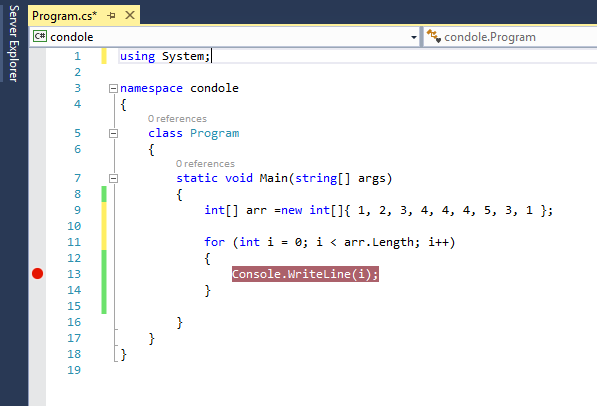
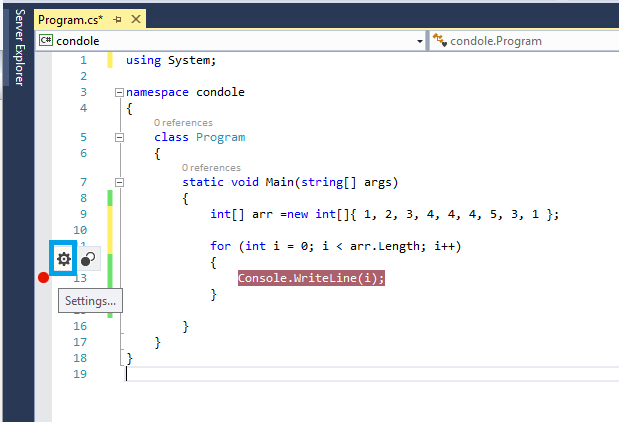
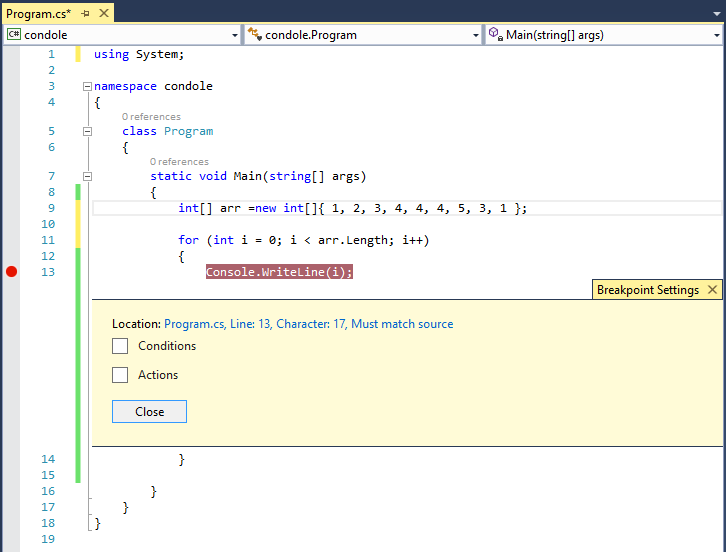
stop debugger after a specific number of breakpoint hit

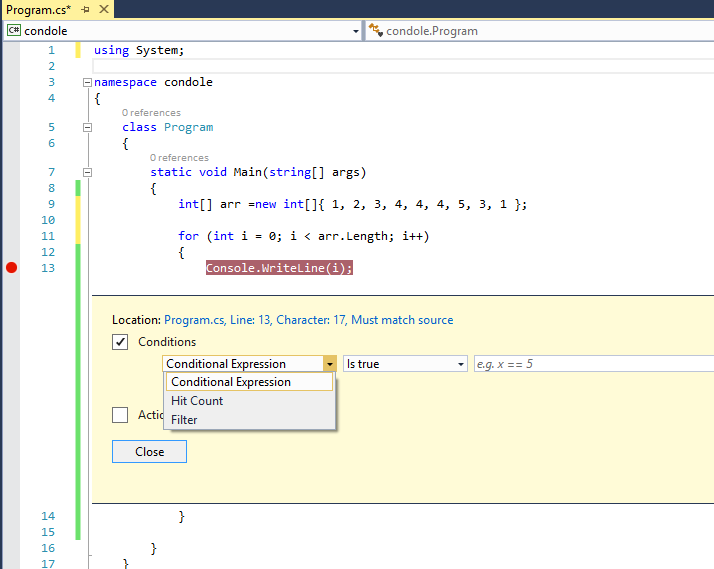
You can stop the debugger at breakpoint after a specific number of breakpoint hit.

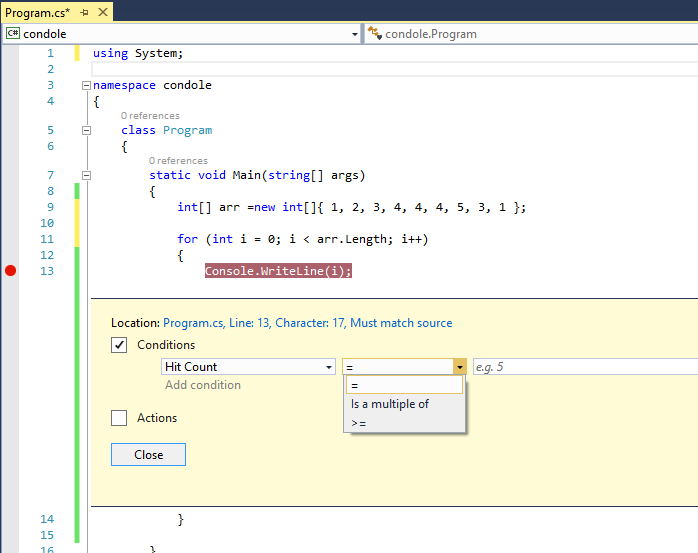
**Hit Count** is used to keep track on how many times the debugger has paused at some particular breakpoint.



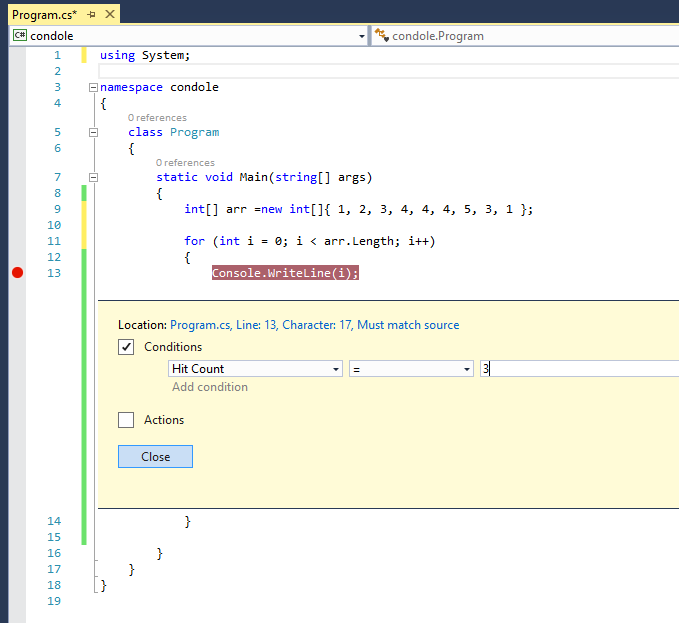


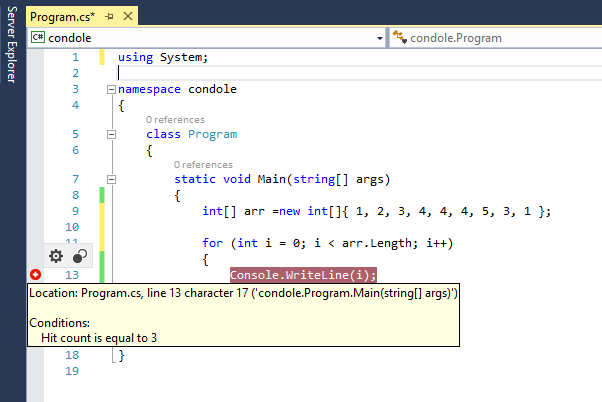




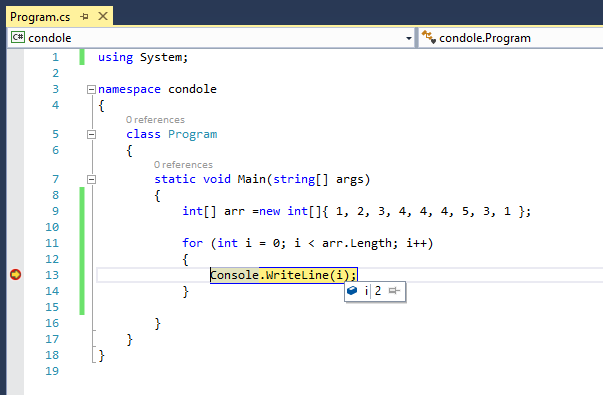


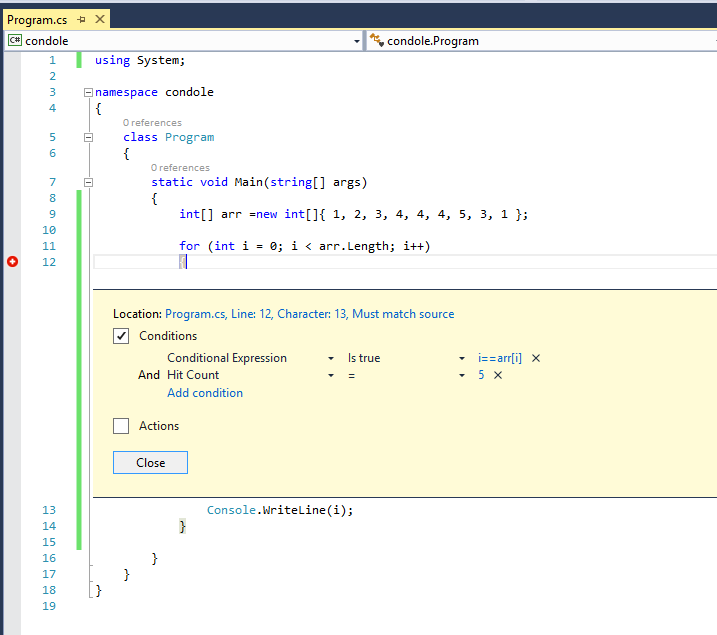
By default, it’s set to always.  So, whenever the breakpoints hits, hit count will increase automatically but the debugger will stop every time in the breakpoint.  You can also specific the count of hit over here to stop the debugger. For this example we can selection option 2 and set the values of count to 3 as shown in figure below.

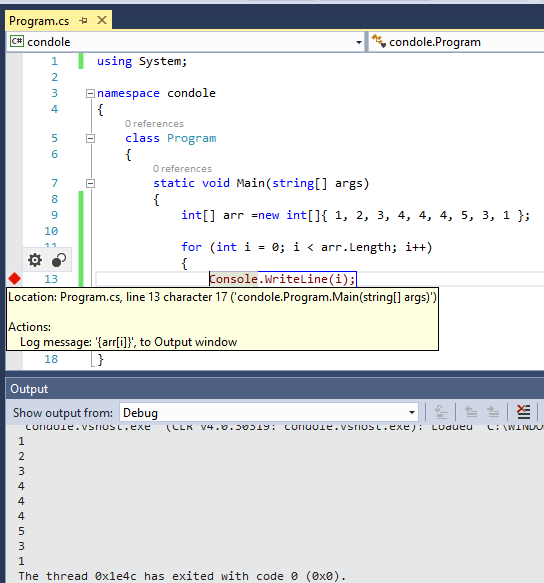


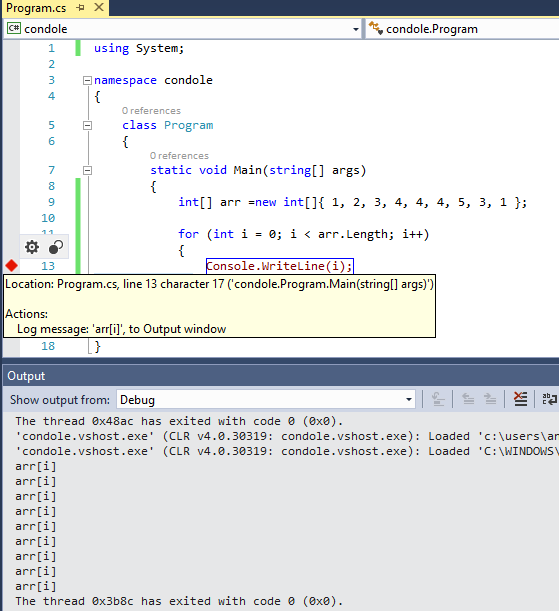


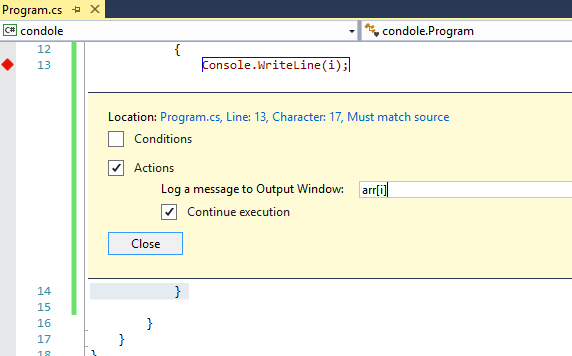
Now, if you try to debug the application, the debugger will only stop for the items when break point hit count is equal to 3.





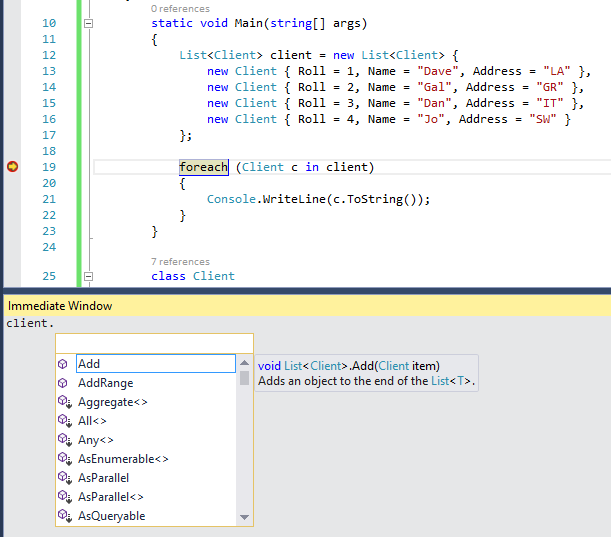




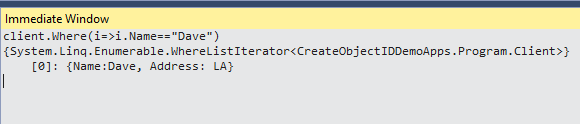


## Immediate Window – Run, Debug & Inspect Lambda Expression

You will start getting all LINQ expression, extension method and write you own query too.

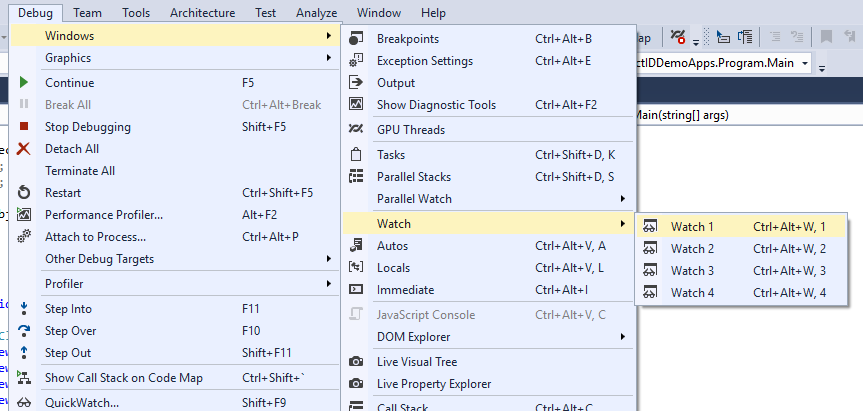


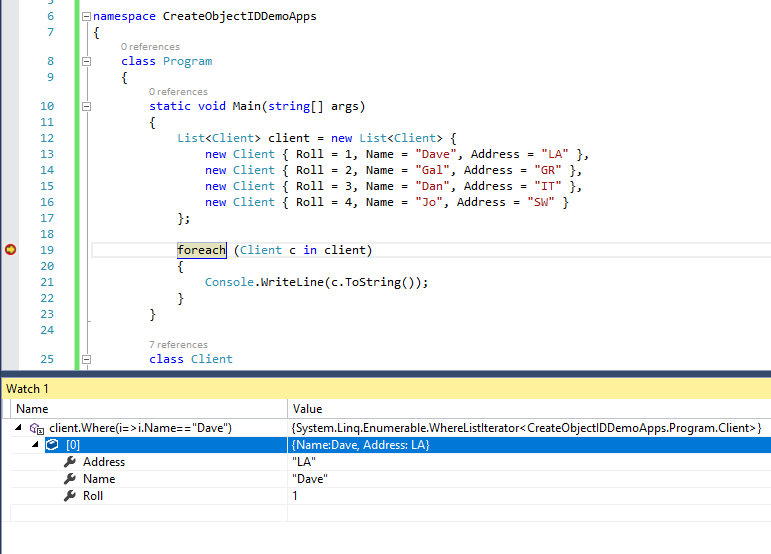
You can simply execute or debug the Lambda expression.



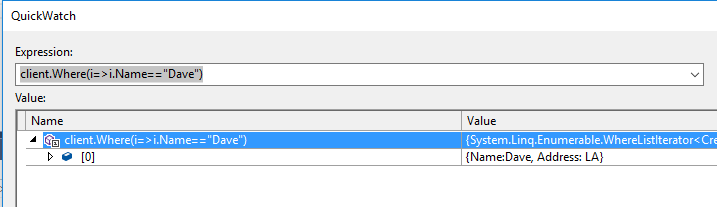
## Watch Window – Run, Debug & Inspect Lambda Expression

Watch windows is one of most commonly used debugging tool with Visual Studio. We generally used to explore the objects, values, properties and other nested objects as a tree structure.  We can do lot of stuff using Watch Window. Now we can do, debugging of Lambda Expression.





And Of course, same things you can do it for **Quick Watch** Too.

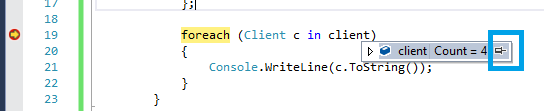


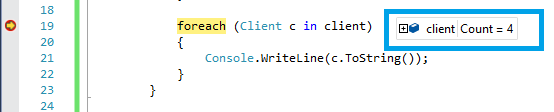
## DataTips – Run, Debug & Inspect Lambda Expression

DataTips is an advanced tooltip message that can be used for inspecting the objects or variables while debugging the application with in Visual Studio.

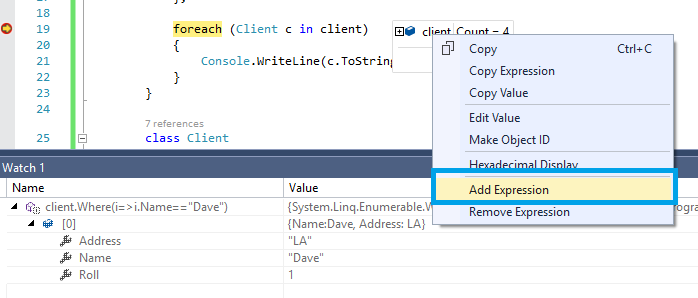
Along with Watch Window and Immediate Window, DataTips also support debugging of Lambda Expression.

**Step 1 :** Add a DataTips during Debugging

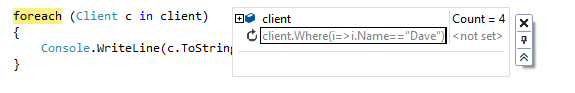


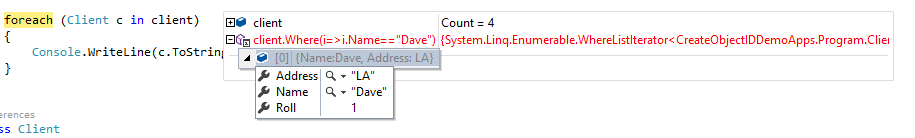


**Step 2:**  Right Click on the DataTips and select “Add Expression” from the context menu.



**Step 3 :** Run and Execute Lambda Expression.





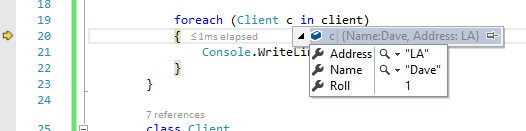
track Out of Scope Objects by assigning an Object ID during debugging

When we set “**Make Object ID**” for an object,  we are informing the Debugger to keep track of that object no matter it’s within scope or out of scope for the current context.

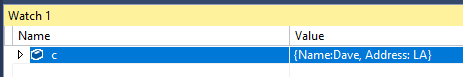
You can create the “Object ID” either from Locals, Autos or from Watch Windows. Object ID is a integer number prefix by a dollar ($) sign. When we create Object ID for an particular object, the Debugger uses an integer value to uniquely identify the object.

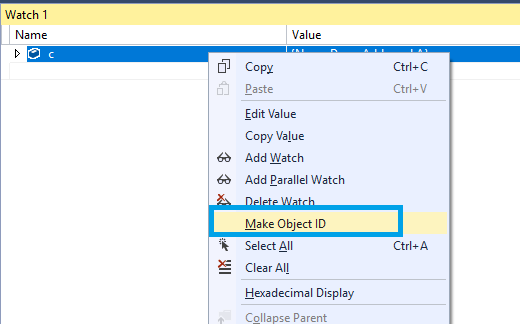
This “**Object ID**” allows you to get the object details even if it is out of scope.

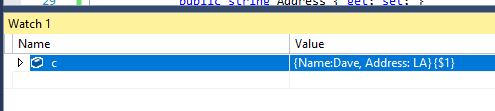
Let’s try to understand with a very simple example, where we are iterating a set of string through a loop. If you inspect the value of “item” for first time it should be “name1”.



Now If you move to next items, there is no way to get the value of previous items. In that case you can just select “Make Object ID” for the selected object.

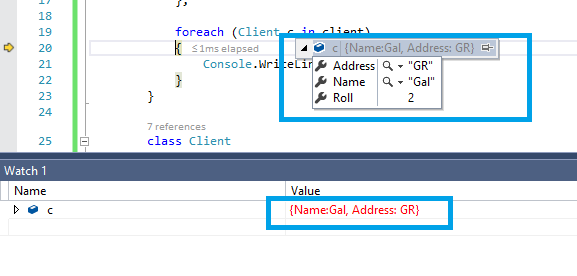




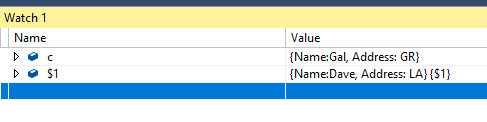


Object the Object ID is created, a $ < Value> will be added with the item.

Now, Let’s think you move to the next item in the collection.



Now, to access the previous value you can just type “$1” and it will show the value of previously tagged item associated with this object id.



You can access one object’s values which is already went out of scope. **This is super powerful feature during debugging**; You can even object / values across methods / modules.

Think about this following add-on. Your context moved to anther method, and there if you try to access “item” values; you will get message that “item” does not exist.

But, instead, when you are trying to access it with object it you have it live.

