **Debug** configuration of your program is compiled with full symbolic debug information and no optimization.
  - The **Release** configuration of your program is fully optimized and contains no symbolic debug information.
  - Must be in Debug configuration to debug your program.



Getting Acquainted with Visual Studio Debugger

- Debugger Windows
  - Autos
  - Locals
  - Watch
  - Call Stack
  - Command Window
  - QuickWatch Dialog
  - Breakpoints window
  - Threads
  - Modules
  - Processes
  - Memory
  - Disassembly
  - Registers

- Execution Control
  - Starting or Continuing Execution
  - Stopping
  - Breaking Execution
  - Stepping Into and Out of code
  - Jumping to another location

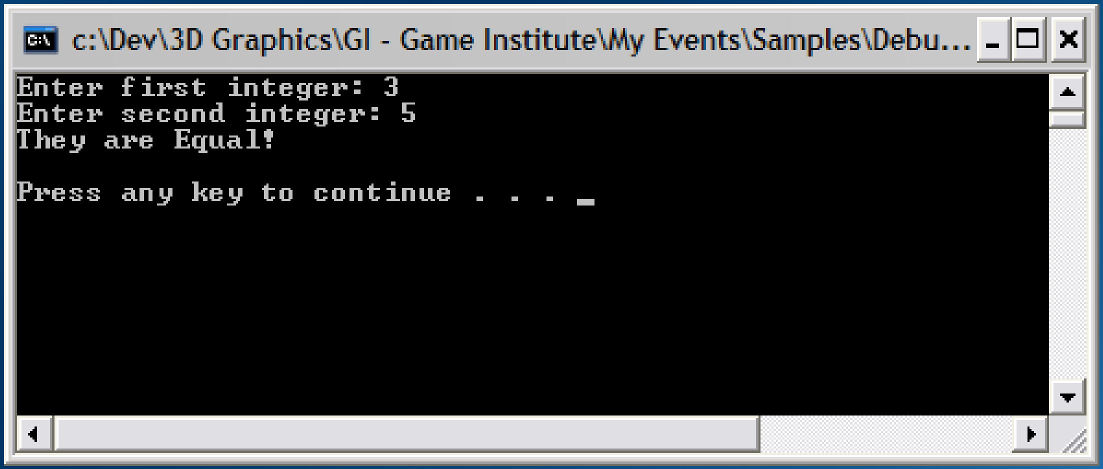
Debugging Example #1
Console app

This simple console program should determine whether two integers are equal.

Code compiled just fine,

0 warnings, 0 errors

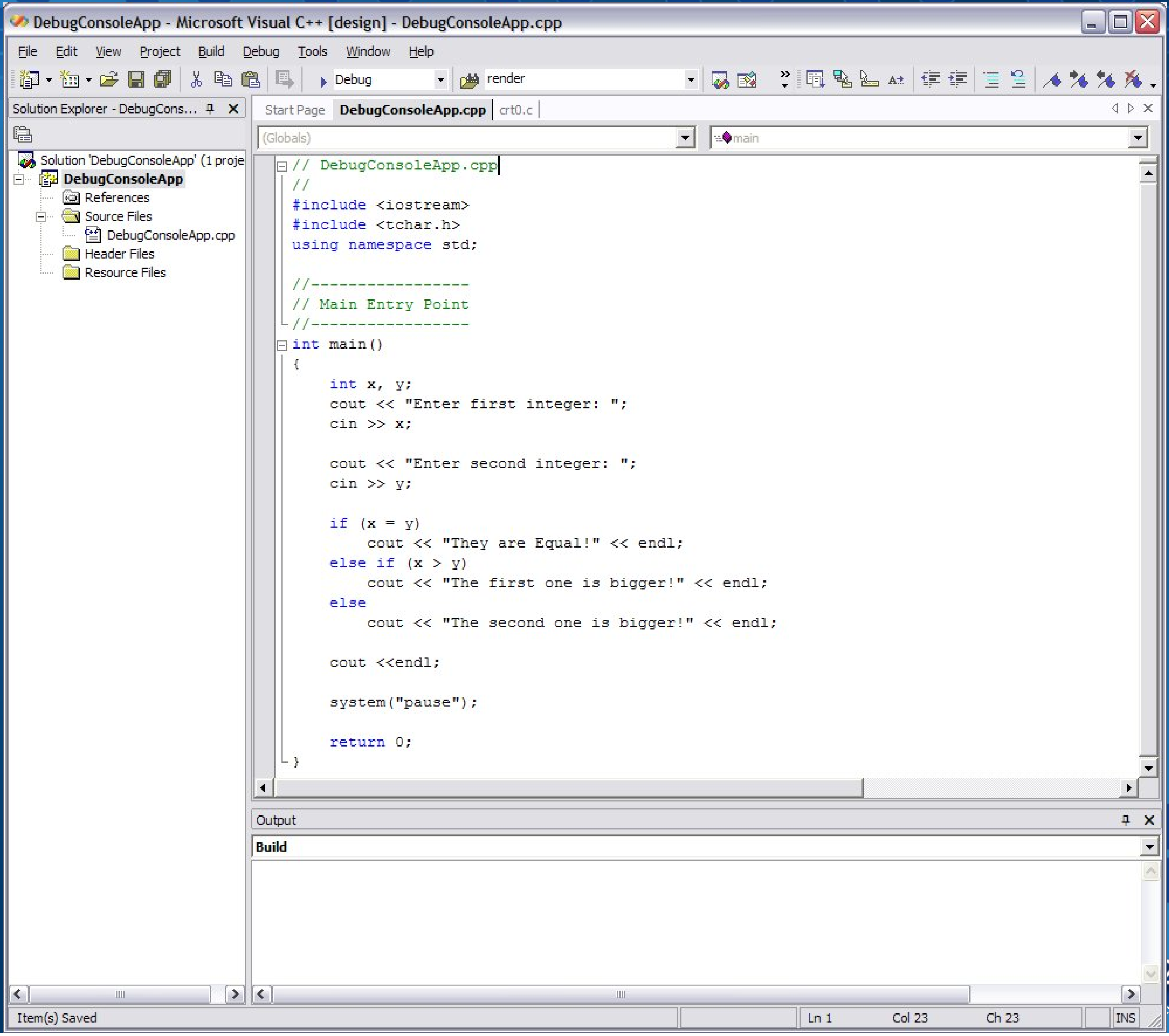
… BUT the code obviously has a logical error! 3 does not equal 5!


c:\Dev\3D Graphics\GI - Game Institute\My Events\Samples\Debu...

```
Enter first integer: 3
Enter second integer: 5
They are Equal!

Press any key to continue . . . _
```

Microsoft® Visual Studio 2005

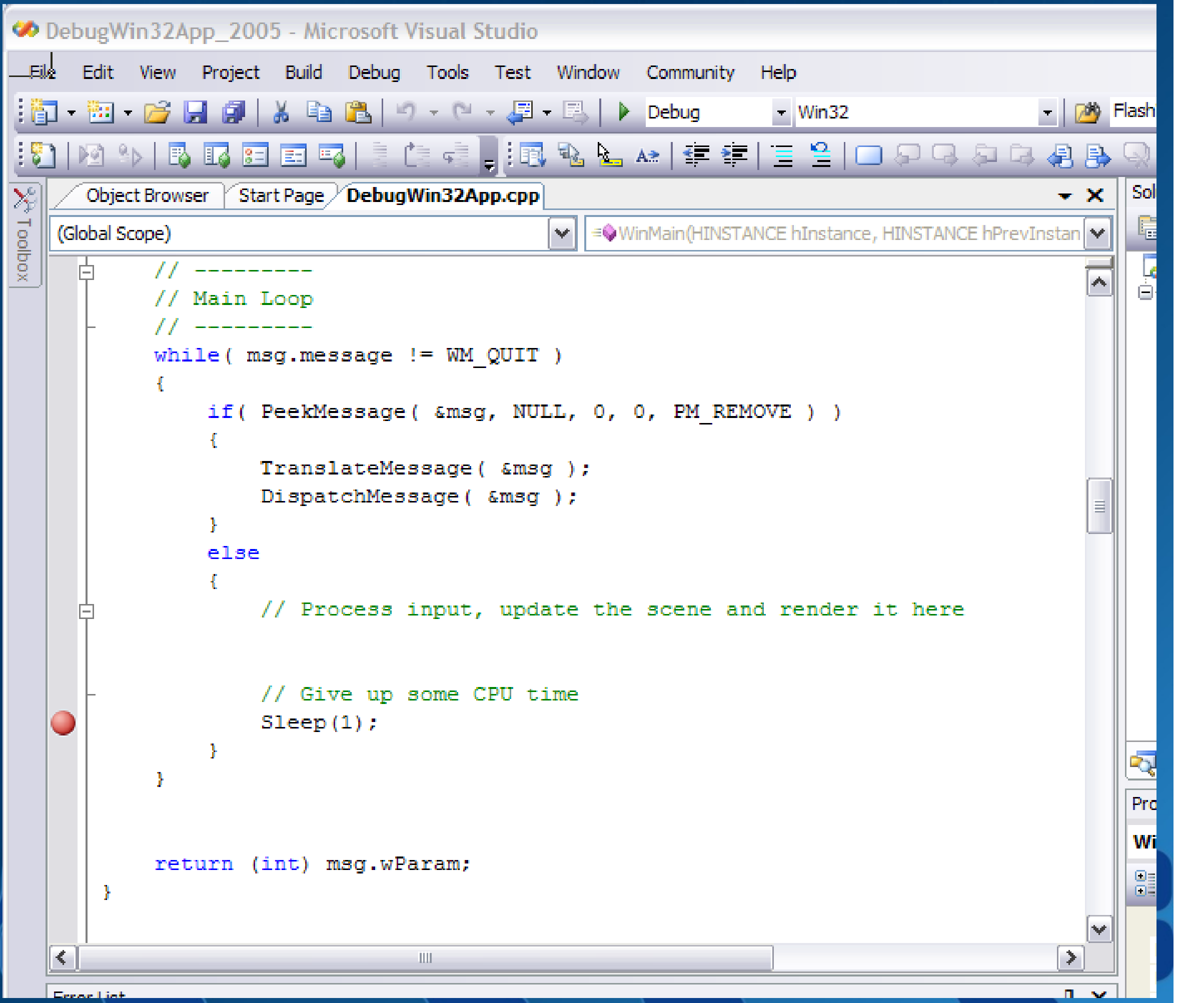
Debugging Example #1 (a console app.)


DebugConsoleApp - Microsoft Visual C++ [design] - DebugConsoleApp.cpp
File Edit View Project Build Debug Tools Window Help
Debug
render
Solution Explorer - DebugCons...
Solution 'DebugConsoleApp' (1 proje
DebugConsoleApp
References
Source Files
DebugConsoleApp.cpp
Header Files
Resource Files
Start Page
DebugConsoleApp.cpp
crt0.c
(Globals)
main
Output
Build
Item(s) Saved
Ln 1
Col 23
Ch 23
INS

```
// DebugConsoleApp.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//-----------------
// Main Entry Point
//-----------------
int main()
{
    int x, y;
    cout << "Enter first integer: ";
    cin >> x;

    cout << "Enter second integer: ";
    cin >> y;

    if (x = y)
        cout << "They are Equal!" << endl;
    else if (x > y)
        cout << "The first one is bigger!" << endl;
    else
        cout << "The second one is bigger!" << endl;

    cout <<endl;

    system("pause");

    return 0;
}
```

What is a Breakpoint?

- Breakpoints are user-defined code locations that pause execution
- You know them by the little, red “dot” in the left margin of the editor window
- F9 to add or remove (toggle)
- Or left-mouse click in margin
- Unlimited number of them to use.



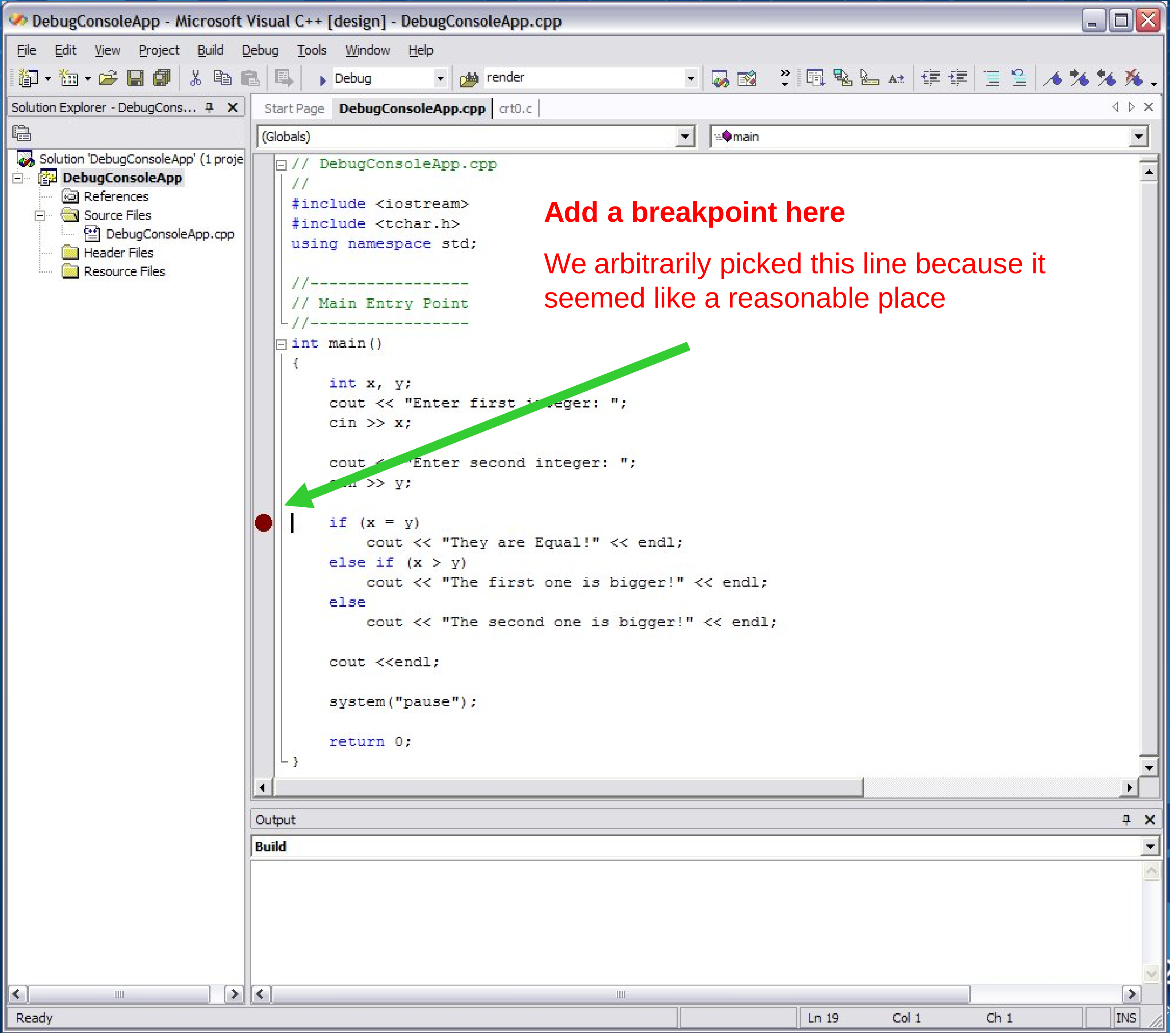
DebugWin32App_2005 - Microsoft Visual Studio
File Edit View Project Build Debug Tools Test Window Community Help
Debug
Win32
Object Browser
Start Page
DebugWin32App.cpp
(Global Scope)
WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstan

```
    // ---------
    // Main Loop
    // ---------
    while( msg.message != WM_QUIT )
    {
        if( PeekMessage( &msg, NULL, 0, 0, PM_REMOVE ) )
        {
            TranslateMessage( &msg );
            DispatchMessage( &msg );
        }
        else
        {
            // Process input, update the scene and render it here


            // Give up some CPU time
            Sleep(1);
        }
    }


    return (int) msg.wParam;
}
```

Microsoft® Visual Studio 2005

Microsoft Visual Studio is a registered trademark of Microsoft Corp.

Debugging Example #1 (continued)



DebugConsoleApp - Microsoft Visual C++ [design] - DebugConsoleApp.cpp
File Edit View Project Build Debug Tools Window Help
Debug
render
Solution Explorer - DebugCons...
Solution 'DebugConsoleApp' (1 proje
DebugConsoleApp
References
Source Files
DebugConsoleApp.cpp
Header Files
Resource Files
Start Page
DebugConsoleApp.cpp
crt0.c
(Globals)
main
Output
Build
Ready
Ln 19
Col 1
Ch 1
INS

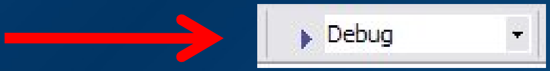
```
// DebugConsoleApp.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//-----------------
// Main Entry Point
//-----------------
int main()
{
    int x, y;
    cout << "Enter first integer: ";
    cin >> x;

    cout << "Enter second integer: ";
    cin >> y;

    if (x = y)
        cout << "They are Equal!" << endl;
    else if (x > y)
        cout << "The first one is bigger!" << endl;
    else
        cout << "The second one is bigger!" << endl;

    cout <<endl;

    system("pause");

    return 0;
}
```

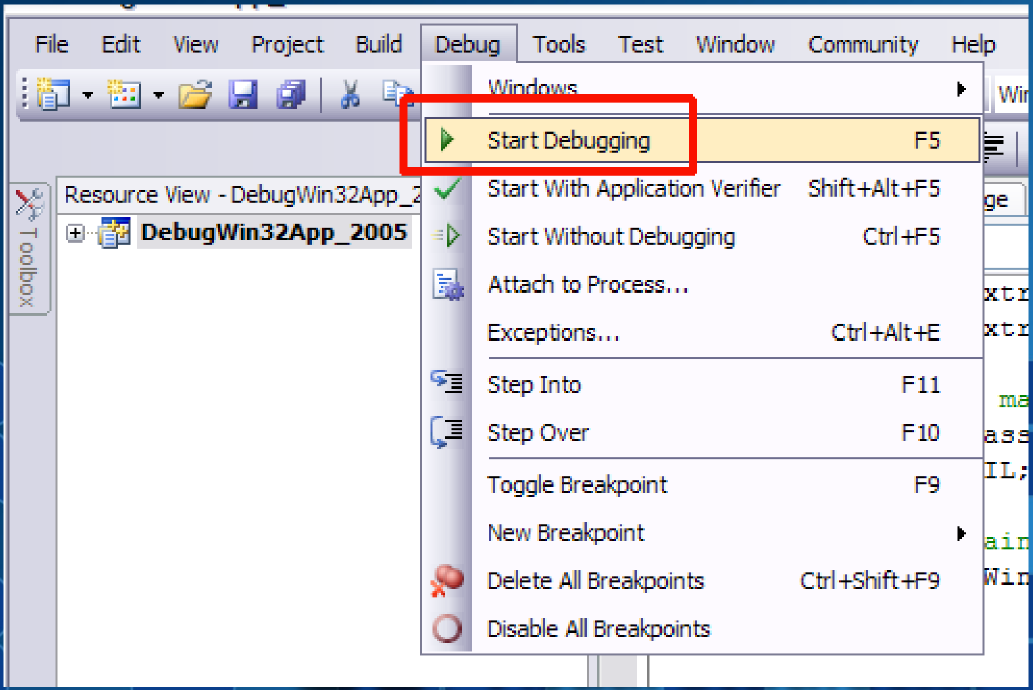
Add a breakpoint here

We arbitrarily picked this line because it seemed like a reasonable place

Starting the Debugging Session

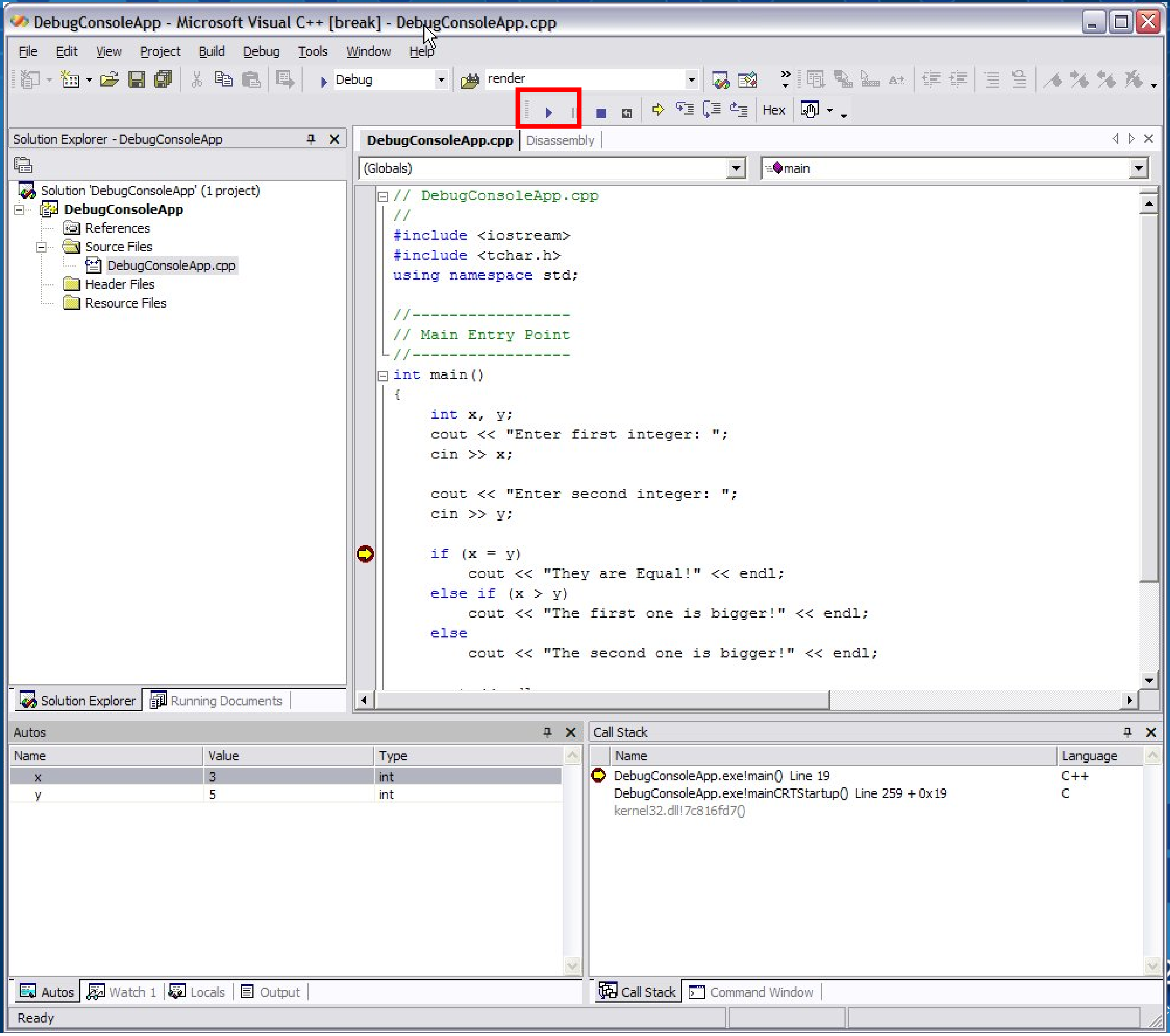
- Make sure you are in a Debug configuration
- Press F5
- Or click on Debug icon
- Or select menu Debug – Start Debugging





Microsoft Visual Studio is a registered trademark of Microsoft Corp.

Debugging Example #1 - Running in the debugger



```
// DebugConsoleApp.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//-----------------
// Main Entry Point
//-----------------
int main()
{
    int x, y;
    cout << "Enter first integer: ";
    cin >> x;

    cout << "Enter second integer: ";
    cin >> y;

    if (x = y)
        cout << "They are Equal!" << endl;
    else if (x > y)
        cout << "The first one is bigger!" << endl;
    else
        cout << "The second one is bigger!" << endl;
```

Autos

Name	Value	Type
x	3	int
y	5	int

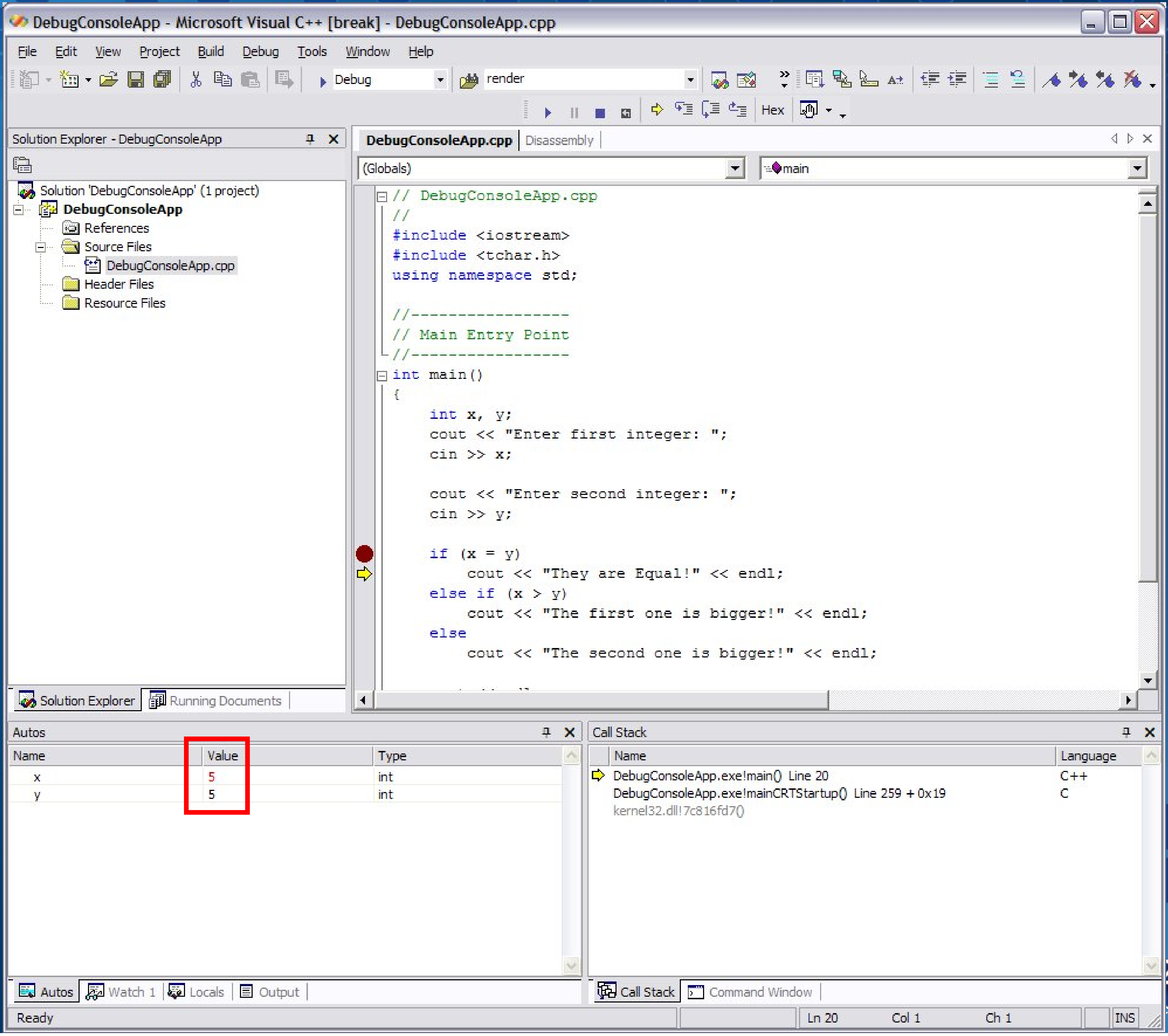
Call Stack

Name	Language
DebugConsoleApp.exe!main() Line 19	C++
DebugConsoleApp.exe!mainCRTStartup() Line 259 + 0x19	C
kernel32.dll!7c816fd7()	

2005

rosoft Corp.

Debugging Example #1 Stepping, examine variables



```
// DebugConsoleApp.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//-----------------
// Main Entry Point
//-----------------
int main()
{
    int x, y;
    cout << "Enter first integer: ";
    cin >> x;

    cout << "Enter second integer: ";
    cin >> y;

    if (x = y)
        cout << "They are Equal!" << endl;
    else if (x > y)
        cout << "The first one is bigger!" << endl;
    else
        cout << "The second one is bigger!" << endl;
```

Autos

Name	Value	Type
x	5	int
y	5	int

Call Stack

Name	Language
DebugConsoleApp.exe!main() Line 20	C++
DebugConsoleApp.exe!mainCRTStartup() Line 259 + 0x19	C
kernel32.dll!7c816fd7()	

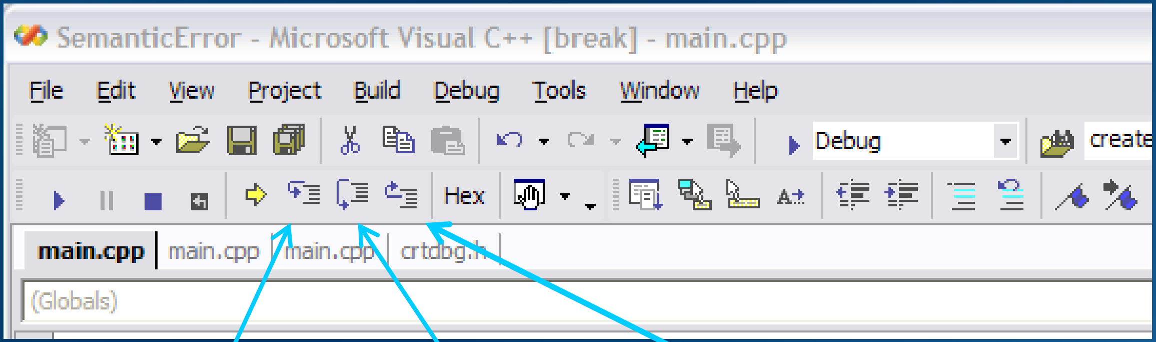
2005

rosoft Corp.

Execution Control

Stepping through your code

- Starting / Stopping
- Breaking
- Stepping through your application
  - (F10, F11 or Toolbar buttons)
- Run to a specific location
  - **Run To Cursor** (right-click menu)


SemanticError - Microsoft Visual C++ [break] - main.cpp
File Edit View Project Build Debug Tools Window Help
Debug
create
Hex
main.cpp
main.cpp
main.cpp
crtdbg.h
(Globals)

F11
Step Into

F10
Step Over

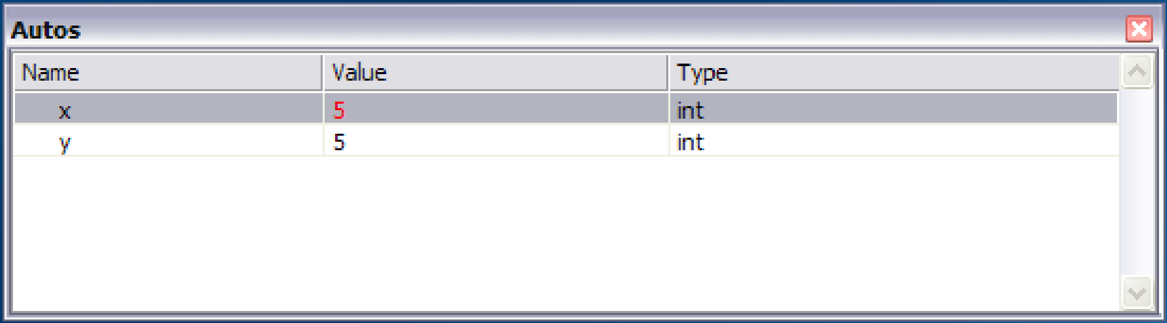
Shift + F11
Step Out

Equivalent hot-keys

Microsoft® Visual Studio 2005

Autos Window

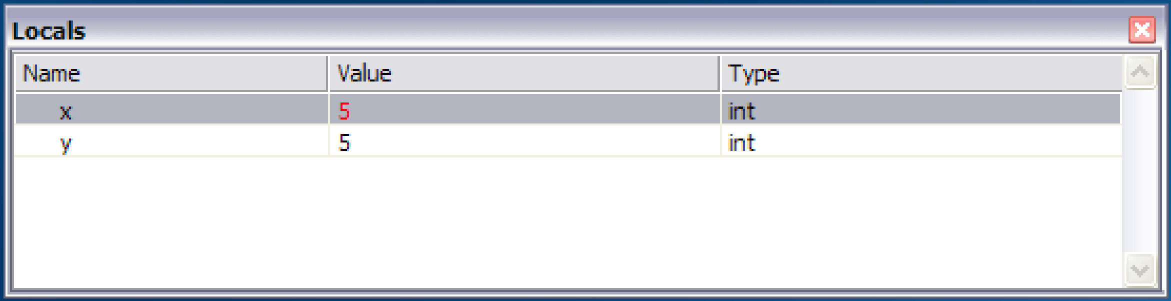
- **Name**
  - The names of all variables in the current statement and the previous statement. The current statement is the statement at the current execution location, which is the statement that will be executed next if execution continues.
- **Value**
  - The value contained by each variable. By default, integer variables are represented in decimal form.
- **Type**
  - The data type of each variable listed in the **Name** column.



Name	Value	Type
x	5	int
y	5	int

Locals Window

- **Name**
  - This column contains the names of all local variables in the current scope.
- **Value**
  - The value contained by each variable. By default, integer variables are represented in decimal form.
- **Type**
  - The data type of each variable listed in the **Name** column.

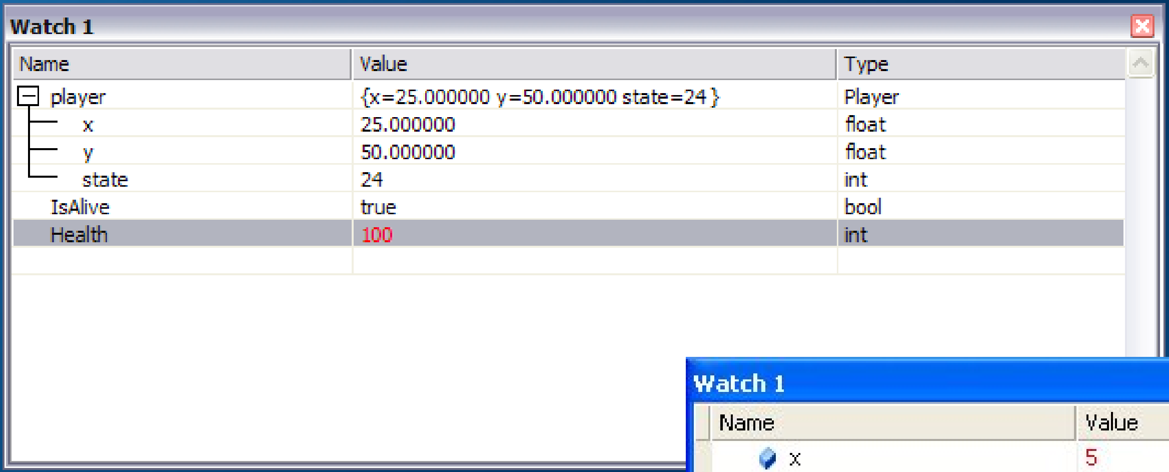


Locals

Name	Value	Type
x	5	int
y	5	int

Watch window(s)

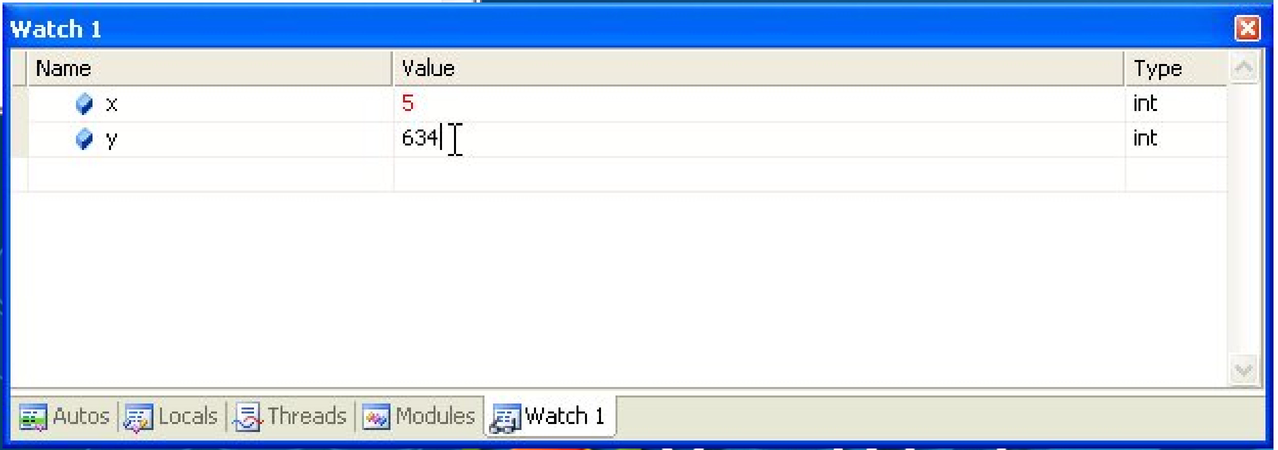
- Watch window displays Name, Value, and Type of variables
- Type in or click-drag variables into window
- Change values live while at break
- You have 4 independent Watch windows



Watch 1

Name	Value	Type
player	{x=25.000000 y=50.000000 state=24 }	Player
x	25.000000	float
y	50.000000	float
state	24	int
IsAlive	true	bool
Health	100	int

(VS 2005 & VC++ Express)



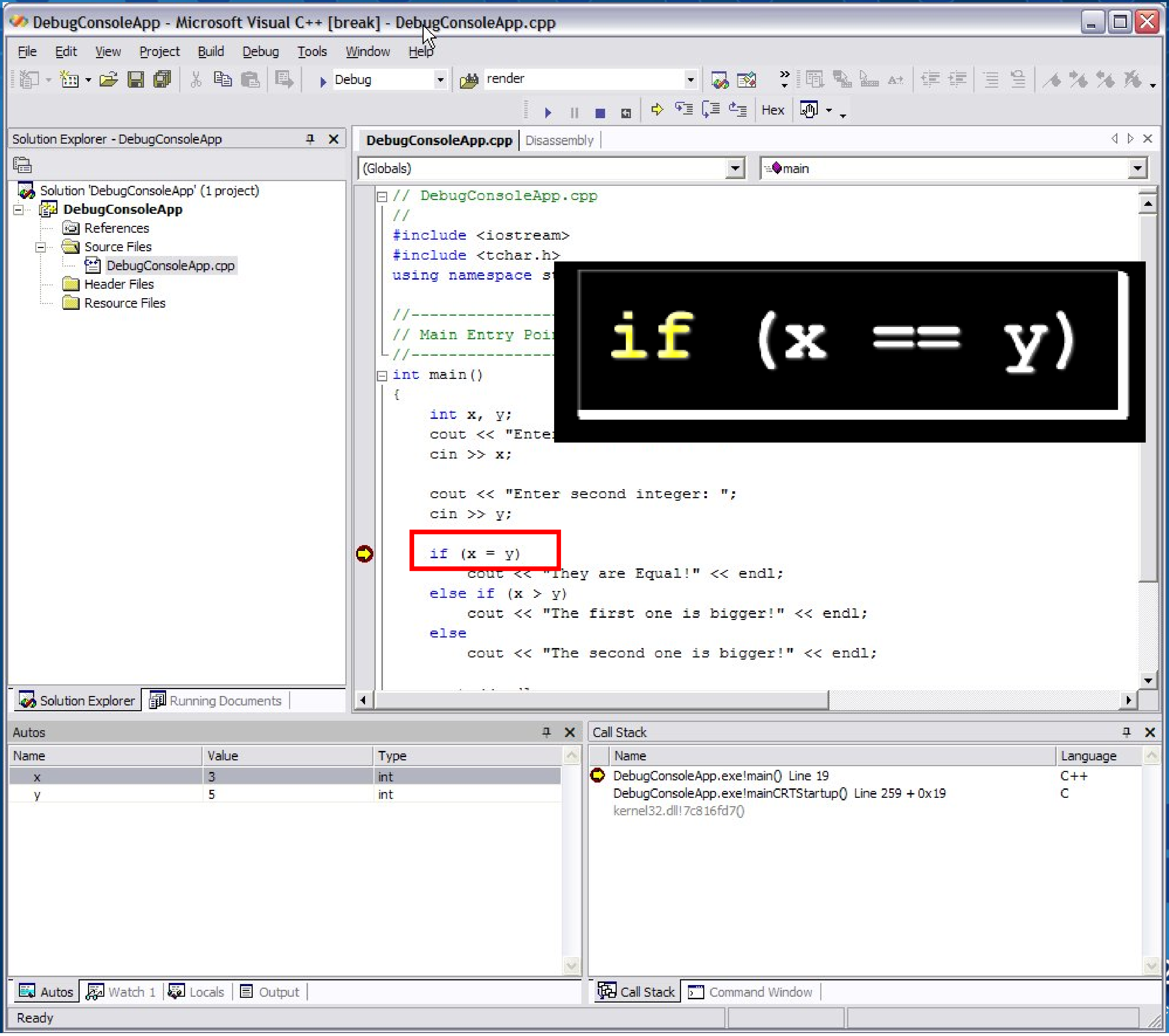
Watch 1

Name	Value	Type
x	5	int
y	634	int

Visual Studio 2005

Microsoft Visual Studio is a registered trademark of Microsoft Corp.

Debugging Example #1 - Found error



DebugConsoleApp - Microsoft Visual C++ [break] - DebugConsoleApp.cpp

```
// DebugConsoleApp.cpp
//
#include <iostream>
#include <tchar.h>
using namespace s

//----------------
// Main Entry Poi
//----------------
int main()
{
    int x, y;
    cout << "Ente
    cin >> x;

    cout << "Enter second integer: ";
    cin >> y;

    if (x = y)
        cout << "They are Equal!" << endl;
    else if (x > y)
        cout << "The first one is bigger!" << endl;
    else
        cout << "The second one is bigger!" << endl;
```

```
if (x == y)
```

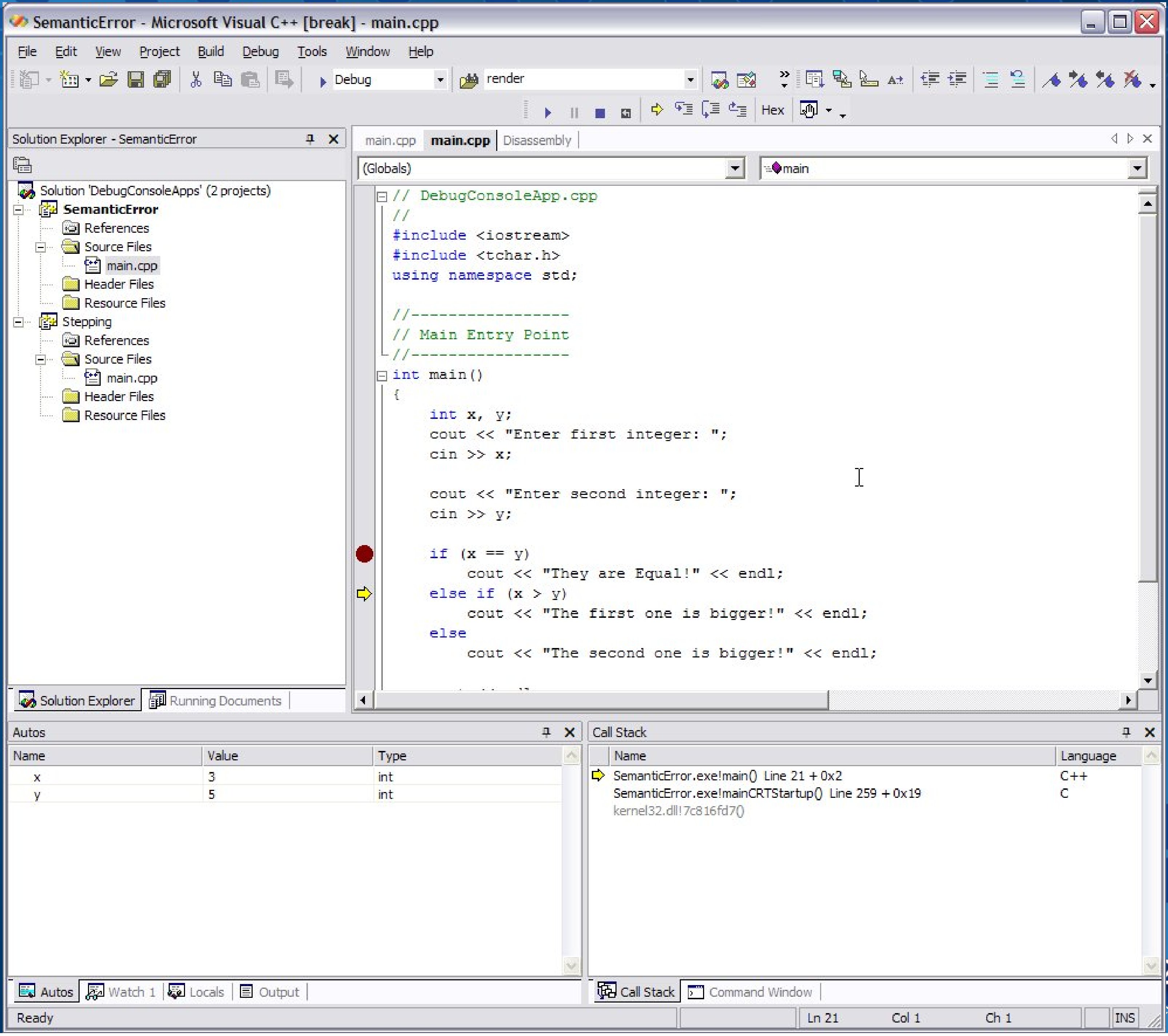
Autos

Name	Value	Type
x	3	int
y	5	int

Call Stack

Name	Language
DebugConsoleApp.exe!main() Line 19	C++
DebugConsoleApp.exe!mainCRTStartup() Line 259 + 0x19	C
kernel32.dll!7c816fd7()	

Debugging Example #1 - Fixed error, recompiled, run, step



SemanticError - Microsoft Visual C++ [break] - main.cpp

Solution Explorer - SemanticError

```
Solution 'DebugConsoleApps' (2 projects)
  SemanticError
    References
    Source Files
      main.cpp
    Header Files
    Resource Files
  Stepping
    References
    Source Files
      main.cpp
    Header Files
    Resource Files
```

main.cpp | main.cpp | Disassembly

(Globals) | main

```
// DebugConsoleApp.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//-----------------
// Main Entry Point
//-----------------
int main()
{
    int x, y;
    cout << "Enter first integer: ";
    cin >> x;

    cout << "Enter second integer: ";
    cin >> y;

    if (x == y)
        cout << "They are Equal!" << endl;
    else if (x > y)
        cout << "The first one is bigger!" << endl;
    else
        cout << "The second one is bigger!" << endl;
```

Autos

Name	Value	Type
x	3	int
y	5	int

Call Stack

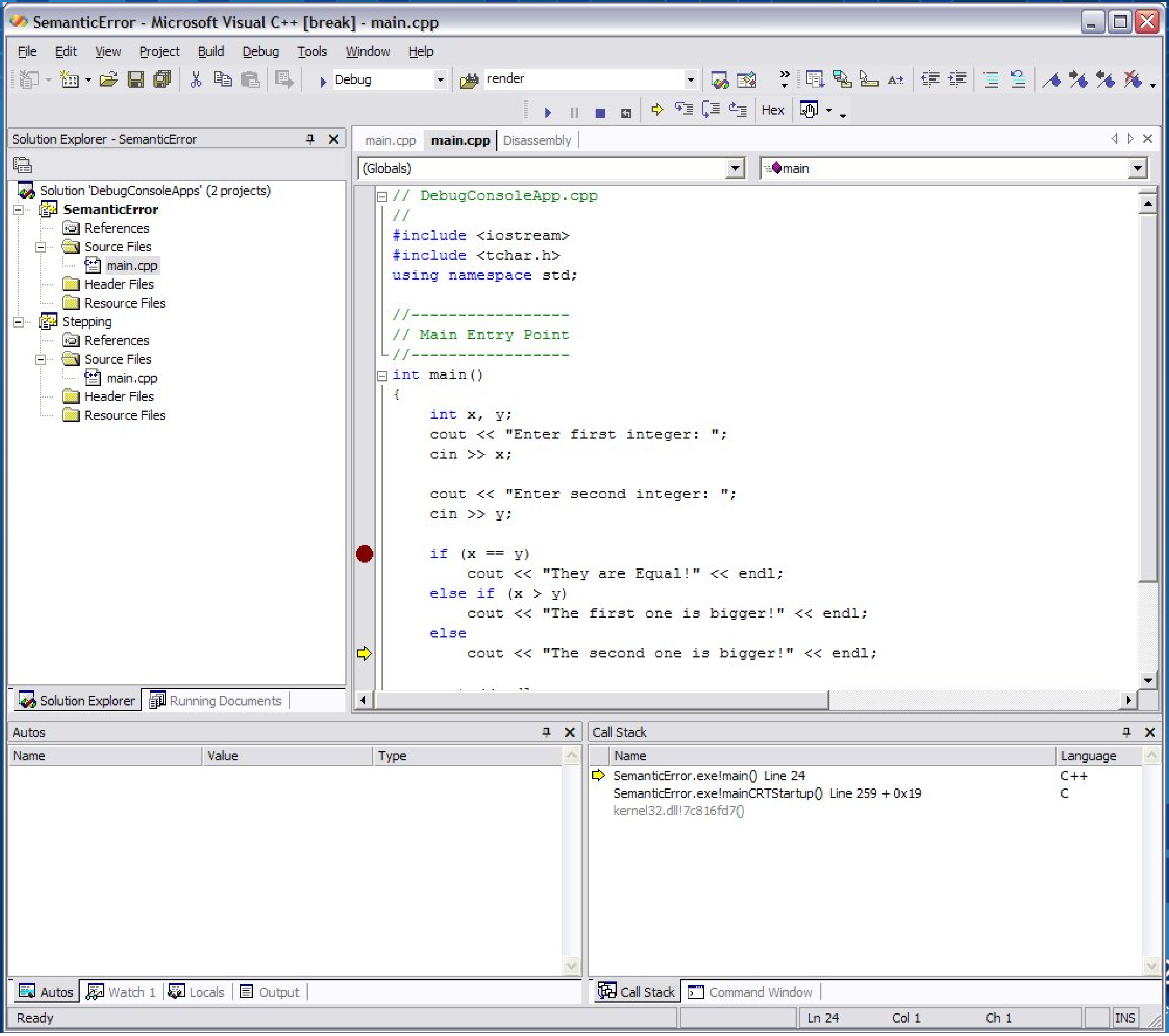
Name	Language
SemanticError.exe!main() Line 21 + 0x2	C++
SemanticError.exe!mainCRTStartup() Line 259 + 0x19	C
kernel32.dll!7c816fd7()	

Ready   Ln 21   Col 1   Ch 1   INS

2005

rosoft Corp.

Debugging Example #1 - Step. Hey the code worked!



```
// DebugConsoleApp.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//-----------------
// Main Entry Point
//-----------------
int main()
{
    int x, y;
    cout << "Enter first integer: ";
    cin >> x;

    cout << "Enter second integer: ";
    cin >> y;

    if (x == y)
        cout << "They are Equal!" << endl;
    else if (x > y)
        cout << "The first one is bigger!" << endl;
    else
        cout << "The second one is bigger!" << endl;
```

Call Stack

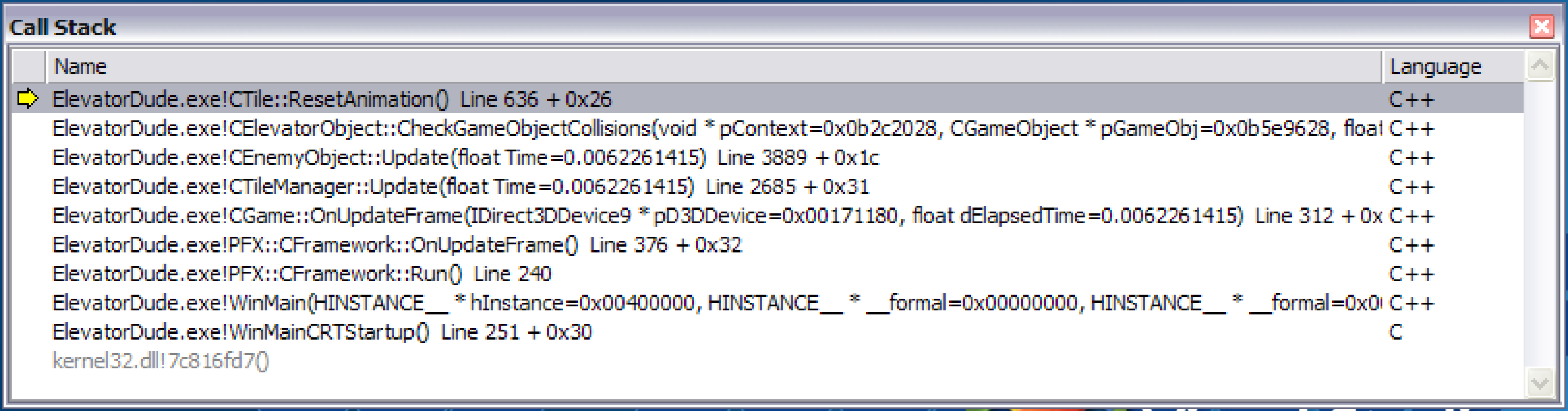
Name	Language
SemanticError.exe!main() Line 24	C++
SemanticError.exe!mainCRTStartup() Line 259 + 0x19	C
kernel32.dll!7c816fd7()	

2005

rosoft Corp.

The Call Stack

- Call stack window displays each function name in the order they were called.
- Yellow arrow identifies the stack frame where the execution pointer is located
- Double-clicking on a function name takes you to the function in source code
- Click Debug – Windows – Call Stack to show window (if hidden). It is shown by default.



Call Stack

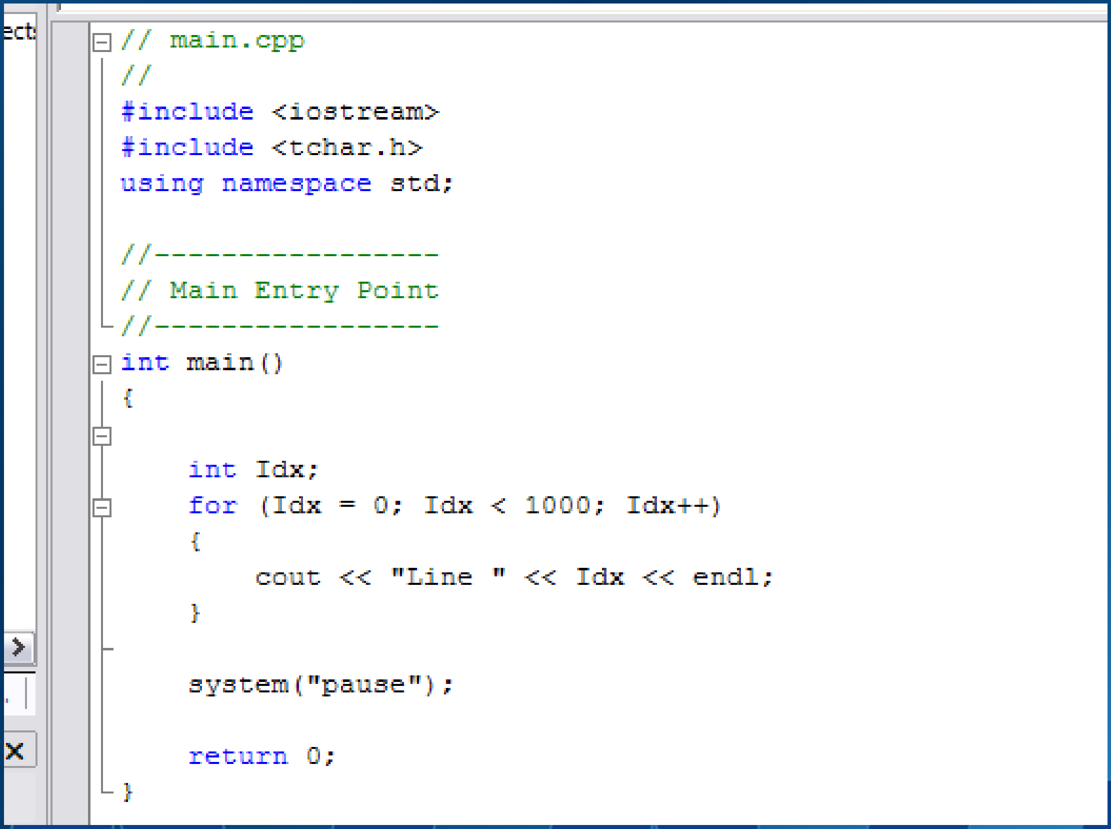
	Name	Language
⇨	ElevatorDude.exe!CTile::ResetAnimation() Line 636 + 0x26	C++
	ElevatorDude.exe!CElevatorObject::CheckGameObjectCollisions(void * pContext=0x0b2c2028, CGameObject * pGameObj=0x0b5e9628, floa	C++
	ElevatorDude.exe!CEnemyObject::Update(float Time=0.0062261415) Line 3889 + 0x1c	C++
	ElevatorDude.exe!CTileManager::Update(float Time=0.0062261415) Line 2685 + 0x31	C++
	ElevatorDude.exe!CGame::OnUpdateFrame(IDirect3DDevice9 * pD3DDevice=0x00171180, float dElapsedTime=0.0062261415) Line 312 + 0x	C++
	ElevatorDude.exe!PFX::CFramework::OnUpdateFrame() Line 376 + 0x32	C++
	ElevatorDude.exe!PFX::CFramework::Run() Line 240	C++
	ElevatorDude.exe!WinMain(HINSTANCE__ * hInstance=0x00400000, HINSTANCE__ * __formal=0x00000000, HINSTANCE__ * __formal=0x0	C++
	ElevatorDude.exe!WinMainCRTStartup() Line 251 + 0x30	C
	kernel32.dll!7c816fd7()	

Example #2

How to Use Conditional Breakpoints

These are breakpoints that only “break” based on a specific condition.

In this example, we will put a conditional breakpoint in the “for” loop and the breakpoint will only stop when our condition is met.



```
// main.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//------------------
// Main Entry Point
//------------------
int main()
{

    int Idx;
    for (Idx = 0; Idx < 1000; Idx++)
    {
        cout << "Line " << Idx << endl;
    }

    system("pause");

    return 0;
}
```

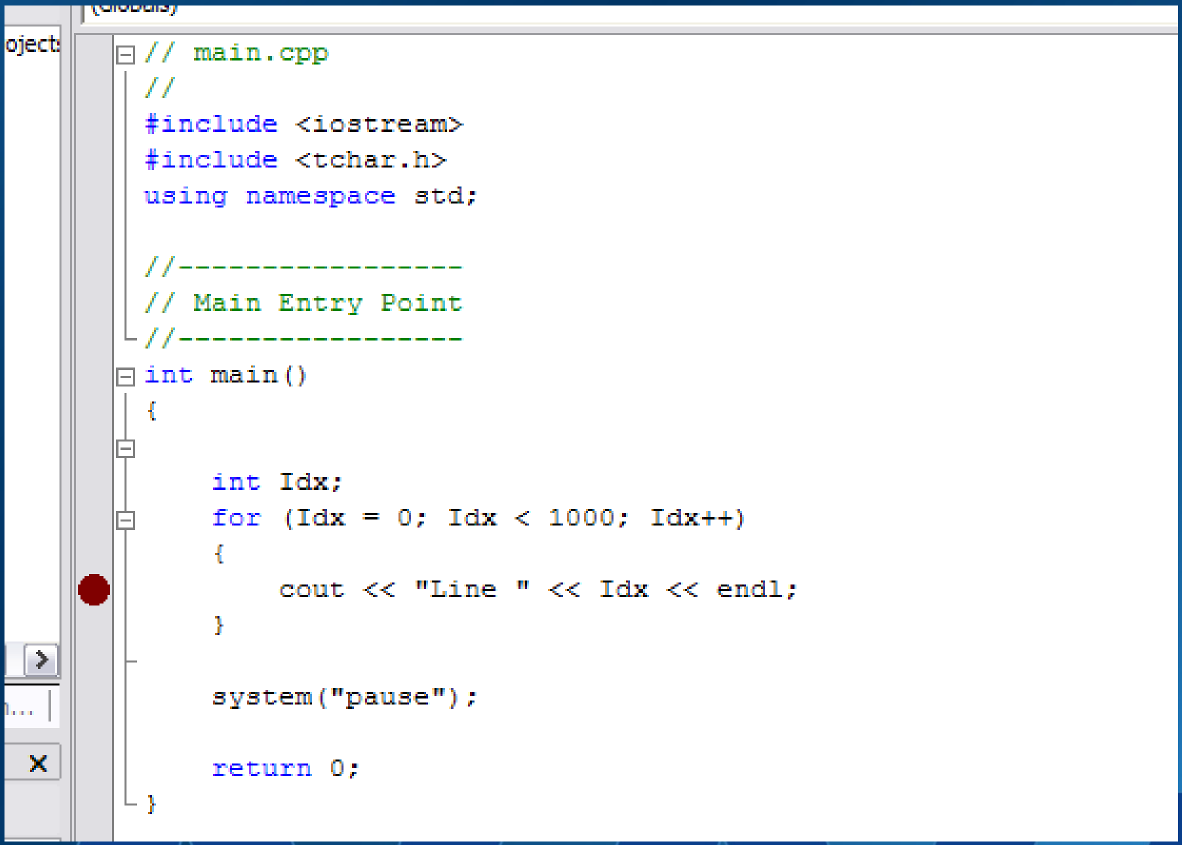
Example #2

Add a breakpoint with a condition

Let's say you want to break execution only when a condition is met rather than break each time the loop cycles.

1. Add a breakpoint on the line you're interested in.

Then we'll configure a condition to it.

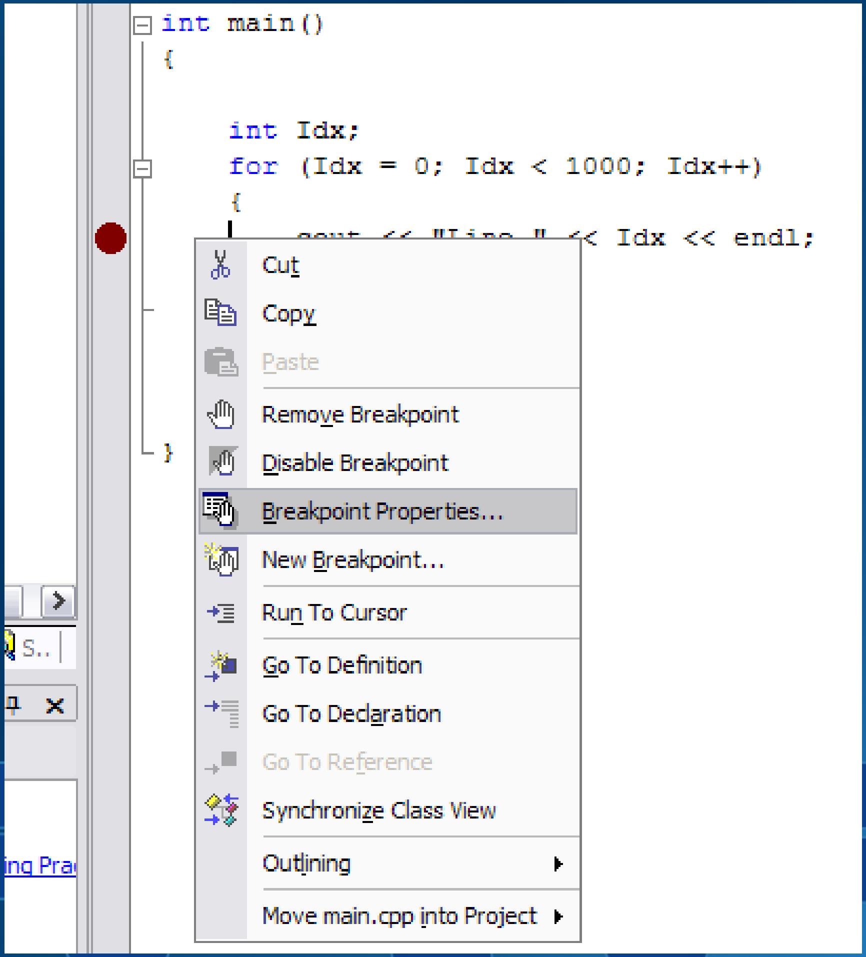


```
// main.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//------------------
// Main Entry Point
//------------------
int main()
{

    int Idx;
    for (Idx = 0; Idx < 1000; Idx++)
    {
        cout << "Line " << Idx << endl;
    }

    system("pause");

    return 0;
}
```

Example #2

Breakpoint Properties (VS 2003)

2. Right-mouse click on the breakpoint
3. Select **Breakpoint Properties**…

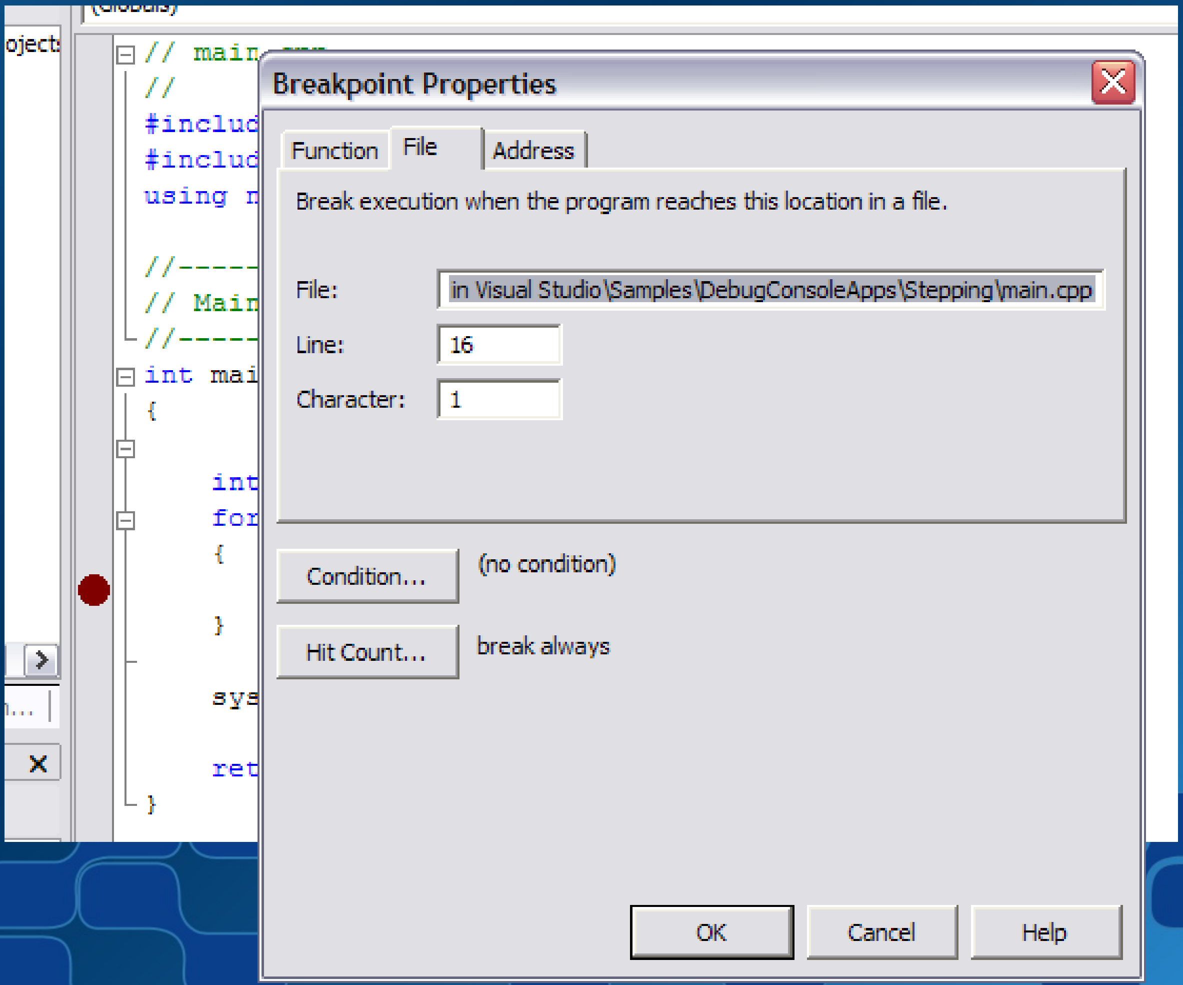


```
int main()
{

    int Idx;
    for (Idx = 0; Idx < 1000; Idx++)
    {
        cout << "Line " << Idx << endl;

}
```

Example #2

Open Breakpoint Properties Dialog (VS 2003)

4. The breakpoint dialog will open.



```
// main.cpp
//
#includ
#includ
using n

//-----
// Main
//-----
int mai
{

    int
    for
    {

    }

    sys

    ret
}
```

Breakpoint Properties

Break execution when the program reaches this location in a file.

File: in Visual Studio\Samples\DebugConsoleApps\Stepping\main.cpp

Line: 16

Character: 1

Condition... (no condition)

Hit Count... break always

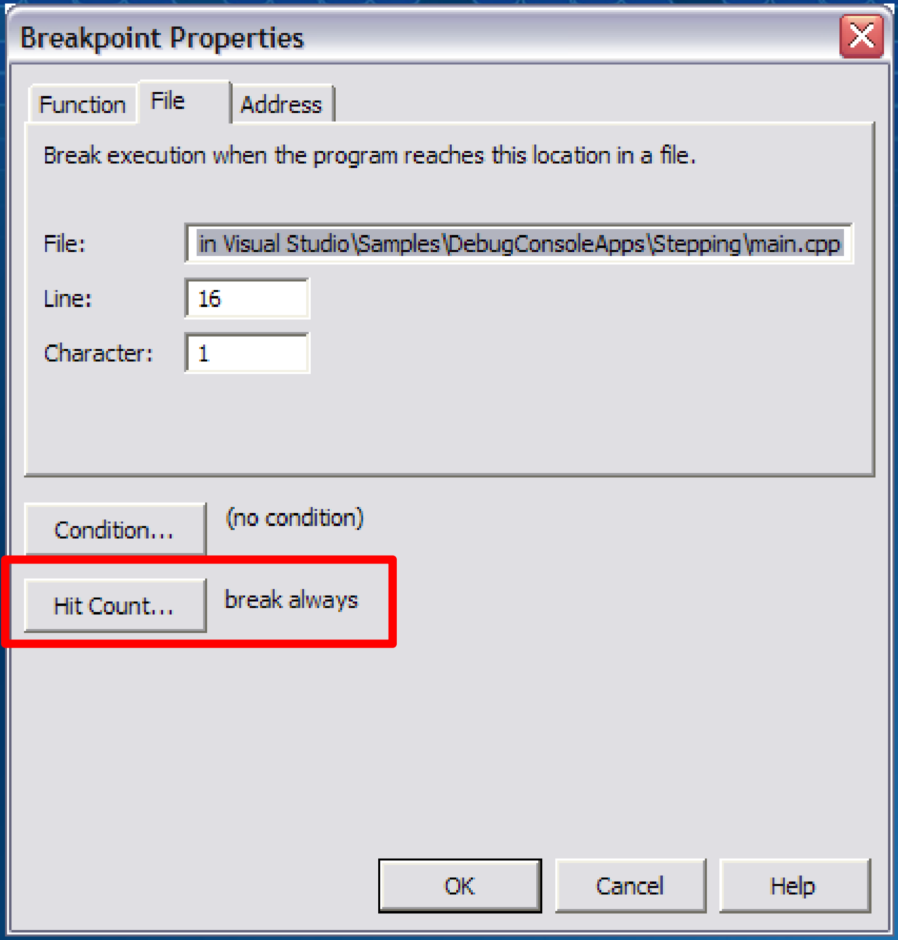
Example #2

Set hit count condition (VS 2003)

5. Click on **Hit Count…** button

   Select frequency of the break

   Default is "break always"



Breakpoint Properties

Break execution when the program reaches this location in a file.

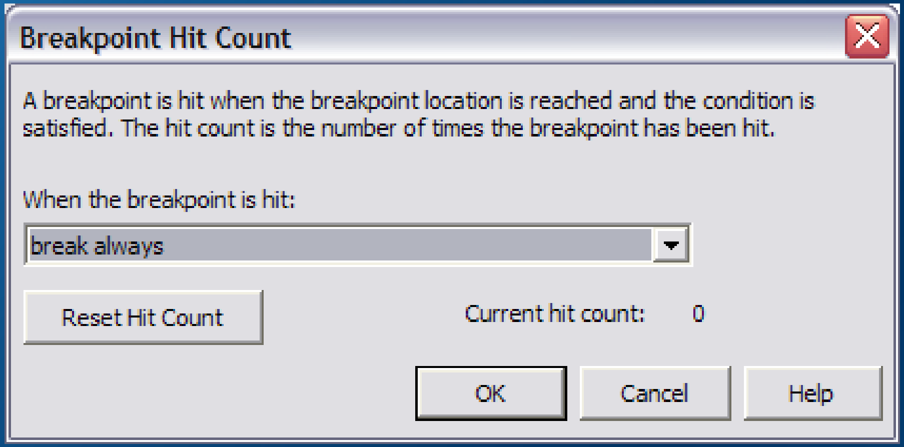
File: in Visual Studio\Samples\DebugConsoleApps\Stepping\main.cpp

Line: 16

Character: 1

Condition… (no condition)

Hit Count… break always



Breakpoint Hit Count

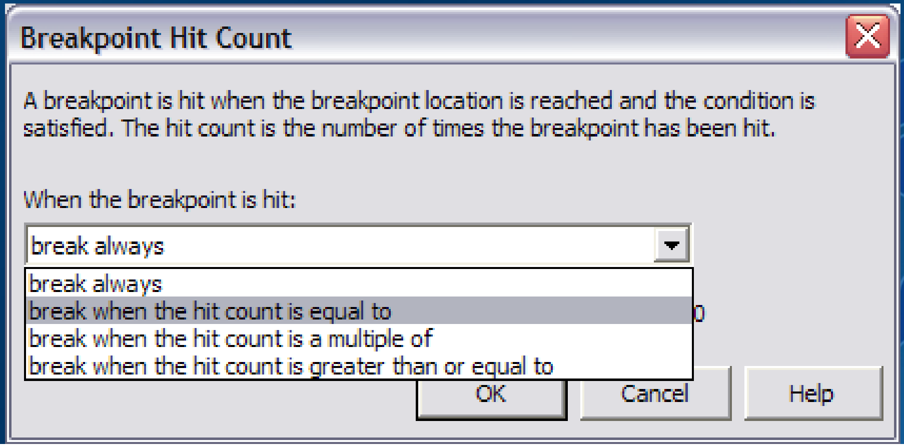
A breakpoint is hit when the breakpoint location is reached and the condition is satisfied. The hit count is the number of times the breakpoint has been hit.

When the breakpoint is hit:

break always

Reset Hit Count

Current hit count: 0



Breakpoint Hit Count

A breakpoint is hit when the breakpoint location is reached and the condition is satisfied. The hit count is the number of times the breakpoint has been hit.

When the breakpoint is hit:

break always

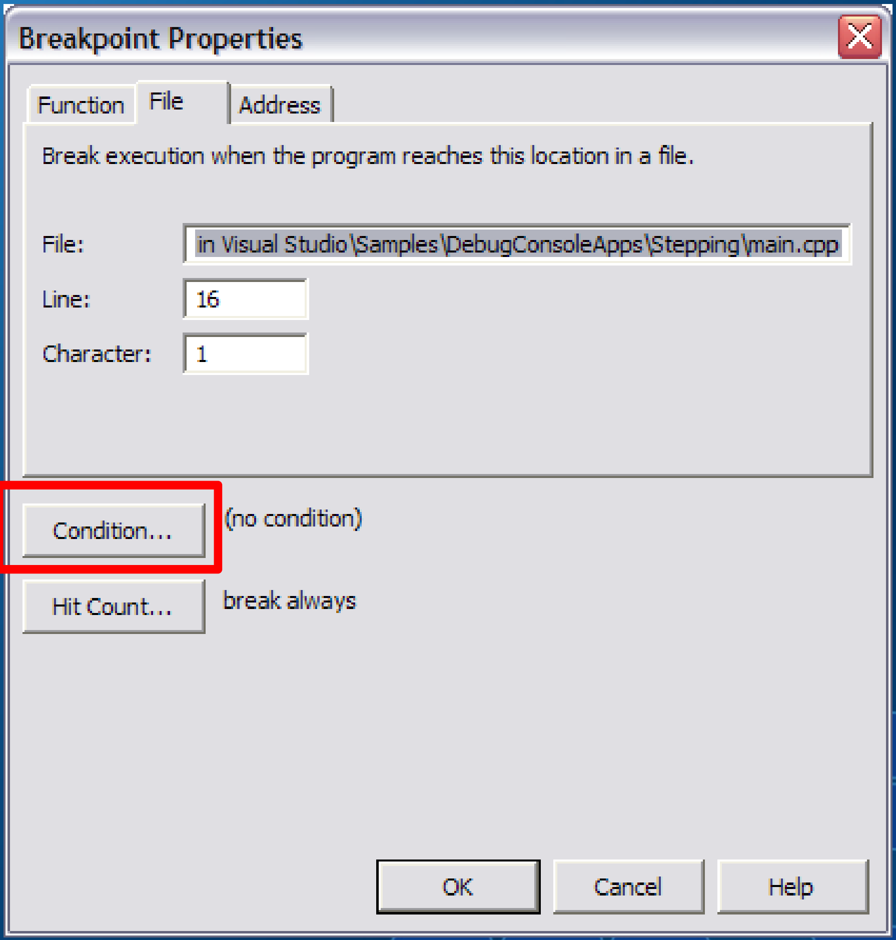
- break always
- break when the hit count is equal to
- break when the hit count is a multiple of
- break when the hit count is greater than or equal to

Microsoft® Visual Studio 2005

Example #2

Set a condition (VS 2003)

6. Click on **Condition…** to set the criteria for breaking.



Breakpoint Properties

Break execution when the program reaches this location in a file.

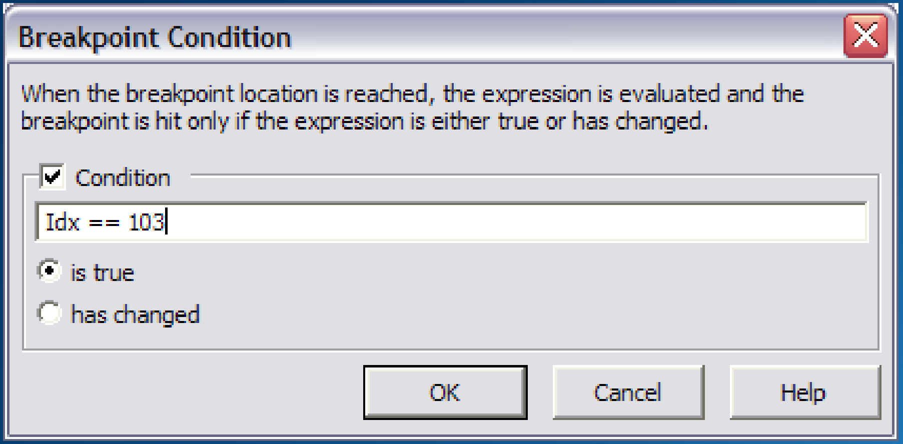
File: in Visual Studio\Samples\DebugConsoleApps\Stepping\main.cpp

Line: 16

Character: 1

(no condition)

break always



Breakpoint Condition

When the breakpoint location is reached, the expression is evaluated and the breakpoint is hit only if the expression is either true or has changed.

[✓] Condition

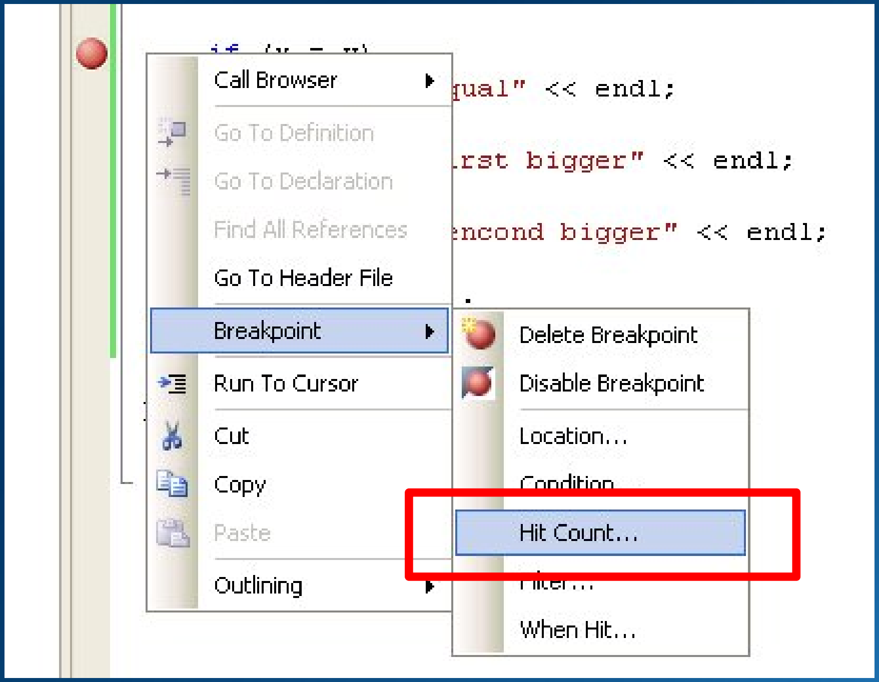
Idx == 103

[● is true] has changed

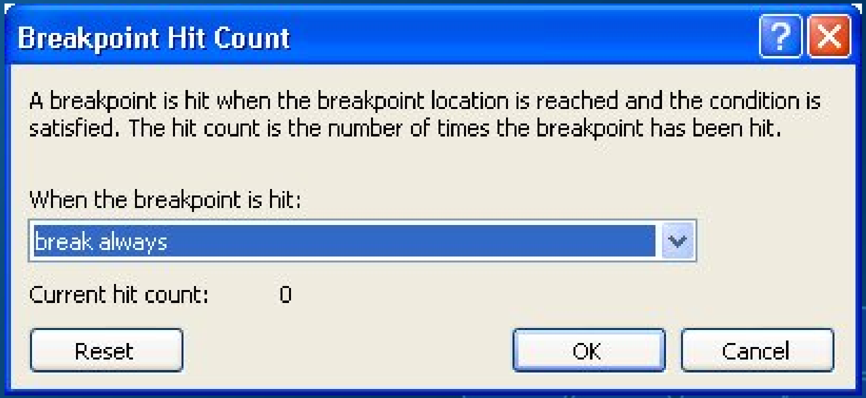
Example #2

Set hit count condition (VS 2005 & V C++ Express 2005)

- Right-mouse click on the breakpoint
- Select **Hit Count**…



```
ual" << endl;
rst bigger" << endl;
econd bigger" << endl;
```



A breakpoint is hit when the breakpoint location is reached and the condition is satisfied. The hit count is the number of times the breakpoint has been hit.

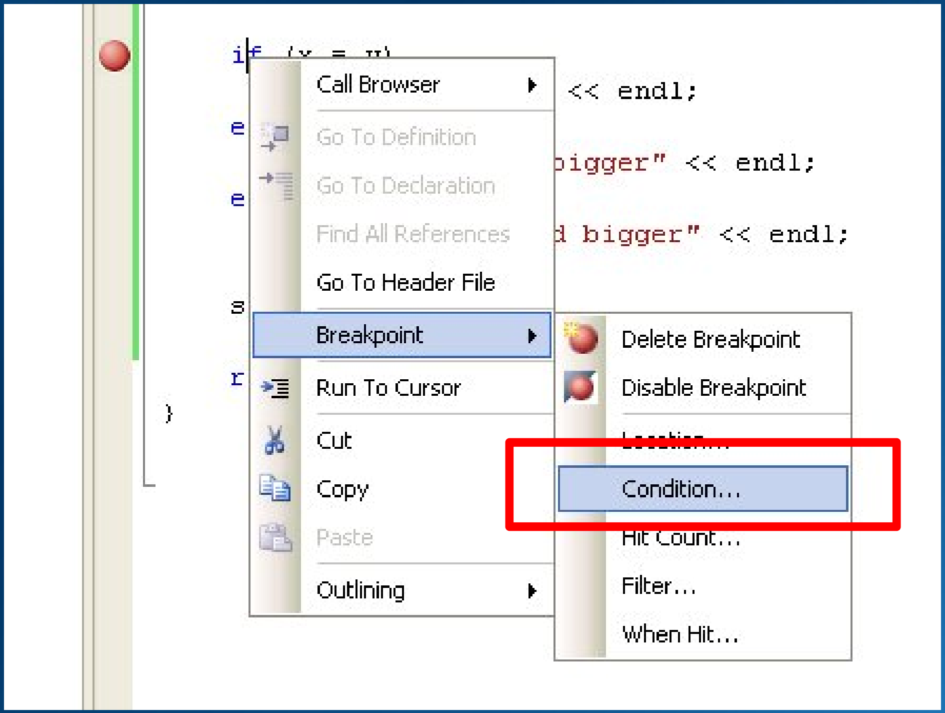
When the breakpoint is hit: break always

Current hit count: 0

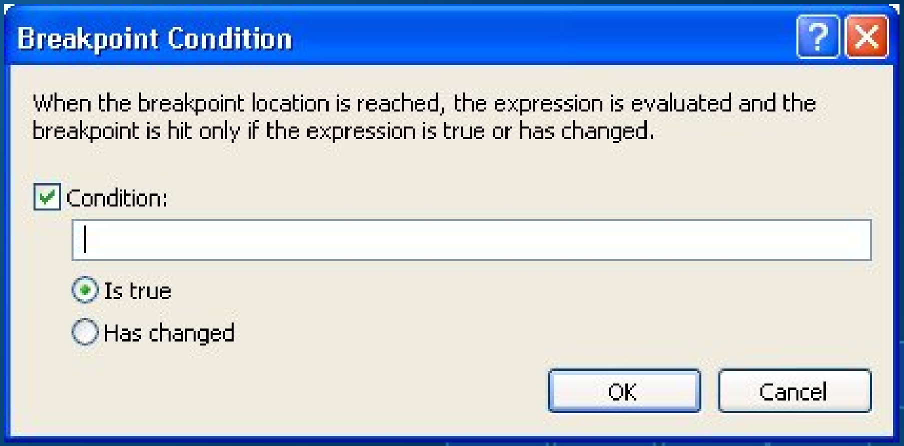
Example #2

Set a condition (VS 2005 & V C++ Express 2005)

- Right-mouse click on the breakpoint
- Select **Condition**…



```
if (x = y)
    << endl;
bigger" << endl;
d bigger" << endl;
}
```



Breakpoint Condition

When the breakpoint location is reached, the expression is evaluated and the breakpoint is hit only if the expression is true or has changed.

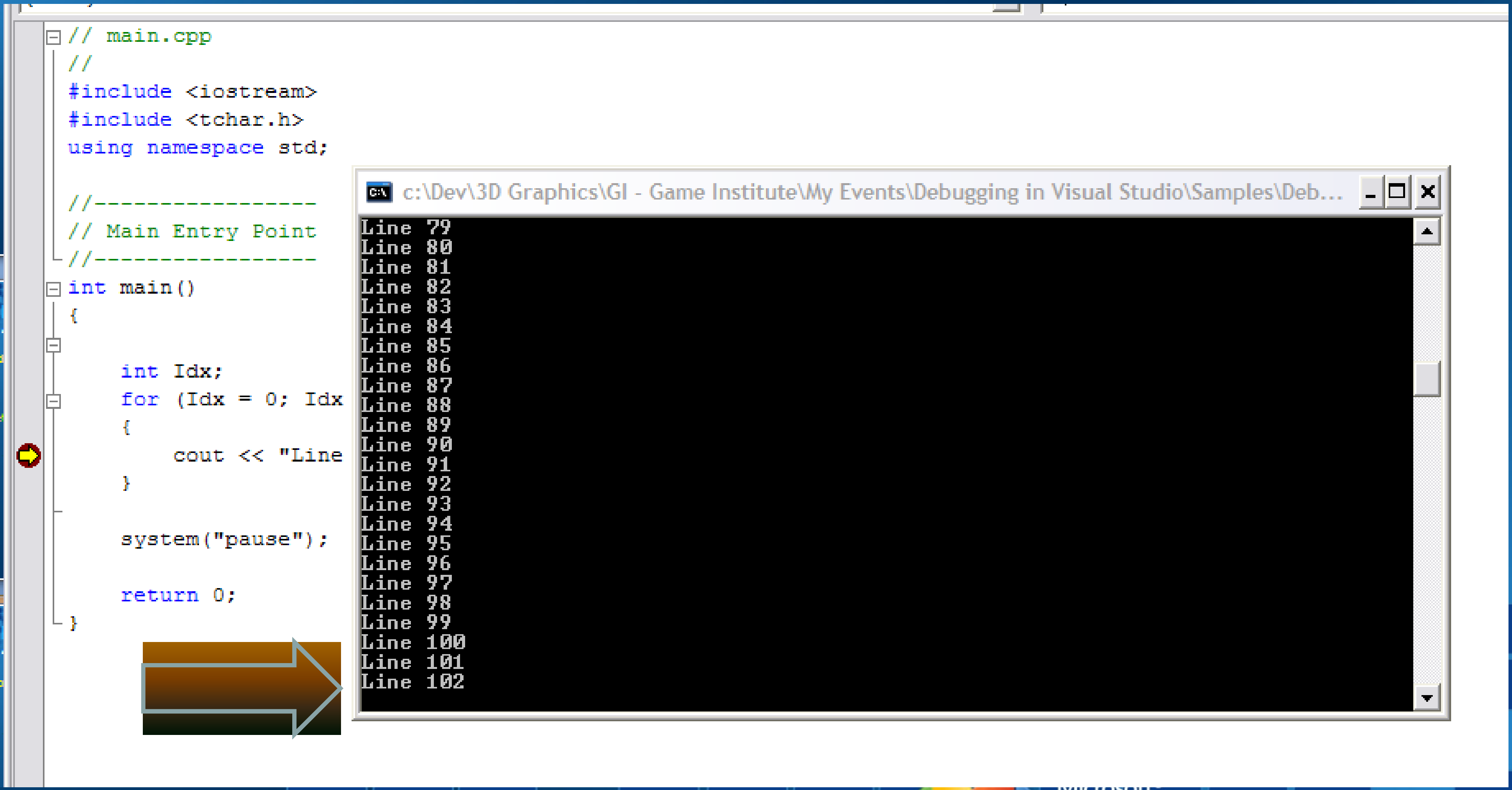
☑ Condition:

[✓ Is true]

Has changed

Example #2

Result - Code breaks at the desired condition



```
// main.cpp
//
#include <iostream>
#include <tchar.h>
using namespace std;

//------------------
// Main Entry Point
//------------------
int main()
{

    int Idx;
    for (Idx = 0; Idx
    {
        cout << "Line
    }

    system("pause");

    return 0;
}
```

c:\Dev\3D Graphics\GI - Game Institute\My Events\Debugging in Visual Studio\Samples\Deb...

```
Line 79
Line 80
Line 81
Line 82
Line 83
Line 84
Line 85
Line 86
Line 87
Line 88
Line 89
Line 90
Line 91
Line 92
Line 93
Line 94
Line 95
Line 96
Line 97
Line 98
Line 99
Line 100
Line 101
Line 102
```

Microsoft Visual Studio 2005

Microsoft Visual Studio is a registered trademark of Microsoft Corp.

Memory Leaks!

How do you know you have them?

- Basic project setup to detect them
- We will use the C Run-Time library
- After building and running the program, the output window will display any memory leaks.
- We can call another function to force a breakpoint when the suspect memory is allocated.

Microsoft® Visual Studio 2005

Memory Leaks!
Using some C Run-Time Functions

_CrtDumpMemoryLeaks()

Performs leak checking where called. You want to place this call at all possible exits of your app.

_CrtSetDbgFlag ()

Sets debugging flags for the C run-time library.

_CrtSetDbgFlag () flag	What it does
_CRTDBG_REPORT_FLAG	Gets current flag(s)
_CRTDBG_LEAK_CHECK_DF	Perform automatic leak checking at program exit through a call to _CrtDumpMemoryLeaks

Microsoft® Visual Studio 2005

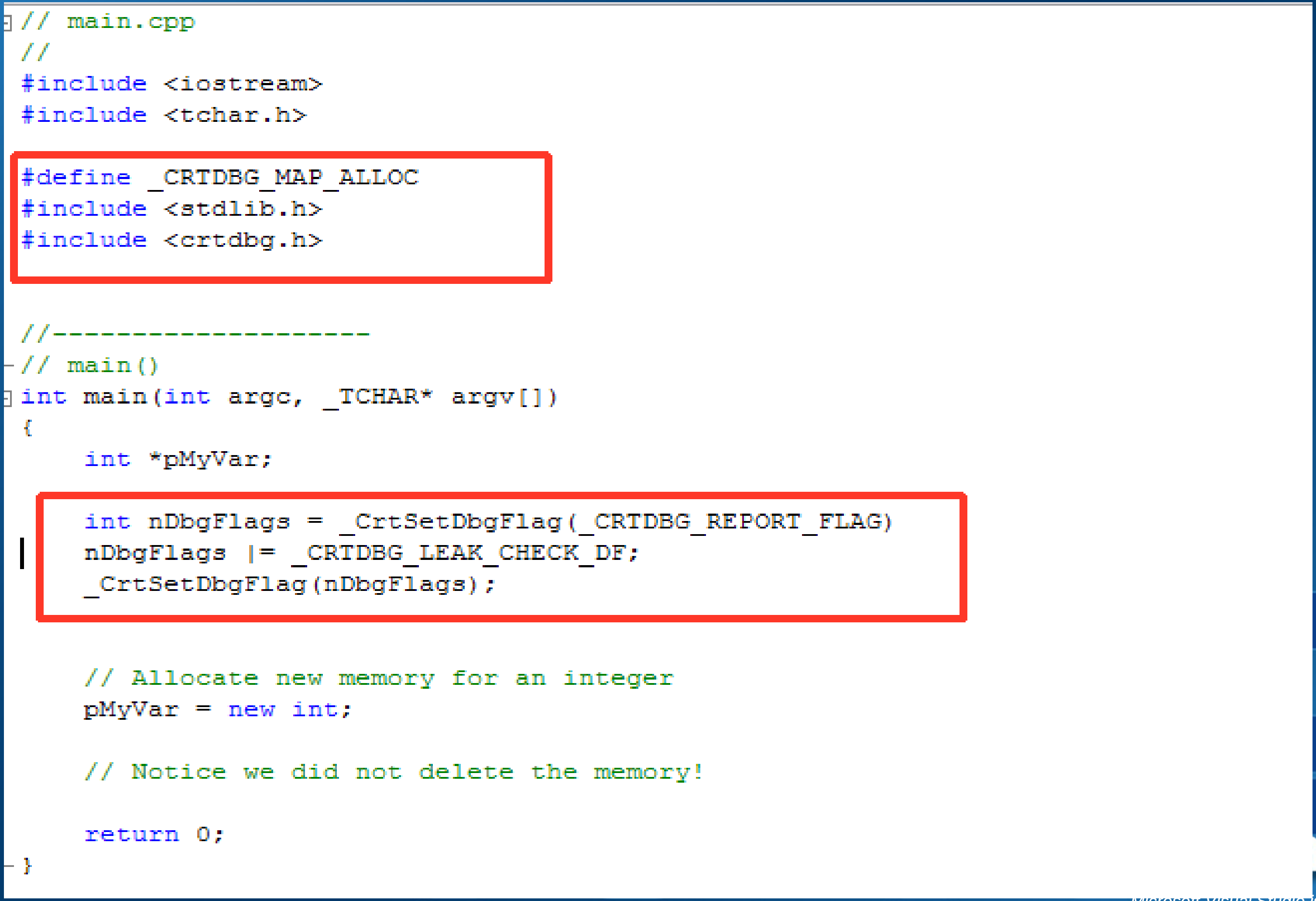
Example #3

Memory Leaks

Setting up for detection for Console or Win32

"Hook" into the C Run-time libraries to use the debug heap

1. Include the following lines in your program as the basics.



```
// main.cpp
//
#include <iostream>
#include <tchar.h>

#define _CRTDBG_MAP_ALLOC
#include <stdlib.h>
#include <crtdbg.h>


//---------------------
// main()
int main(int argc, _TCHAR* argv[])
{
    int *pMyVar;

    int nDbgFlags = _CrtSetDbgFlag(_CRTDBG_REPORT_FLAG)
    nDbgFlags |= _CRTDBG_LEAK_CHECK_DF;
    _CrtSetDbgFlag(nDbgFlags);


    // Allocate new memory for an integer
    pMyVar = new int;

    // Notice we did not delete the memory!

    return 0;
}
```

l Studio 2005

Memory Leaks

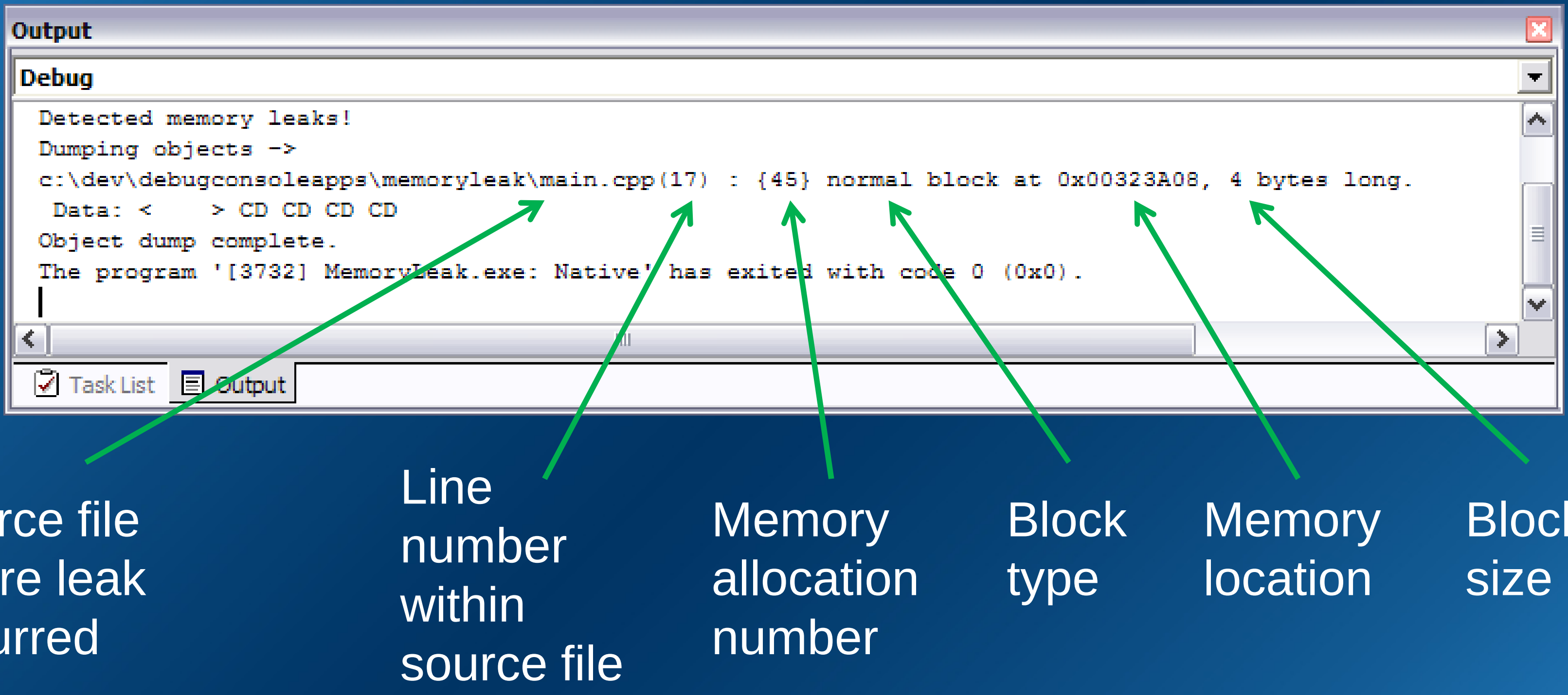
_CRTDBG_MAP_ALLOC_

- Including crtdbg.h, you map the malloc and free functions to their Debug versions, _malloc_dbg and _free_dbg, which keep track of memory allocation and deallocation
- Without `#define _CRTDBG_MAP_ALLOC`:
  - Memory allocation number (inside curly braces)
  - Block type (normal, client or CRT)
  - Memory location in hex
  - Size of block in bytes
  - Contents of the first 16 bytes in hex
- With it defined you get all the above plus:
  - File name
  - Line number

Microsoft® Visual Studio 2005

Memory Leaks

Output window dump

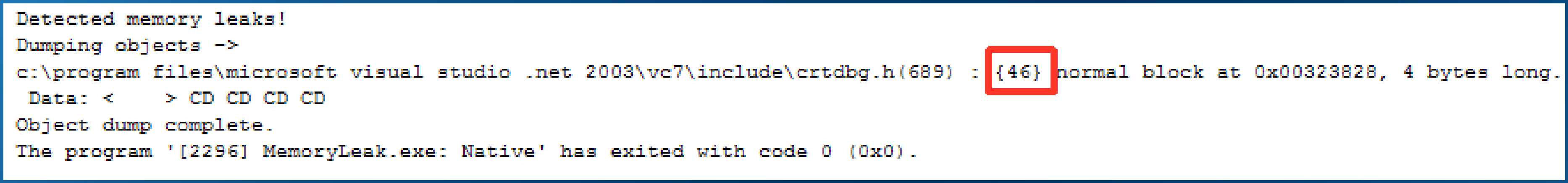


```
Detected memory leaks!
Dumping objects ->
c:\dev\debugconsoleapps\memoryleak\main.cpp(17) : {45} normal block at 0x00323A08, 4 bytes long.
 Data: <    > CD CD CD CD
Object dump complete.
The program '[3732] MemoryLeak.exe: Native' has exited with code 0 (0x0).
```

Source file where leak occurred
Line number within source file
Memory allocation number
Block type
Memory location
Block size

Memory Leaks

Locating the memory leak



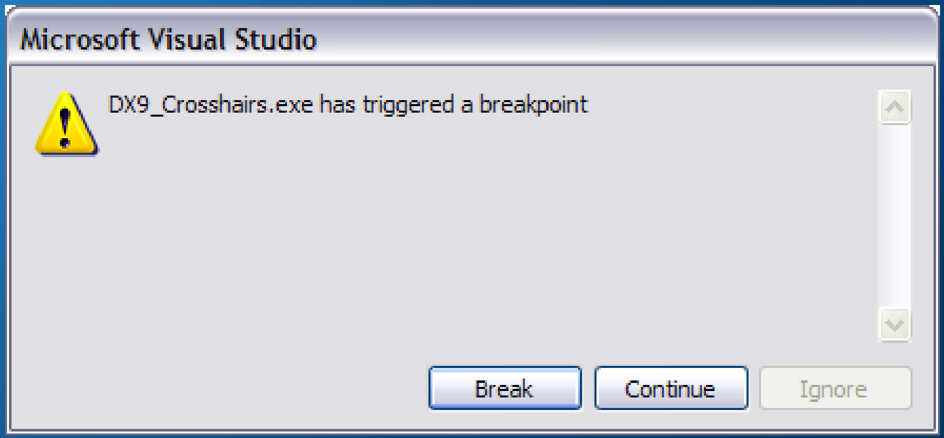
```
Detected memory leaks!
Dumping objects ->
c:\program files\microsoft visual studio .net 2003\vc7\include\crtdbg.h(689) : {46} normal block at 0x00323828, 4 bytes long.
 Data: <    > CD CD CD CD
Object dump complete.
The program '[2296] MemoryLeak.exe: Native' has exited with code 0 (0x0).
```

_CrtSetBreakAlloc(<allocation number>)

- Sets a breakpoint on a specified object allocation order number (debug version only).

Memory Leaks

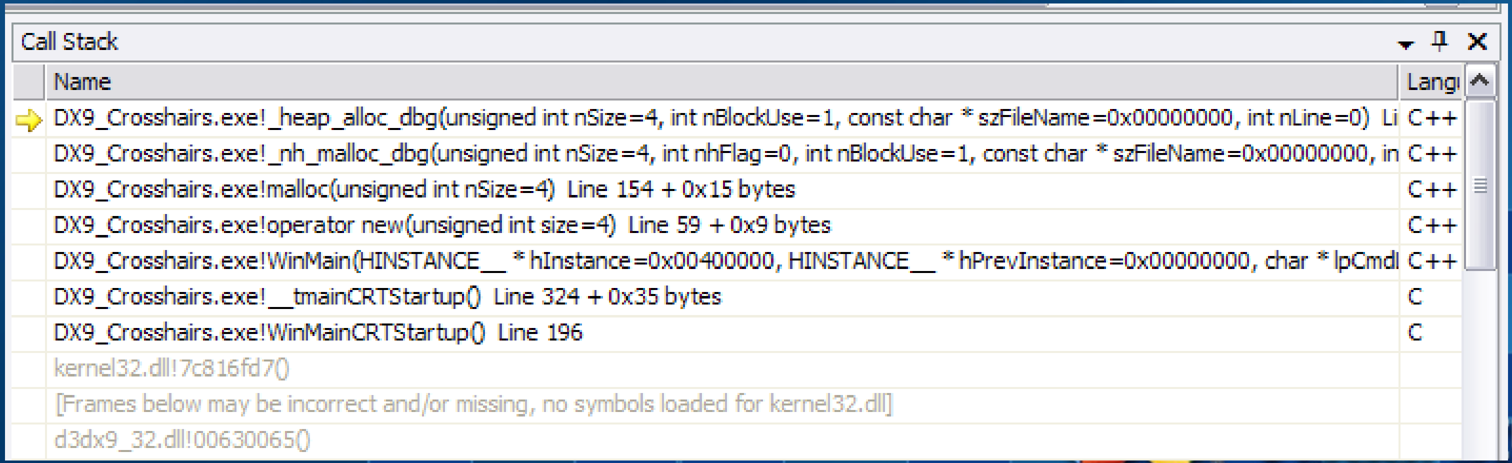
Locating the memory leak



Microsoft Visual Studio

DX9_Crosshairs.exe has triggered a breakpoint

Drill down through Call Stack window to find the last called function that belongs to your application. (not a function from a library)



Call Stack

Name	Lang
DX9_Crosshairs.exe!_heap_alloc_dbg(unsigned int nSize=4, int nBlockUse=1, const char * szFileName=0x00000000, int nLine=0) Li	C++
DX9_Crosshairs.exe!_nh_malloc_dbg(unsigned int nSize=4, int nhFlag=0, int nBlockUse=1, const char * szFileName=0x00000000, in	C++
DX9_Crosshairs.exe!malloc(unsigned int nSize=4) Line 154 + 0x15 bytes	C++
DX9_Crosshairs.exe!operator new(unsigned int size=4) Line 59 + 0x9 bytes	C++
DX9_Crosshairs.exe!WinMain(HINSTANCE__ * hInstance=0x00400000, HINSTANCE__ * hPrevInstance=0x00000000, char * lpCmdl	C++
DX9_Crosshairs.exe!__tmainCRTStartup() Line 324 + 0x35 bytes	C
DX9_Crosshairs.exe!WinMainCRTStartup() Line 196	C
kernel32.dll!7c816fd7()	
[Frames below may be incorrect and/or missing, no symbols loaded for kernel32.dll]	
d3dx9_32.dll!00630065()	

So what have we talked about. . .

- You will spend your time finding semantic errors because the compiler catches syntax errors.
- Visual Studio has a rich suite of debugging tools to help you Trace the execution and Watch variables.
- Control program execution by stopping and stepping through your code.
- Watch variable values to see if they look right.
- Use the C Run-time library for finding memory leaks.

Microsoft® Visual Studio 2005

Debugging in Visual Studio

Any questions?

You can contact me in the forums at the Game Institute or you can email me at:

smjones@gameinstitute.com
info@gameengineer.net

Microsoft® Visual Studio 2005