

Debugger Definition

A program used to control the execution of another program for diagnostic purposes.



Debugger Features / Operations

Single-Stepping

Executing a program one instruction at a time.

Variable Examination

Inspecting the changes in a variable's value during execution.

Breakpoints

Setting temporary halting places within a program.

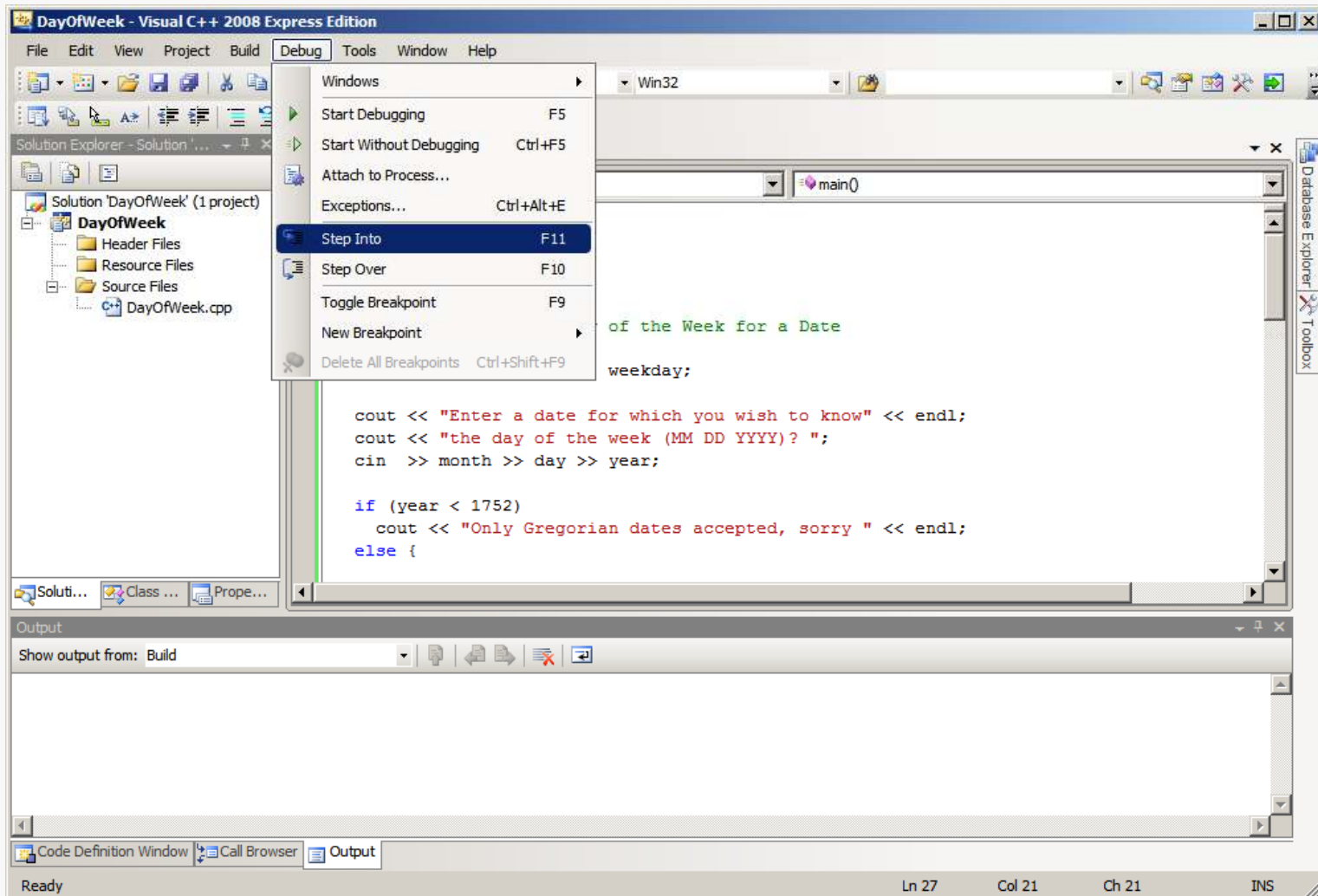
Expression Evaluation

Determining the value of an arbitrary expression during debugging execution.



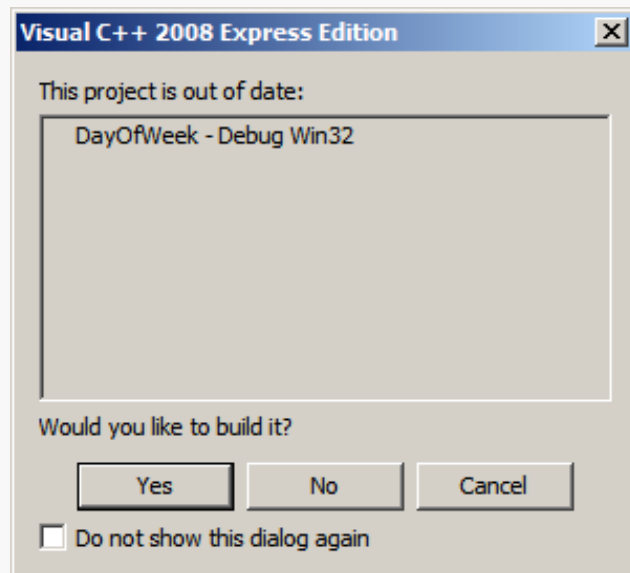
MS Visual C++ GUI Debugger

Allows interactive debugging from within the Integrated Development Environment (IDE) thru the editor window.

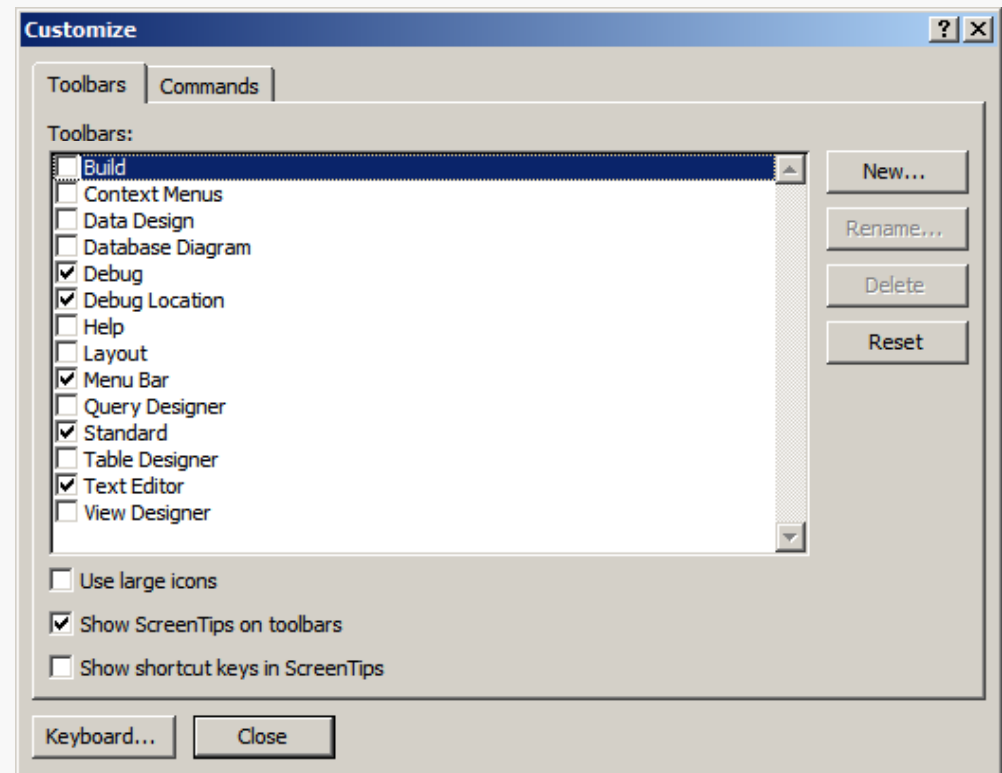


Debugging Code Generation

First load the corrected workspace you created earlier, (Day Of The Week), in the MS Visual C++ tutorial. Debugging may require recompilation to generate the debug trace data.

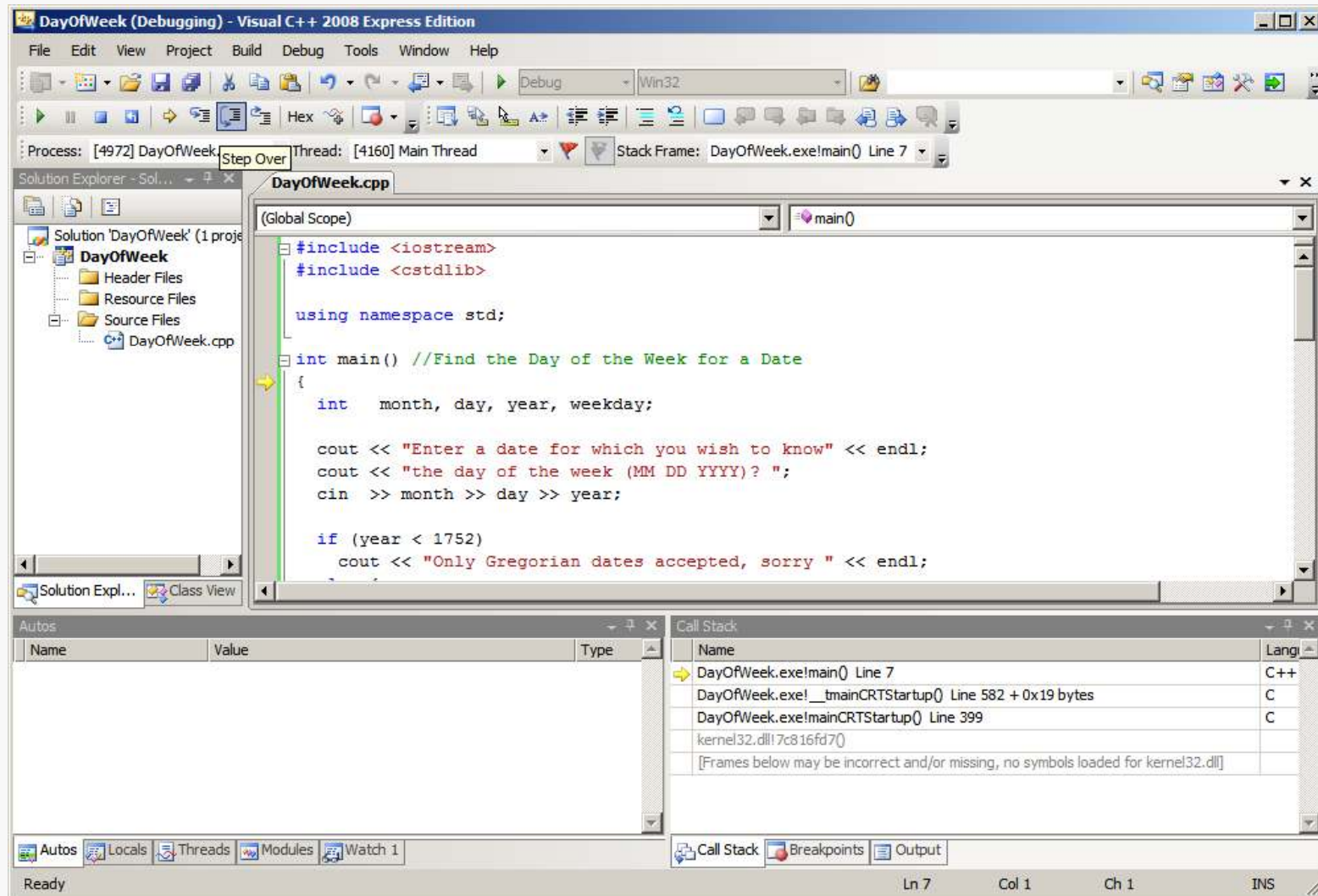


Be sure that the Debug toolbars are displayed. Select Tools menu ► Customize...



Start Trace

To Start The debugger (pausing at / and highlighting the first executable instruction): Choose Step Into (F11) from the Debug menu. This will open a code window with the first line of code pointed to:



Executing code with the debugger



Continue Trace

To continue single stepping instruction by instruction: Repeatedly hit F10 or the Step Over button:



Each click causes one statement to be executed. When you get to the part of the code that needs keyboard input, you will need to type a date in the execution window.

The execution window is used for input and output. To switch to the execution window, look on the  menu bar and click on the

 d:\Courses\Cs1044...

tab with the program name.

Halt Trace

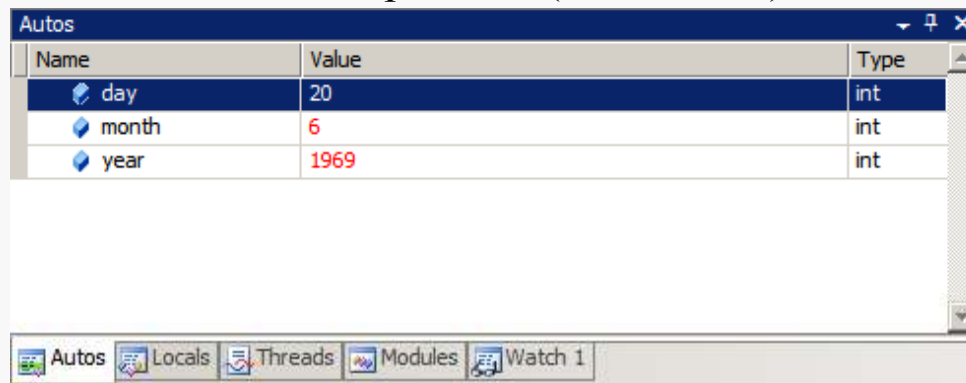
To stop debugger execution of the program: Choose Stop Debugging from the Debug menu or hit Shift+F5 or the halt debug button:



Examination Methods

If you pause the mouse over a variable name, its current value is shown in a popup box. Try this on a few variables.

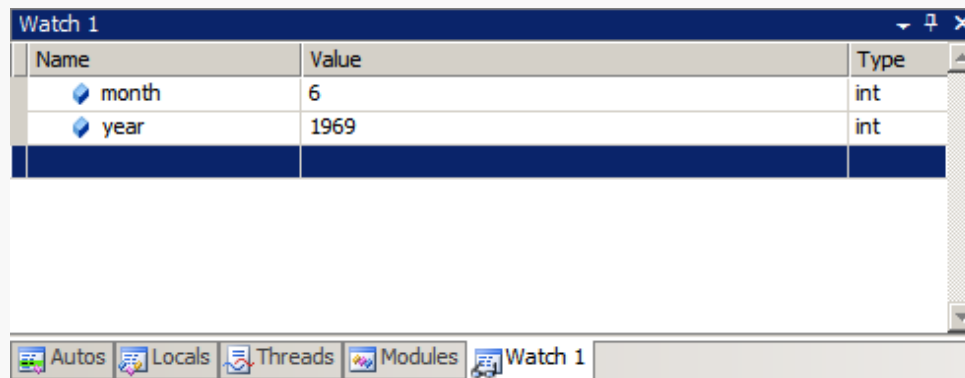
The locals (variables) pane (accessible via the Debug, Windows ► Locals menu) displays variables and their values from the current expression (the auto tab), local to the current function (the locals tab).



The Autos pane in the Visual C++ Debugger shows the current state of variables. It has a title bar 'Autos' and a table with columns 'Name', 'Value', and 'Type'. The table contains three rows: 'day' with value 20, 'month' with value 6, and 'year' with value 1969. All three variables are of type 'int'. The 'Autos' tab is selected in the bottom pane.

Name	Value	Type
day	20	int
month	6	int
year	1969	int


The watch pane (accessible via the Debug, Windows ► Watch ► Watch1 menu) allows variables & expressions to be constantly evaluated while single-stepping through the program. In this window, type month and year. As you step through the program you should notice the value of these variables being changed.

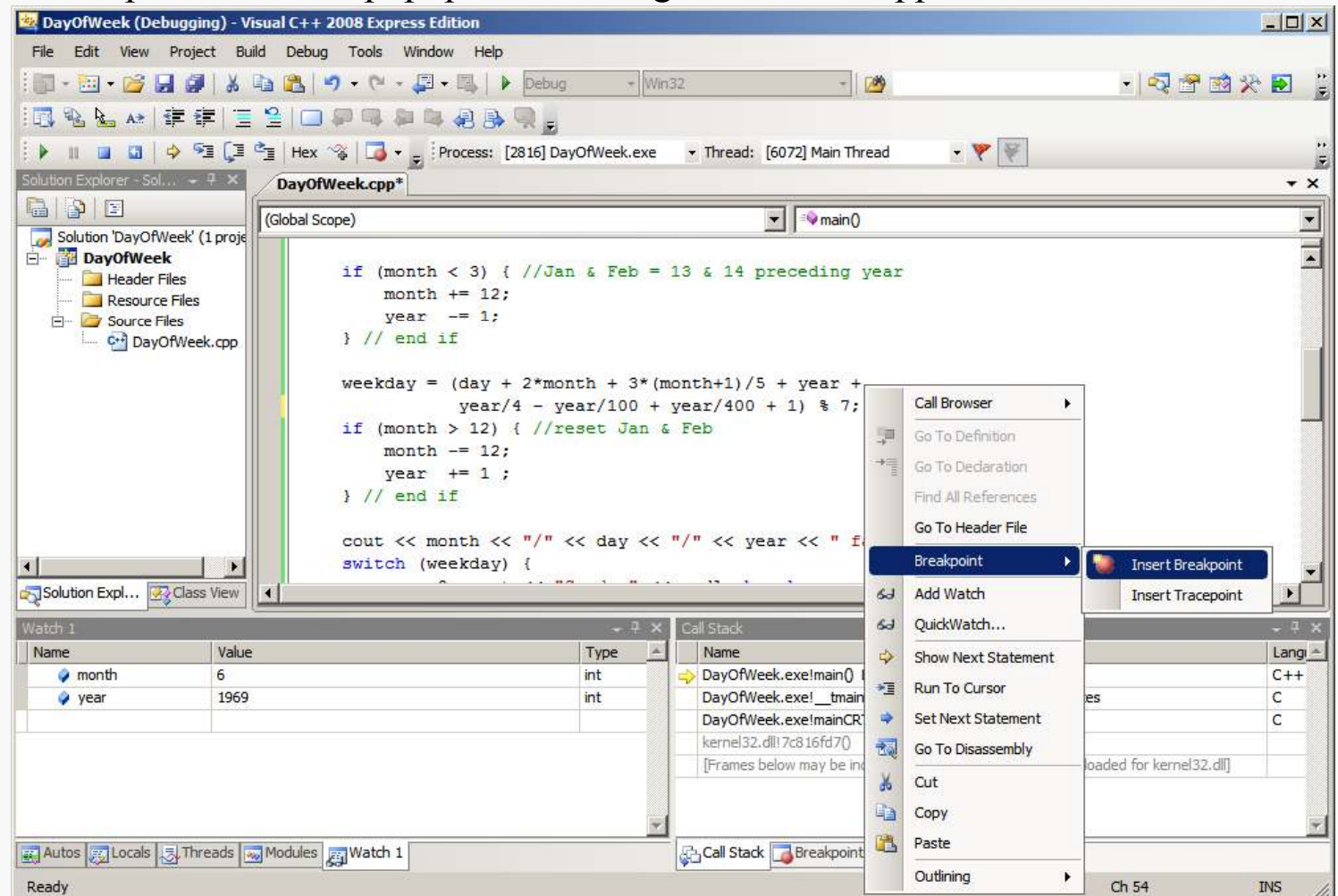


The Watch 1 pane in the Visual C++ Debugger shows variables being watched. It has a title bar 'Watch 1' and a table with columns 'Name', 'Value', and 'Type'. The table contains two rows: 'month' with value 6 and 'year' with value 1969. Both variables are of type 'int'. The 'Watch 1' tab is selected in the bottom pane.


Name	Value	Type
month	6	int
year	1969	int

Setting Unconditional Breakpoints

To display the Breakpoint pane hit the break point button  from the debug toolbar. To set an unconditional breakpoint, position the cursor at the line you wish to break execution. Right-click and select Insert Breakpoint from the popup menu. A big red dot will appear to the left of the line you selected.



Execute To The Breakpoint

To execute the program to the breakpoint you just set, click on the continue button  or hit F5. This will cause the program to run until it reaches the next breakpoint.

Set another breakpoint further along in the code. Then repeat this procedure to get to the next breakpoint. Select one of the breakpoints then right-click and select Delete Breakpoint on the popup menu to delete the breakpoint.


```
if (month < 3) { //Jan & Feb = 13 & 14 preceding year
    month += 12;
    year  -= 1;
} // end if

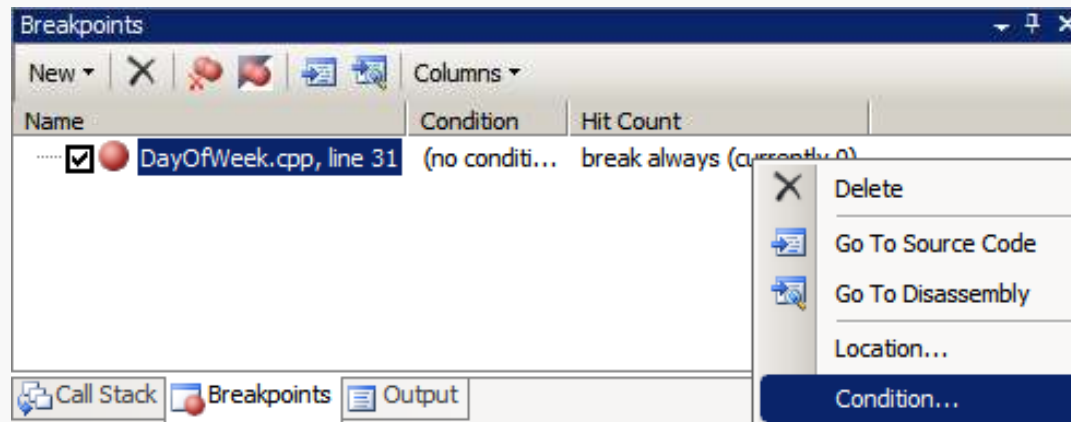
weekday = (day + 2*month + 3*(month+1)/5 + year +
          year/4 - year/100 + year/400 + 1) % 7;
if (month > 12) { //reset Jan & Feb
    month -= 12;
    year  += 1 ;
} // end if

cout << month << "/" << day << "/" << year << " falls on ";
switch (weekday) {
```


Boolean Breakpoints

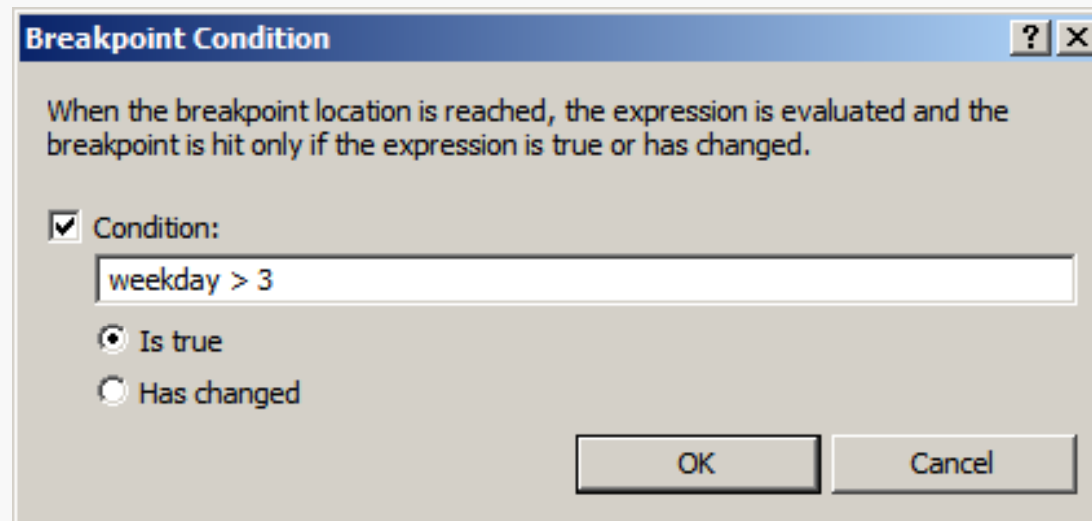
Conditional breakpoints only stop/pause program execution if a specified condition is true. They are commonly used to halt the program during loop execution.


Display the Breakpoint pane, hit the break point button  from the debug toolbar. To make a conditional breakpoint, select the Breakpoint from the Breakpoint pane and right-click. Select Condition... from the popup menu.



Breakpoint Modification

In the Breakpoint Condition dialog type the Boolean expression that you wish to check, as the example shown in the following image:



You can use the continue button  to execute the program until your conditional breakpoint is reached (if, indeed, it ever is reached).