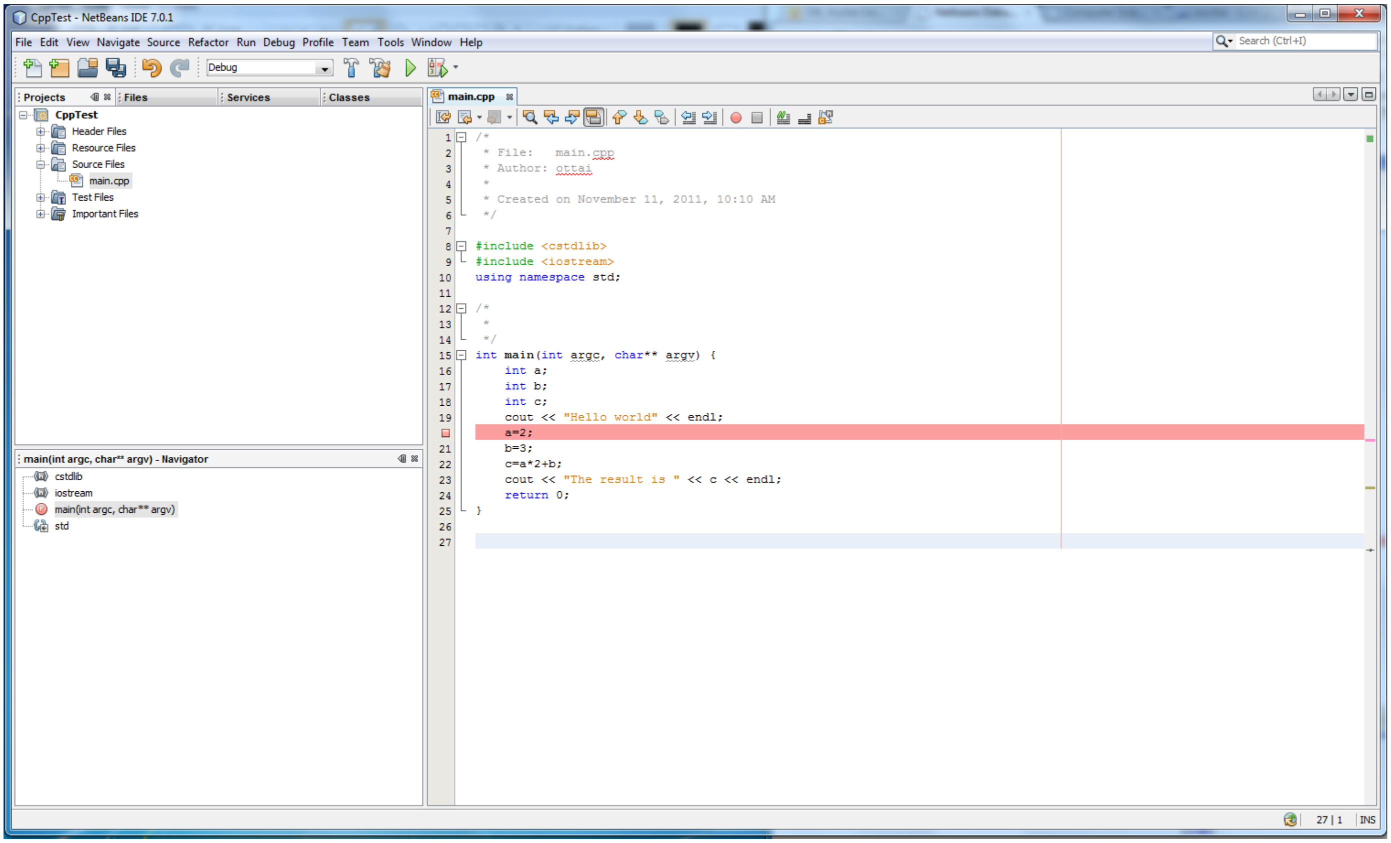
The NetBeans Debugger: A Brief Tutorial

Based on a tutorial by Anousha Mesbah from the University of Georgia

NetBeans provides a debugging tool that lets you trace the execution of a program step by step. You can debug by setting breakpoints so that you can stop and examine live values, or by setting "watches" to watch variables as they change. You can execute your program line by line and examine values within your application to help locate errors.

1. Setting Breakpoints

A breakpoint is a mark in the source code that tells the debugger to stop when execution reaches it. When your program stops on a breakpoint, you can inspect the current values of the variables in your program, then continue the execution of your program one line at a time. To set a breakpoint, click in the gray area of the left margin on the line on which you want to place the breakpoint. NetBeans highlights that line in red and puts a red square mark in the left margin to indicate the breakpoint. To remove a breakpoint, click the red square.

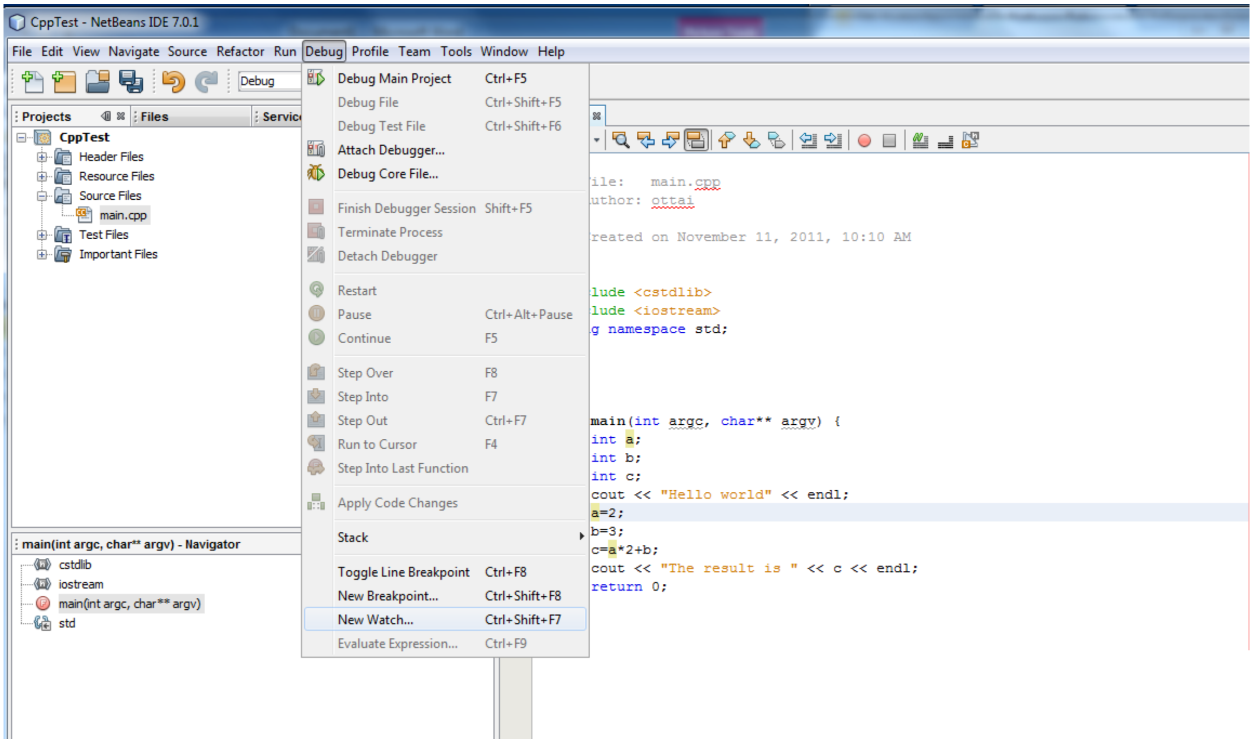

CppTest - NetBeans IDE 7.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Search (Ctrl+I)
Debug
Projects
Files
Services
Classes
CppTest
Header Files
Resource Files
Source Files
main.cpp
Test Files
Important Files
main(int argc, char** argv) - Navigator
cstdlib
iostream
main(int argc, char** argv)
std
main.cpp
27 | 1
INS

```
/*
 * File:   main.cpp
 * Author: ottai
 *
 * Created on November 11, 2011, 10:10 AM
 */

#include <cstdlib>
#include <iostream>
using namespace std;

/*
 *
 */
int main(int argc, char** argv) {
    int a;
    int b;
    int c;
    cout << "Hello world" << endl;
    a=2;
    b=3;
    c=a*2+b;
    cout << "The result is " << c << endl;
    return 0;
}

```

2. Viewing Variables

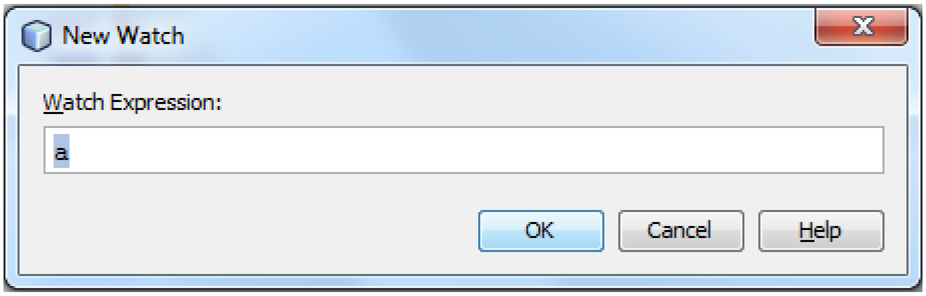
There are several ways to examine the values of variables or expressions during the execution of your program. When your program is stopped at a breakpoint, an easy way is to hover your mouse over the variable. The debugger will display the value of the variable closest to where the cursor is placed. Another way is to add a "watch" on the variable. Variables that have watches on them are displayed in their own list so that you can easily see information about them. To add a watch to a variable or expression, select the variable or expression you want to monitor, then do one of the following:

1. Select ***New Watch*** in the Run menu.


CppTest - NetBeans IDE 7.0.1
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Debug
Projects
Files
Servic
CppTest
Header Files
Resource Files
Source Files
main.cpp
Test Files
Important Files
main(int argc, char** argv) - Navigator
cstdlib
iostream
main(int argc, char** argv)
std
Debug Main Project Ctrl+F5
Debug File Ctrl+Shift+F5
Debug Test File Ctrl+Shift+F6
Attach Debugger...
Debug Core File...
Finish Debugger Session Shift+F5
Terminate Process
Detach Debugger
Restart
Pause Ctrl+Alt+Pause
Continue F5
Step Over F8
Step Into F7
Step Out Ctrl+F7
Run to Cursor F4
Step Into Last Function
Apply Code Changes
Stack
Toggle Line Breakpoint Ctrl+F8
New Breakpoint... Ctrl+Shift+F8
New Watch... Ctrl+Shift+F7
Evaluate Expression... Ctrl+F9

```
ile:   main.cpp
uthor: ottai

reated on November 11, 2011, 10:10 AM

lude <cstdlib>
lude <iostream>
g namespace std;


main(int argc, char** argv) {
int a;
int b;
int c;
cout << "Hello world" << endl;
a=2;
b=3;
c=a*2+b;
cout << "The result is " << c << endl;
return 0;
```

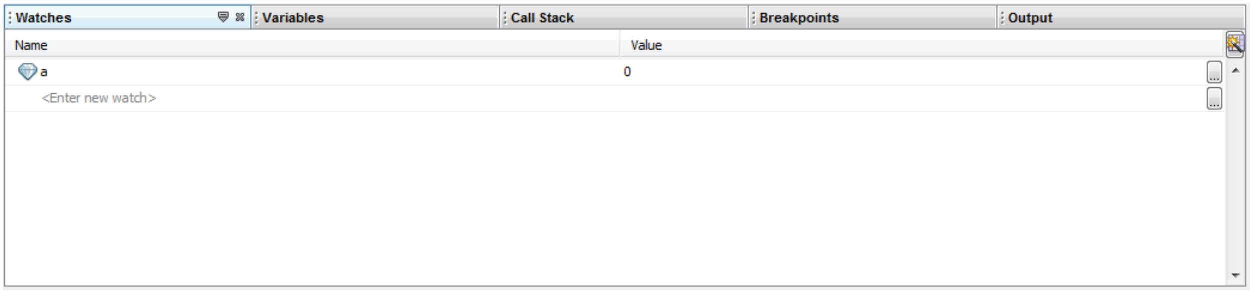
The ***New Watch*** window pop up, click ***OK*** to add the variable to the ***Watch*** list.



Watch Expression: a

2. Right-click the variable, then click ***New Watch***. Click on the OK button in the New Watch Window to add the variable to the watch list.

The ***Watches window*** displays the list of current watches and their types and values.



Name	Value
a	0
<Enter new watch>	

If the Watch window isn't visible, choose ***Debugging***>***Watches*** in the ***Windows*** menu.

To remove a watch from the list, right-click on the watch you want to remove, then choose ***Delete***.

3. Running your program in Debug Mode

There are several ways to execute your program in **debug mode**.

1. Set the breakpoints you want, then choose ***Debug Main Project*** in the ***Run*** menu to execute the program to the first breakpoint. (If you didn’t set any breakpoints, the program will run until it terminates, even in debug mode.)

2. Choose ***Step Into*** in the ***Run*** menu to run your program to the first line of the main method of your program. At this point you can examine the values of variables in the main program, then continue the execution of your program.

3. Plant the typing cursor on a line where you want execution to pause, then choose ***Run to Cursor*** in the ***Run*** menu. Execution will stop at the line in the source code where the typing cursor is currently located.

Once the execution of your program has stopped, you can trace the execution of your program step-by-step using the following buttons (or their keyboard shortcuts):

Step Over ( ) - Executes one line of code. If the line is a call to a method, **step over** executes the entire method as a single command, without stepping into the method's code.

Step Into ( ) - Executes one line of code. If the line is a call to a method, **step into** steps into the method and stops on the first line of the method.

Step Out ( ) - Executes one line of code. However, if the source code line is in a method, it will execute all the remaining code in the method and return to the method's caller.

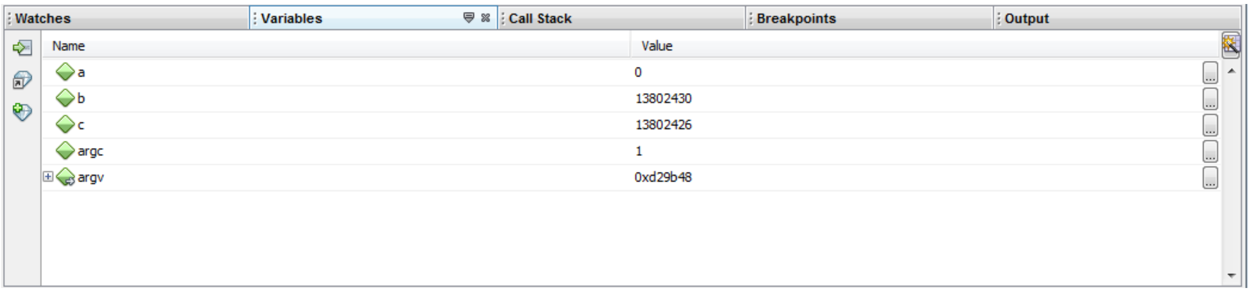
Run to Cursor ( ) - Executes the program to the line where the typing cursor is located.

Continue ( ) - Continues the execution of program at full speed until the next breakpoint or until the program terminates.

---

4. The Local Variables Window

The ***Local Variables*** window displays the name, data type and values of all the values in the current scope. To activate the ***Local Variables*** window, in the **Windows** menu select ***Debugging*** > ***Local Variables***. The debugger allows you to change the value of a variable in the Local Variables window and then continue the execution of your program using that new variable's value.



Name	Value
a	0
b	13802430
c	13802426
argc	1
argv	0xd29b48

---

5. Stopping the Debugger

To stop the debugger and return to normal execution mode, in the ***Run*** menu choose ***Finish Debugger Session***.