

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.

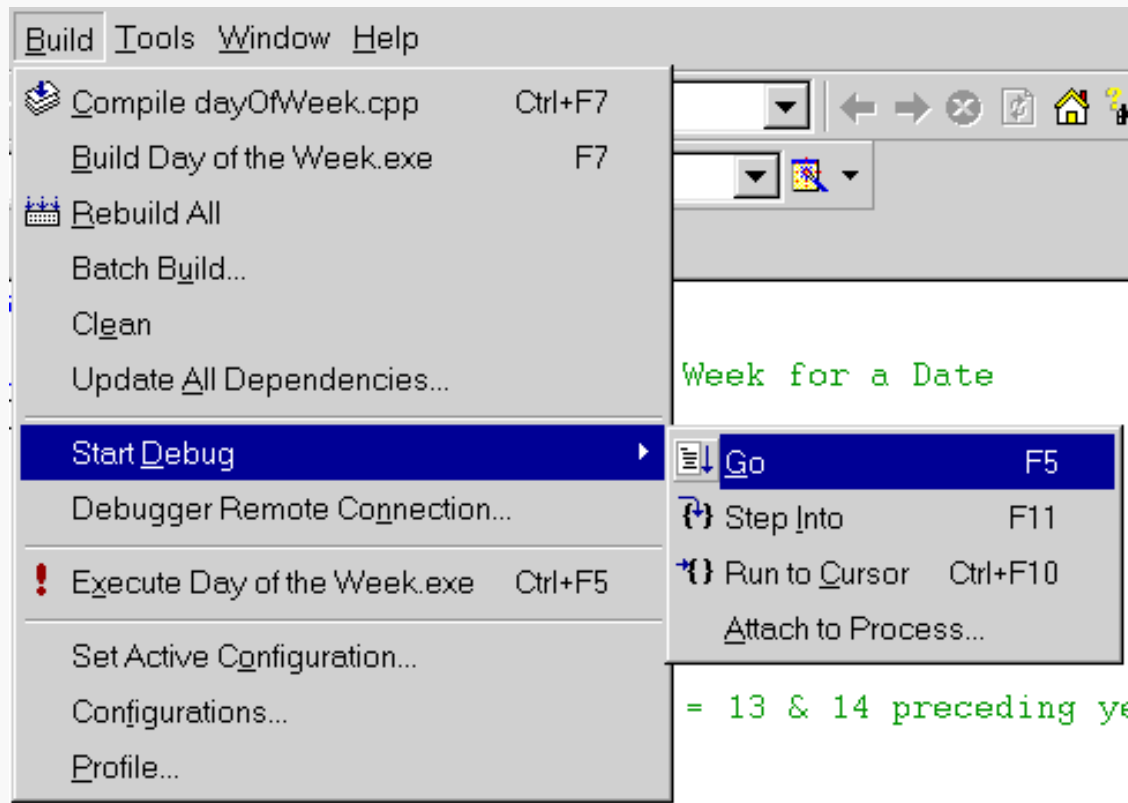
Expression Evaluation

Determining the value of an arbitrary expression during debugging execution.



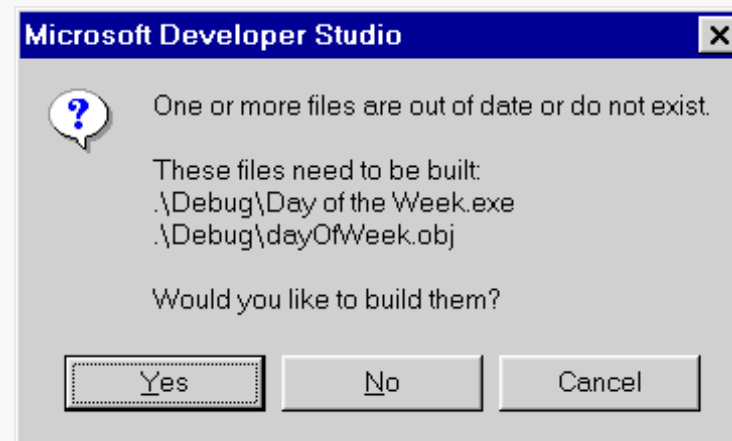
MS Visual C++ GUI Debugger

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 trace data.

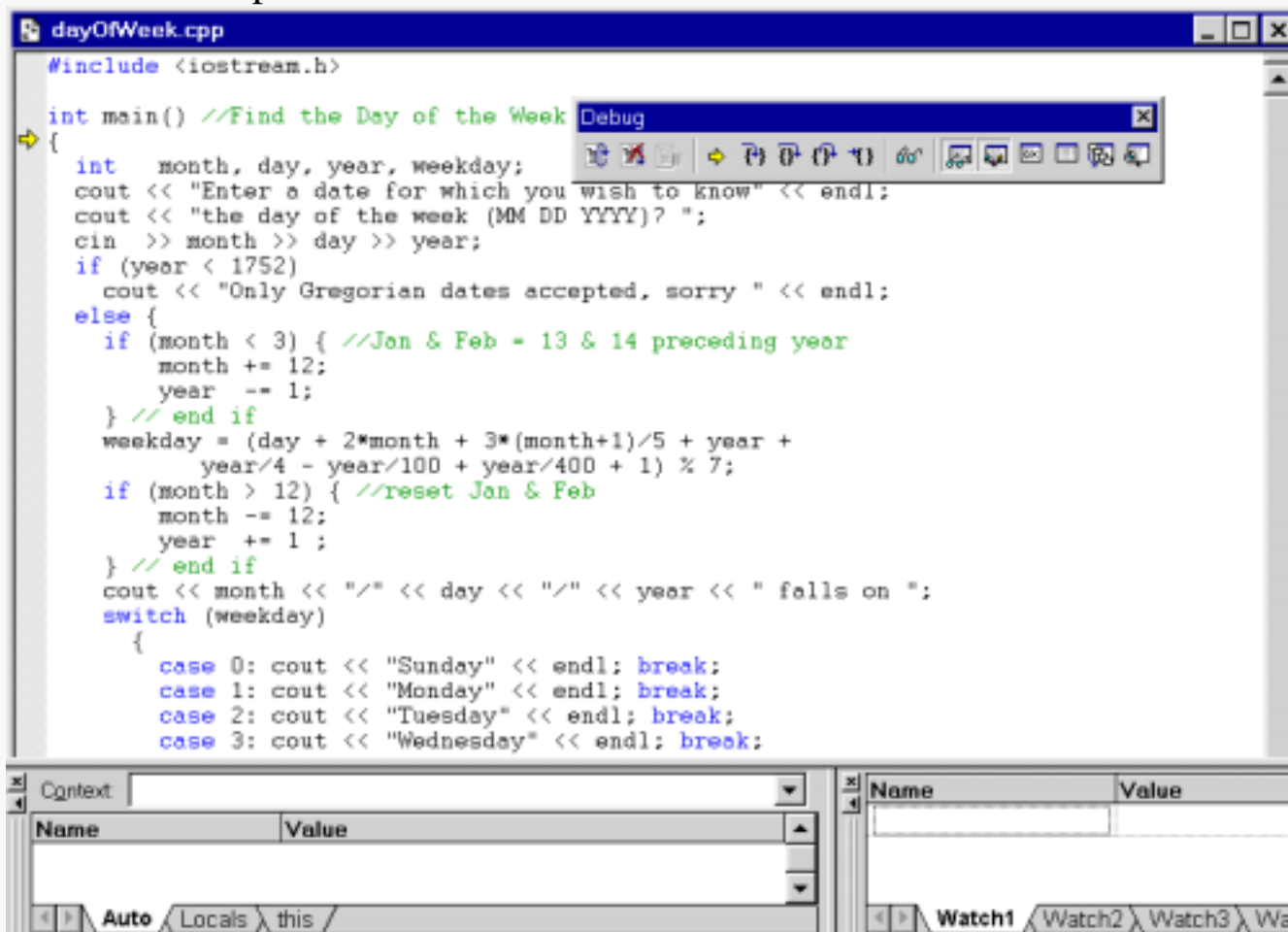


When the debugger is running, the Build menu is replaced by the Debug menu.

Begin Debug Trace


Start Trace

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



Executing code with the debugger

Run to Cursor


Click somewhere lower in the code and choose Run to cursor (Ctrl+F10)  from the Debug menu (or floating toolbar). This will cause the program to run until it reaches the line with the code you clicked on.

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  Start tab with the program name.



Halt Trace

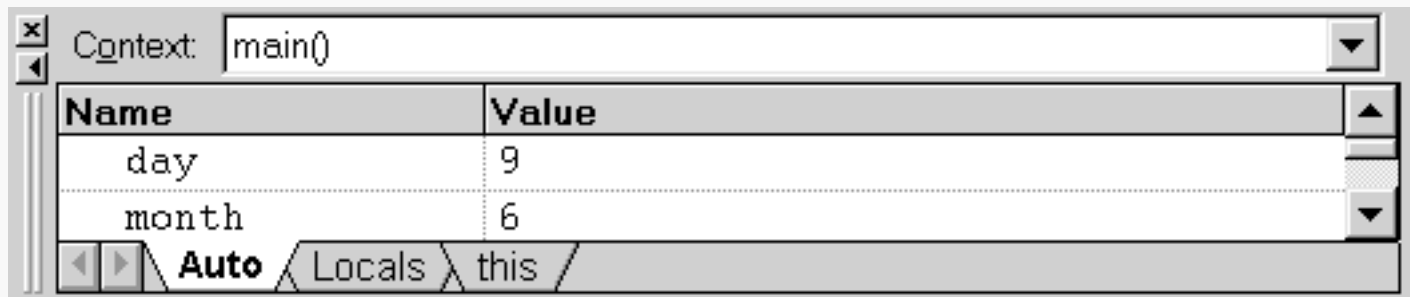
To stop debugger execution of the program: Choose Stop Debugging from the Debug menu or hit Shift+F5 or the halt debug button:



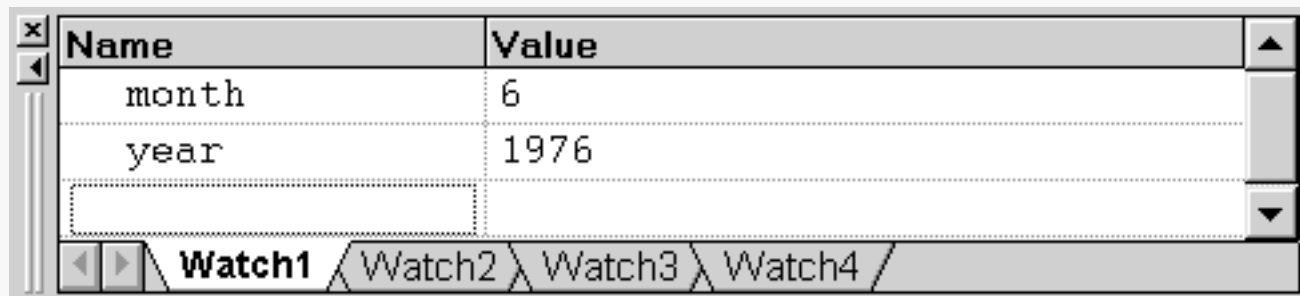
Examination Methods

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 variables window (accessible via the view menu) displays variables and their values from the current expression (the auto tab), local to the current function (the locals tab), or in relation to the *this* pointer (the *this* tab).



The watch window (accessible via the view 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.




Setting Unconditional Breakpoints

To set an unconditional breakpoint, position the cursor at the point you wish to break execution.

Now click on the breakpoint toggle button  or hit F9. You should see a big red dot 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 go 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.

```
#include <iostream.h>

int main() //Find the Day of the Week for a Date
{
    int    month, day, year, weekday;
    cout << "Enter a date for which you wish to know" << endl;
    ● | cout << "the day of the week (MM DD YYYY)? ";
    cin  >> month >> day >> year;
    if (year < 1752)
```

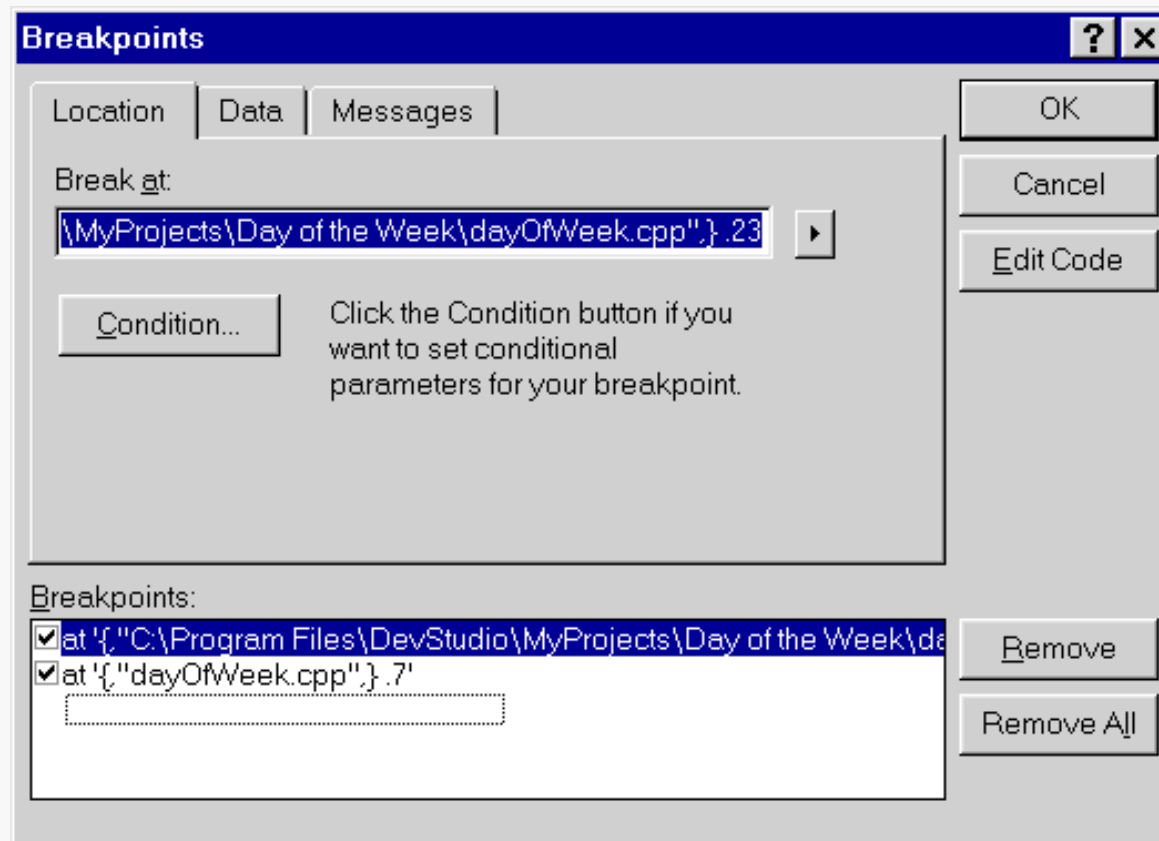
Select one of the breakpoints and click on the breakpoint toggle button  or hit F9 to remove the breakpoint.

The previous debugging session may need to be stopped and restarted at this point in order to set breakpoints.

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.

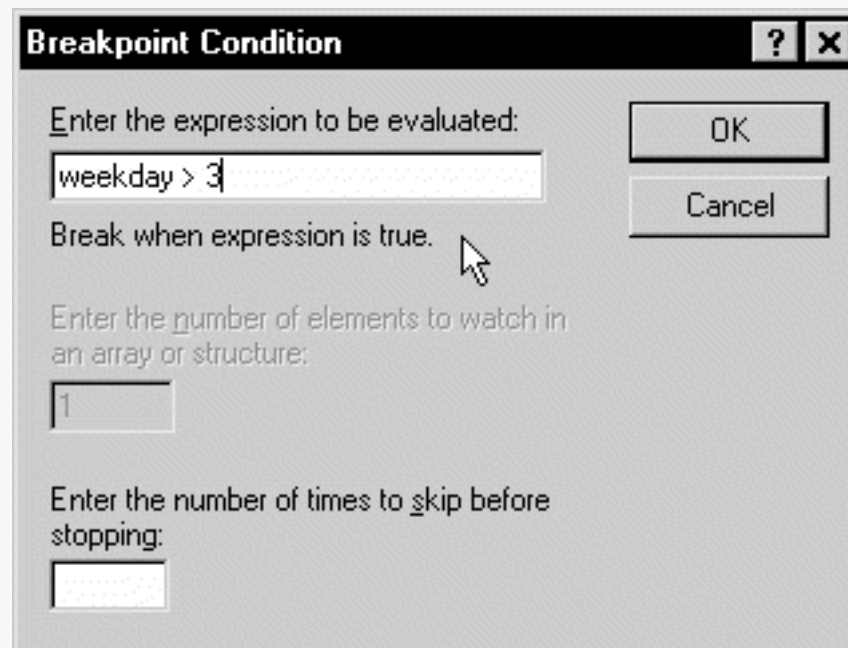
To make a conditional breakpoint, select Breakpoints from the Edit menu (Alt + F9).



Breakpoint Modification

Click on one of the breakpoints you created before (bottom of the window) or select the cursor line from the, (), popup menu to create a new breakpoint.

Now click the Condition button and type the Boolean expression that you wish to check, as the example shown in the following image:



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