(T33)討論DebuggingTools  
CourseGUID: e48417fc-9db5-4e99-822c-706c5ccef6cc  
=======================================================================  
(T33)討論DebuggingTools  
=======================================================================  
1. Debugging tool bar

2. Debugging Window

3. Breakpoints

-----------

4. Debug scenario 1

4.1. Debug scenario 1 - Start Debugging : Alt + F5

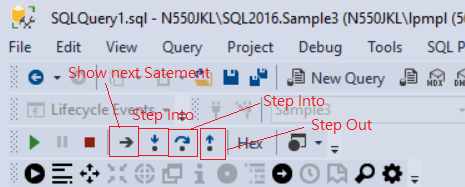
4.2. Debug scenario 1 - Show Next Statement : Alt + Num \*

4.3. Debug scenario 1 - Step Over(F10) , Step Into(F11), Call Stack Window, Locals Window, breakpoints(F9), conditional Breakpoints, Continue (Alt+F5), Step Out(Shift+F11)

-----------

5. Debug scenario 2 - Run to Cursor(Ctrl+F10), Step Over(F11)  
=======================================================================

1. Debugging tool bar



1.

The Debugging tool bar contains the most popular tools for debugging.

1.1.

**Continue (Alt + F5)**

or called

**Start Debugging :** **Alt + F5**



1.2.

**Stop Deugging(Shift + F5)**



1.3.

**Show Next Statement (Alt+Num \*)**

shows the next statement that the debugger is about to execute

Graphical user interface, application

Description automatically generated

1.4.

**Step Into(F11)**

Step into function, store procedure ...ect

in order to debug  function, store procedure ...ect

Graphical user interface, application

Description automatically generated

1.5.

**Step Over (F10)**

If you don't want to step into function, store procedure ...ect

You may use **Step Over(F10)** to step over to next line of   function, store procedure ...ect

Graphical user interface, application

Description automatically generated

1.6.

**Step Out(Shift + F11)**

If you are in the middle of function, store procedure ...ect

You know there is no problem in the function, store procedure ...ect

you want to get out from function, store procedure ...ect

You may use **Step Out(Shift + F11)**

Graphical user interface, application

Description automatically generated

1.7.

**Run to Cursor(Ctrl+F10)**

**Run to Cursor(Ctrl+F10)** command executes all the statements in a batch up to the current cursor position

At the moment, the next statement little yellow arrow is pointing

PRINT @Count;

I am in the middle of the while loop

I don't think loop has any problem

you want to get out from the while loop

but stay in side of store procedure

In this case, we may use "**Run to cursor**"

Now, move your mouse and point to

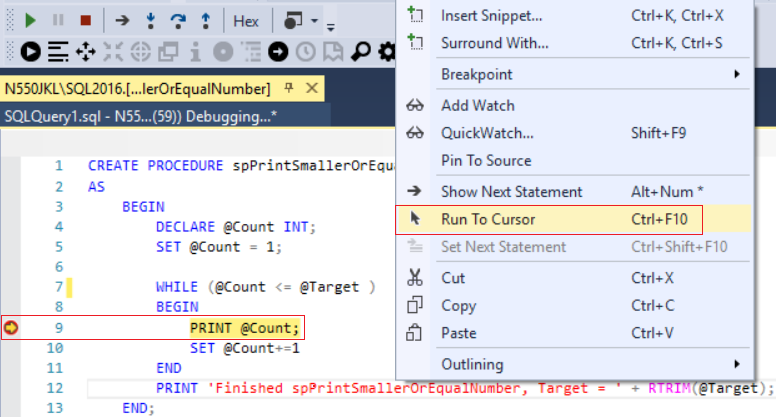
PRINT 'Finished spPrintSmallerOrEqualNumber, Target = ' + RTRIM(@Target);

**Right click --> Run to the Cursor**

After you use "**Run to Cursor**"

The next statement is pointing to

PRINT 'Finished spPrintSmallerOrEqualNumber, Target = ' + RTRIM(@Target);



=====================================================

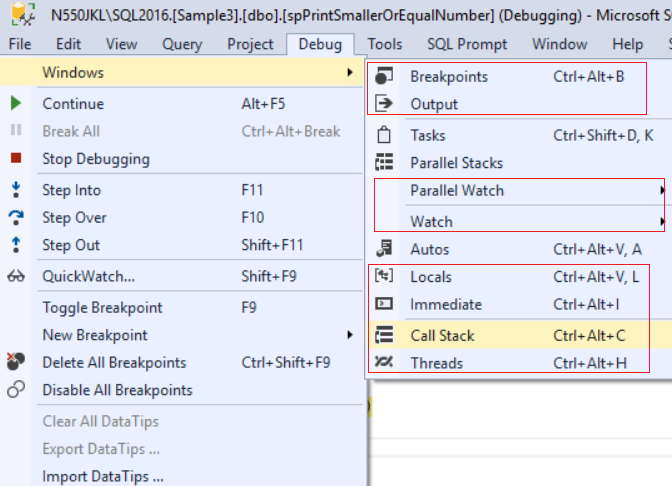
2. Debugging Window

**2.**

**Windows**

If you can not see any debug window, you may have to add from tool bars.

**Debug --> Windows --> ....**



**2.1. Call Stack Window**

**Now, I have stepped into** **the Store Procedure**

The next statement yellow arrow is currently pointing

SET @Count = 1;

**Call Stack Window** in SSMS

allow user to navigate call stack to view each local variable value in different stack level.

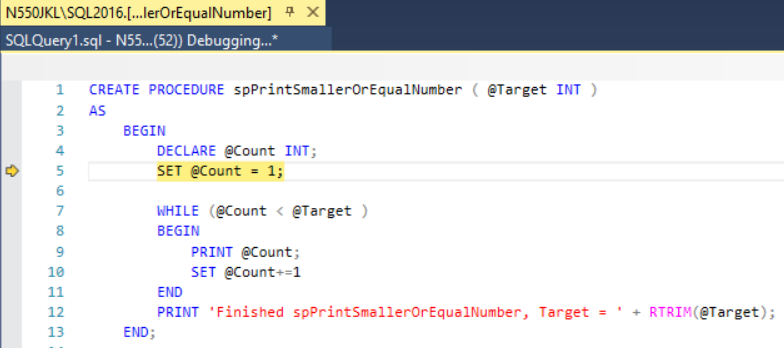
Stack means **First in Last out**, thus in this case,

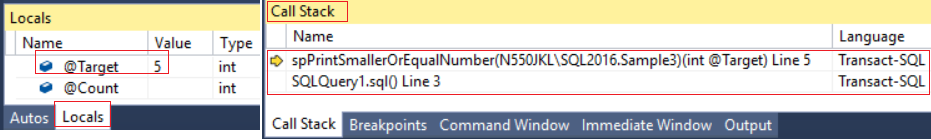
we know

**SQLQuery1.sql() Line 3**      invoke

**spPrintSmallerOrEqualNumber(N550JKL\SQL2016.Sample3)(int @Target) Line 5**

Thus,   **SQLQuery1.sql() Line 3**    is    **First in Last out**





-------------------------------------------------------------------------------------------------------

**2.2. Locals Window**

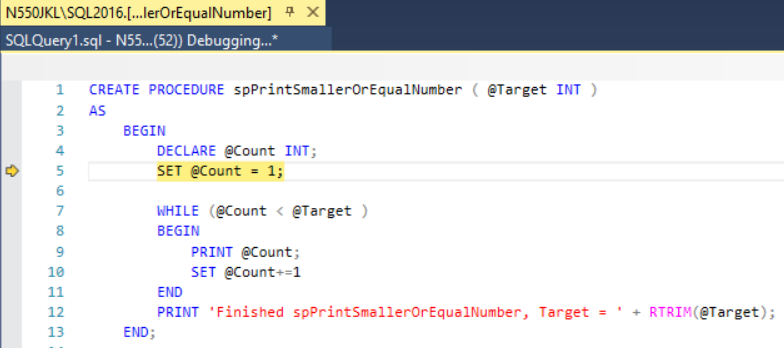
**Locals Window in SSMS** displays the current **variables values** and **parameters**.

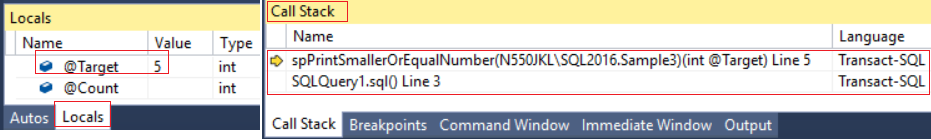
Because

SET @Count = 1;

is next statement, that means we have not set @Count=1 at that moment yet.

Thus, we can see @Count has no value in the Locals Window





-------------------------------------------------------------------------------------------------------

**2.3. Immediate Window**

**Locals Window in SSMS** displays the current **variables values** and **parameters**.

At this moment,

@Target=5

@Count=5

**Immediate Window** in SSMS

allow user to print variable value with expression

and then get the value immediately.

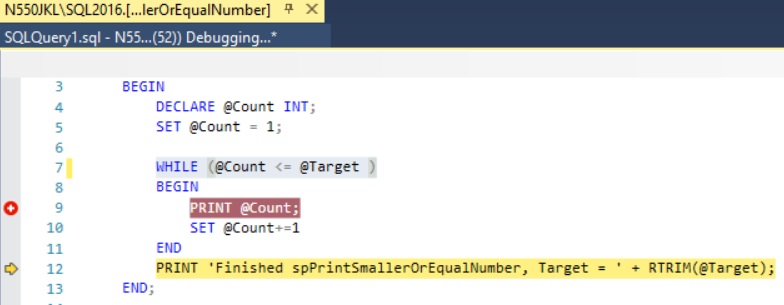
E.g.

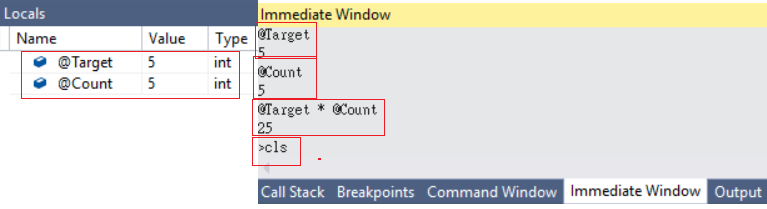
Enter @Count, it will return 5

Enter @Target, it will return 5

Enter (@Target \* @Count), it i will return 25

If you enter **>cls** ,  it will clear the screen





-------------------------------------------------------------------------------------------------------

**2.4. Watch Window**

**Watch Window in SSMS** :

Watch the Value of variable, and it is similar to **Locals Window**.

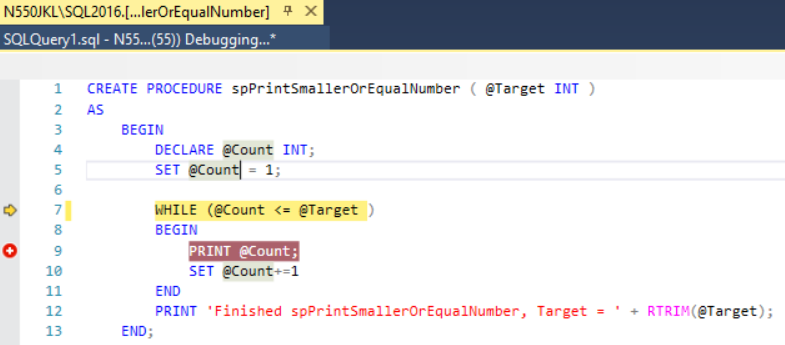
At the moment, the next statement little yellow arrow is pointing to

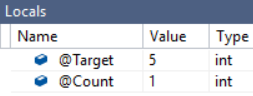
WHILE (@Count <= @Target )

At the moment, see the **Locals Window**

@Target = 5

@Count = 1



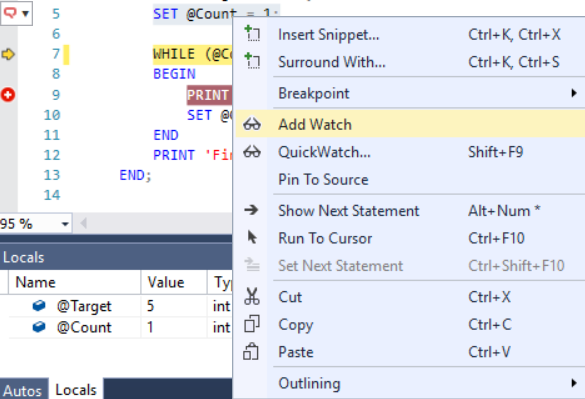


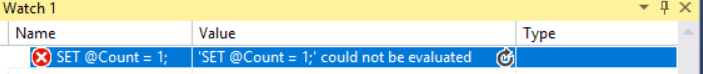
Move your mouse to

SET @Count = 1;

and mouse point to **@Count --> Right click --> Add watch**

Then you will **add** the **@Count** variable to **Watch Window**





=====================================================

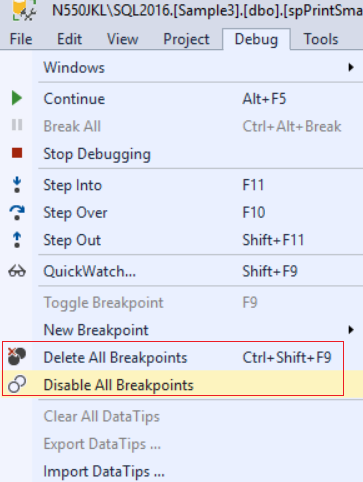
3. Breakpoints

**3. Breakpoints**

**3.1. Breakpoints (F9)**

**3.2. Debug --> Enable/Disable all** **Breakpoints**Or

**Delete all break points**

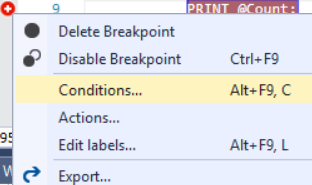


**3.3.**

**Conditional Breakpoint**

Breakpoint --> Right click --> Conditions...

Conditional Breakpoints are hit only when the condition is met.



=====================================================

4. Debug scenario 1

Open 2 query window.

Copy the query code from T033\_01 to Query Window 1.

Run T033\_01 to build spPrintSmallerOrEqualNumber in Query Window 1.

Once you finised, close the Query Window 1.

--======================================================================

--T033\_01

--spPrintSmallerOrEqualNumber

--Drop Store Procedure exists then DROP it

IF ( EXISTS ( SELECT    \*

              FROM      INFORMATION\_SCHEMA.ROUTINES

              WHERE     ROUTINE\_TYPE = 'PROCEDURE'

                        AND LEFT(ROUTINE\_NAME, 3) NOT IN ( 'sp\_', 'xp\_', 'ms\_' )

                        AND SPECIFIC\_NAME = 'spPrintSmallerOrEqualNumber' ) )

    BEGIN

        DROP PROCEDURE spPrintSmallerOrEqualNumber;

    END;

GO -- Run the previous command and begins new batch

CREATE PROCEDURE spPrintSmallerOrEqualNumber ( @Target INT )

AS

    BEGIN

        DECLARE @Count INT;

        SET @Count = 1;

             --\*\* Logic error here

             WHILE (@Count < @Target )

             BEGIN

                    PRINT @Count;

                    SET @Count+=1

        END

        PRINT 'Finished spPrintSmallerOrEqualNumber, Target = ' + RTRIM(@Target);

    END;

GO -- Run the previous command and begins new batch

Now, In the Query Window 2,

Copy the query code from T033\_02 to Query Window 2.

we will run T033\_02

--======================================================================

--T033\_02

--Debug spPrintSmallerOrEqualNumber

DECLARE @Target INT

SET @Target = 5

EXECUTE spPrintSmallerOrEqualNumber @Target

Print 'Finished'

In the Query Window 2,

When we run T033\_02.

spPrintSmallerOrEqualNumber should print all the integers

which is less than or equal to the input target value.

E.g.

When target==5

it suppose to print as following.

--1

--2

--3

--4

--5

--Finished spPrintSmallerOrEqualNumber, Target = 5

However, we make mistake here

**--WHILE (@Count < @Target )**

Thus, it actually print as following

--1

--2

--3

--4

--Finished spPrintSmallerOrEqualNumber, Target = 5

We will go through some debug processes with debug tools.

Then we will find out the correct code should be ..

--WHILE (@Count <= @Target )

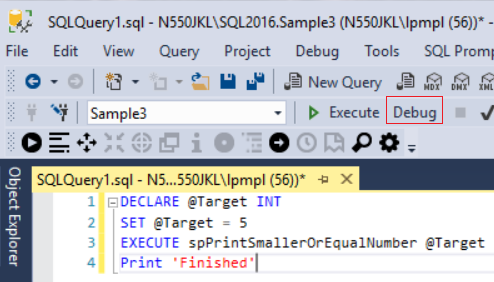
Let's go through some debug processes with debug tools from now.

4.1. Debug scenario 1 - Start Debugging : Alt + F5

There are three way to start debugging

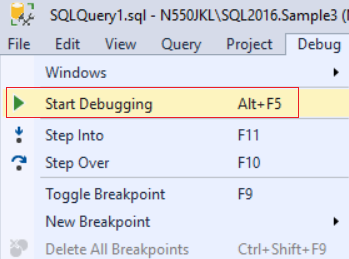
A.

Press the "Debug" button on the tool bars to start debugging.



B.

Debug --> Start Debugging



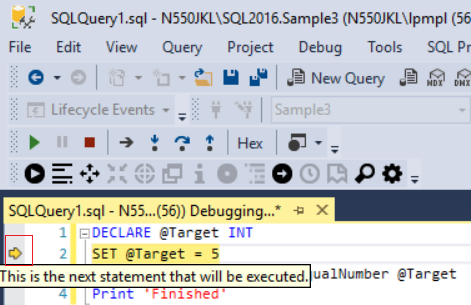
C.

Press **Alt+F5**

4.2. Debug scenario 1 - Show Next Statement : Alt + Num \*

Once you press "Debug" button, then you will see a little yellow arrow in the right hand side.

This little yellow arrow is the next statement that the debugger is about to execute.



During the debugging process,

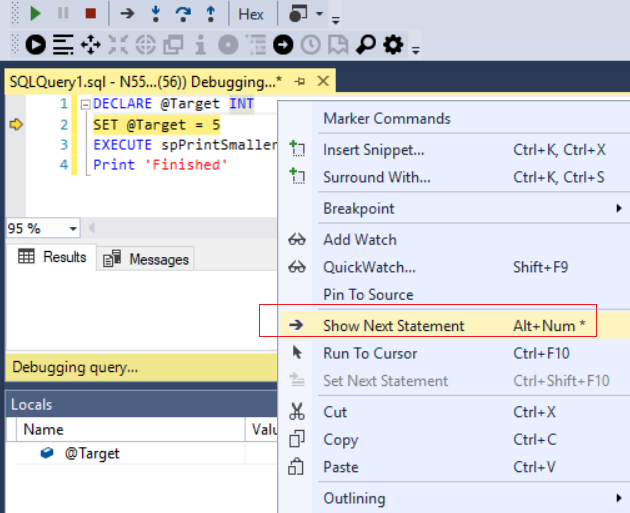
you might have to open a lot of windows and get lost.

You want to find out the position which the current statement is executing.

**any where in the query window --> Show Next Statement**

**Show Next Statement**

shows the next statement that the debugger is about to execute.

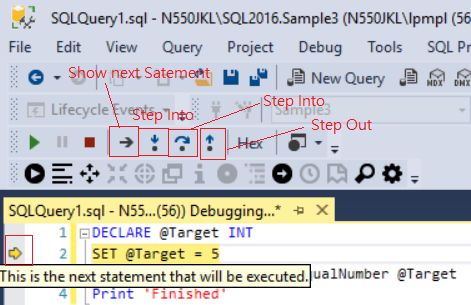


4.3. Debug scenario 1 - Step Over(F10) , Step Into(F11), Call Stack Window, Locals Window, breakpoints(F9), conditional Breakpoints, Continue (Alt+F5), Step Out(Shift+F11)

Once you press "Debug" button, then you will see a little yellow arrow in the right hand side.

This little yellow arrow is the next statement that the debugger is about to execute.

Now, you may press **(Step Over :** **F10)**  or **(Step Into :** **F11)**



-------------------------------------------------------------------------------------------------

Now, you may press **(Step Over :** **F10)**  or **(Step Into :** **F11)**

If you press **(Step Over :** **F10)**

The next statement will point to

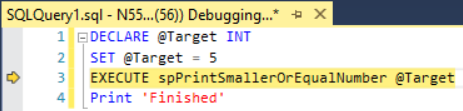
Print 'Finished'

Otherwise, if you press **(Step Into :** **F11)**

The next statement will step into that Store Procedure, **spPrintSmallerOrEqualNumber**

In this case, I would like to press **(Step Into :** **F11)**

Because I want to debug the Store Procedure, **spPrintSmallerOrEqualNumber**

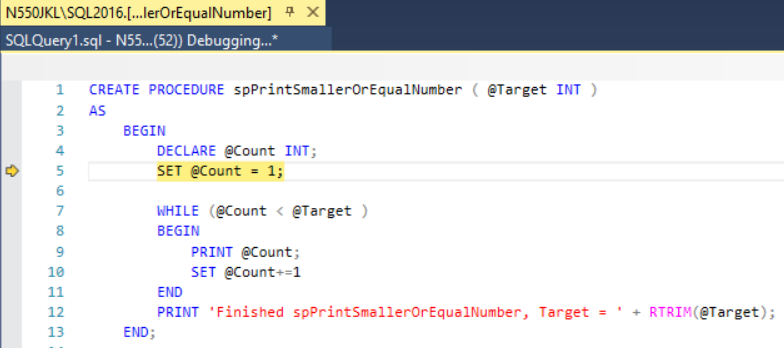


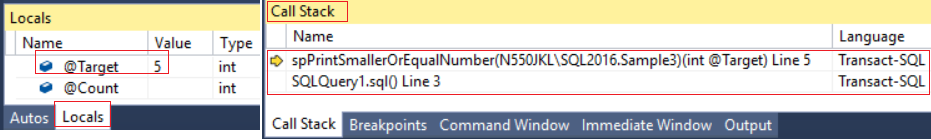
-------------------------------------------------------------------------------------------------

Now, I have **stepped into** the Store Procedure, **spPrintSmallerOrEqualNumber**

The next statement yellow arrow is currently pointing

SET @Count = 1;





Call Stack Window

**Call Stack Window** in SSMS

allow user to navigate call stack to view each local variable value in different stack level.

Stack means **First in Last out**, thus in this case,

we know

**SQLQuery1.sql() Line 3**      invoke

**spPrintSmallerOrEqualNumber(N550JKL\SQL2016.Sample3)(int @Target) Line 5**

Thus,   **SQLQuery1.sql() Line 3**    is    **First in Last out**

Locals Window

**Locals Window in SSMS** displays the current **variables values** and **parameters**.

Because

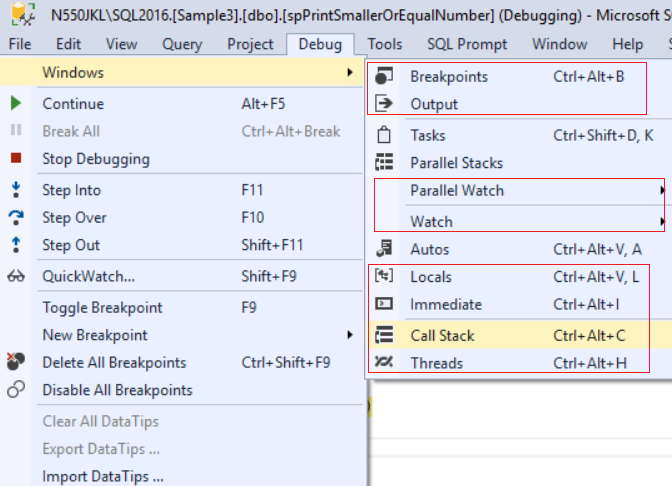
SET @Count = 1;

is next statement, that means we have not set @Count=1 at that moment yet.

Thus, we can see @Count has no value in the Locals Window

If you can not see any debug window, you may have to add from tool bars.

**Debug --> Windows --> ....**



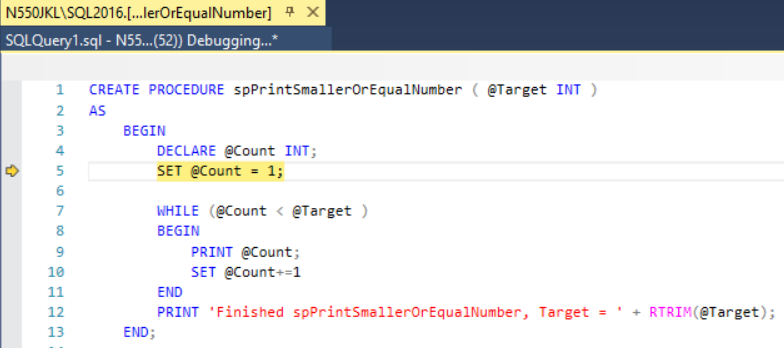
----------------------------------------

During this Store Procedure, **spPrintSmallerOrEqualNumber**

There is no different you press **(Step Over :** **F10)**  or **(Step Into :** **F11)**

Because I would like to debug the Store Procedure, **spPrintSmallerOrEqualNumber**

I prefer to press  **(Step Over :** **F10)** here



-------------------------------------------------------------------------------------------------

Now, the next statement point to

WHILE (@Count < @Target )

I know there is something wrong in

PRINT @Count;

I don't want to waste time to press **(Step Over :** **F10)**  or **(Step Into :** **F11)**

I want to go straight the problem point.

Thus, I need a **Breakpoint**

There are 2 ways to set up **Breakpoint**

A.

left click to the gray margin on the left hand side to add breakpoint in SSMS.

You may click the existent breakpoint to remove it.

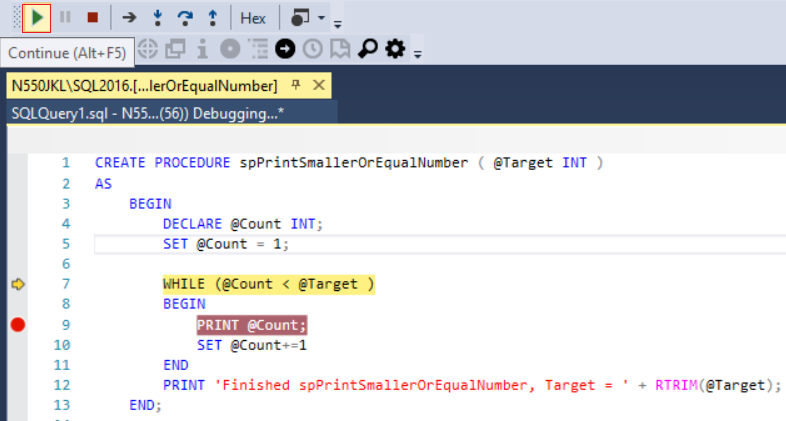
B.

Move cursor to the line you want to set up breakpoint, then press **F9**

Now, we set the breakpoint at

PRINT @Count;

Now Press **Continue (Alt+F5)**



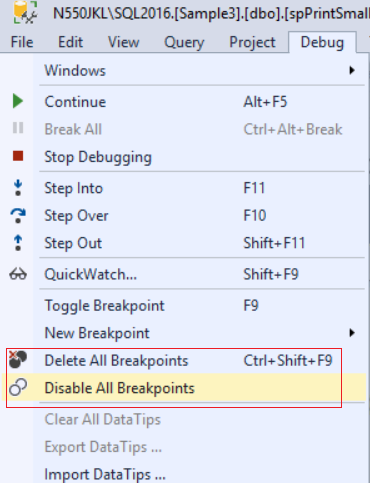
If you want to **Enable/Disable All Breakpoints**, or **delete all breakpoints**.

**Debug --> Delete All Breakpoints**

**Debug --> Enable/Disable All Breakpoints**

In current case, we still need that breakpoint,

please don't delete that breakpoint.



-------------------------------------------------------------------------------------------------

After Press Continue **(Alt+F5)**

It will hit the breakpoint in the while loop at first time.

At the moment, from the **locals window**,

we can see **@Target=5, @Cont=1**.

Let's press Continue **(Alt+F5) again**

to hit the breakpoint in the while loop at 2nd time.

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

-------------------------------------------------------------------------------------------------

After Press Continue **(Alt+F5)**

It will hit the breakpoint in the while loop at 2nd time.

At the moment, from the **locals window**,

we can see **@Target=5, @Cont=2**.

Graphical user interface, text, application, email

Description automatically generated

Table

Description automatically generated

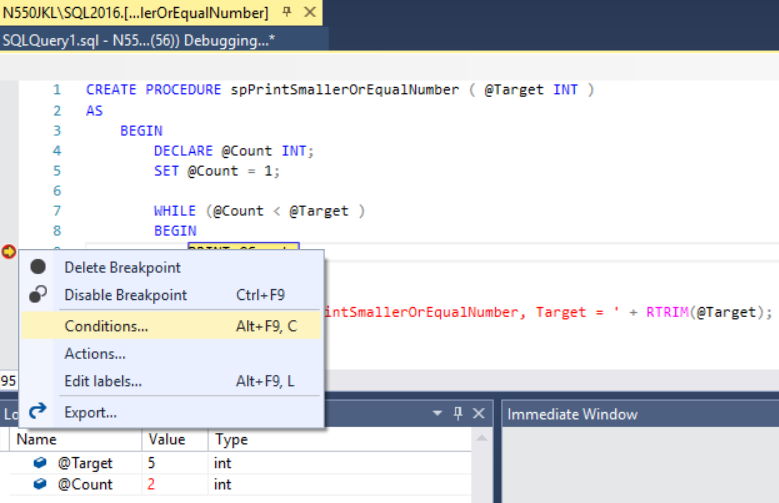
I don't want to waste time to keep pressing continue,

thus, I want to use **conditional Breakpoints**

**Mouse point to breakpoint --> Right clock --> conditions**

--> Type the condition.

In this case, condition is**@Count=@Target-1**



-->

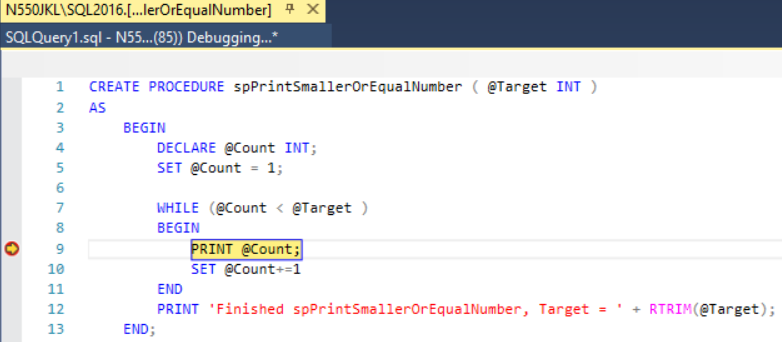
Graphical user interface, text, application

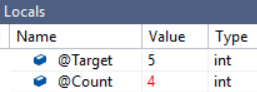
Description automatically generated

Now, Press **Continue (Alt+F5)**

-->

The while loop will jump to the breakpoint when **@Count=@Target-1**





Then I keep pressing **(Step Over :** **F10)**

until the next statement point to

PRINT 'Finished spPrintSmallerOrEqualNumber, Target = ' + RTRIM(@Target);

-->

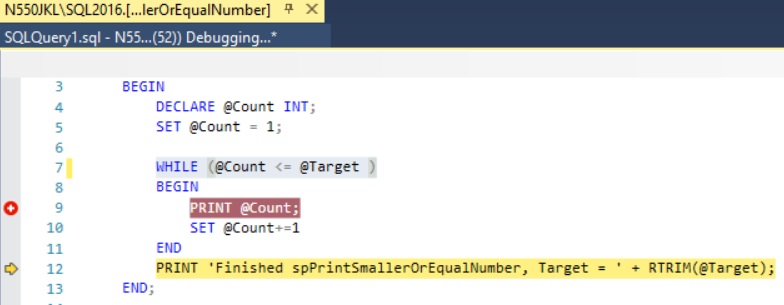
Thus, we know that

WHILE (@Count < @Target )

is logic error, but

WHILE (@Count <= @Target )

is correct answer



Graphical user interface, application

Description automatically generated with medium confidence

Locals Window

**Locals Window in SSMS** displays the current **variables values** and **parameters**.

At this moment,

@Target=5

@Count=5

Immediate Window

**Immediate Window** in SSMS

allow user to print variable value with expression

and then get the value immediately.

E.g.

Enter @Count, it will return 5

Enter @Target, it will return 5

Enter (@Target \* @Count), it i will return 25

If you enter **>cls** ,  it will clear the screen

5. Debug scenario 2 - Run to Cursor(Ctrl+F10), Step Over(F11)

I assume you Debug the main query again, then press "**Continue (Alt+F5)**"

and we hit the "**Condition Breakpoint again**"

The while loop will jump to the breakpoint when **@Count=@Target-1**

We have already fix the while loop

by replacing

WHILE (@Count <= @Target )

to

WHILE (@Count <= @Target )

At the moment, the next statement little yellow arrow is pointing

PRINT @Count;

I am in the middle of the while loop

I don't think loop has any problem

you want to get out from the while loop

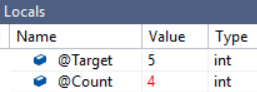
but stay in side of store procedure, **spPrintSmallerOrEqualNumber**

In this case, we may use "**Run to cursor**"

**Run to Cursor** command executes all the statements in a batch up to the current cursor position

Graphical user interface, text, application

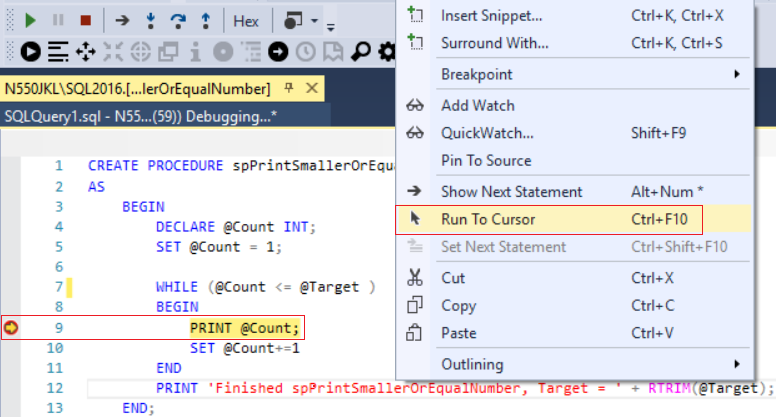
Description automatically generated

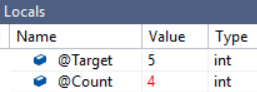


Now, move your mouse and point to

PRINT 'Finished spPrintSmallerOrEqualNumber, Target = ' + RTRIM(@Target);

**Right click --> Run to the Cursor**





After you use "**Run to Cursor**"

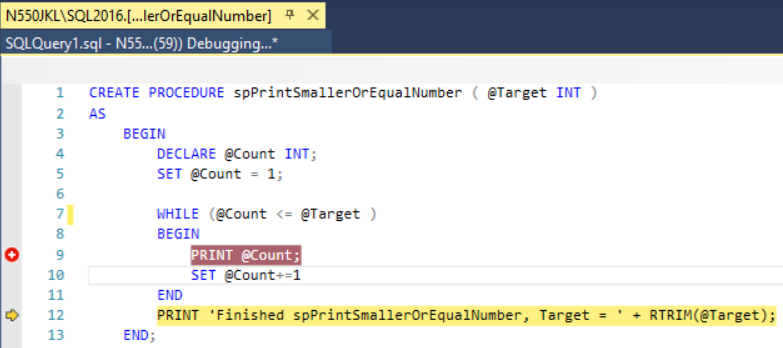
The next statement is pointing to

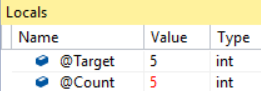
PRINT 'Finished spPrintSmallerOrEqualNumber, Target = ' + RTRIM(@Target);

You may see the @Target=5, and **@Cout=5**

"**Run to Cursor**" can be replacing by using breakpoint(F9) and Continue(Alt+F5)'

I personally get used to use breakpoint(F9) and Continue(Alt+F5)'.





--------------------------------------------------------------------------------------------------------------------

Now, the next statement is pointing to

PRINT 'Finished spPrintSmallerOrEqualNumber, Target = ' + RTRIM(@Target);

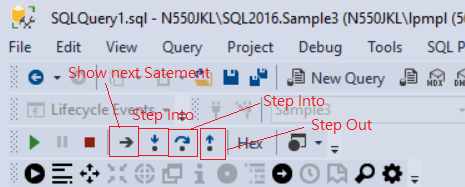
I know this store procedure has no more problem.

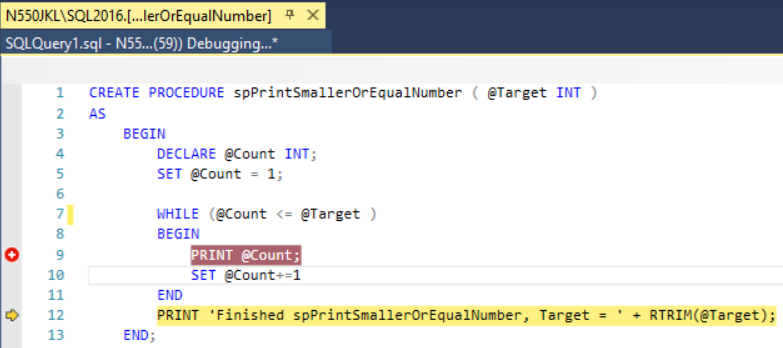
I want to get out from this store procedure.

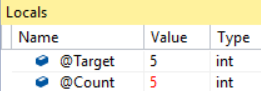
Thus, I can use **Step Out(Shift+F11)**

to get out from this store procedure.

Now press **Step Out(Shift+F11)**







After I press **Step Out(Shift+F11)**

to get out from the store procedure.

The next statement is pointing

Print 'Finished'

in the main query window.

Now, you may stop the debugging by pressing "**Stop Debugging**"



