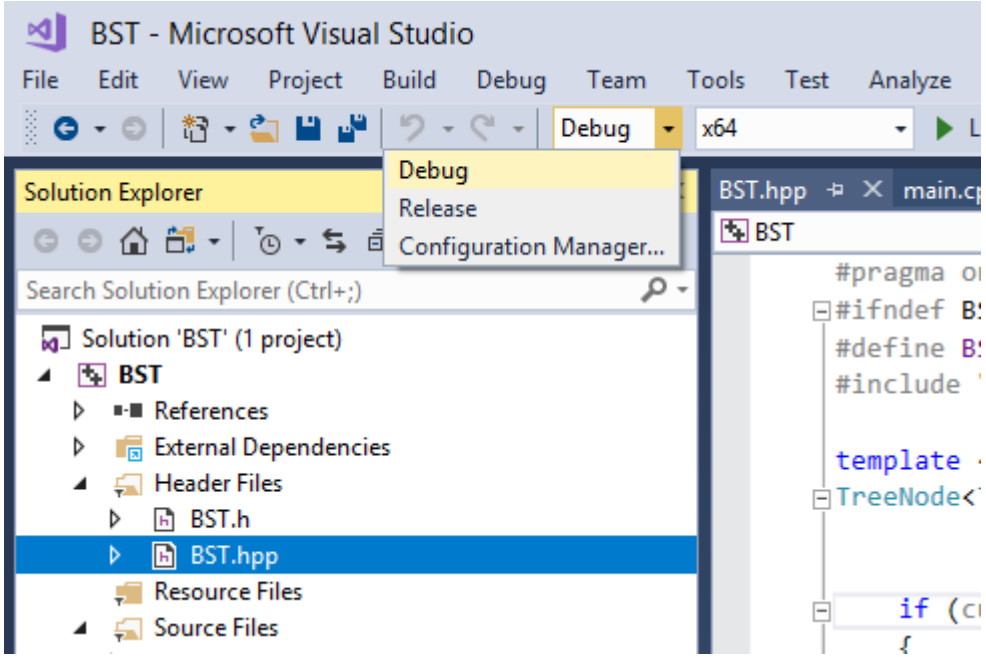
Basic Debugging with MSVC

Before Debugging

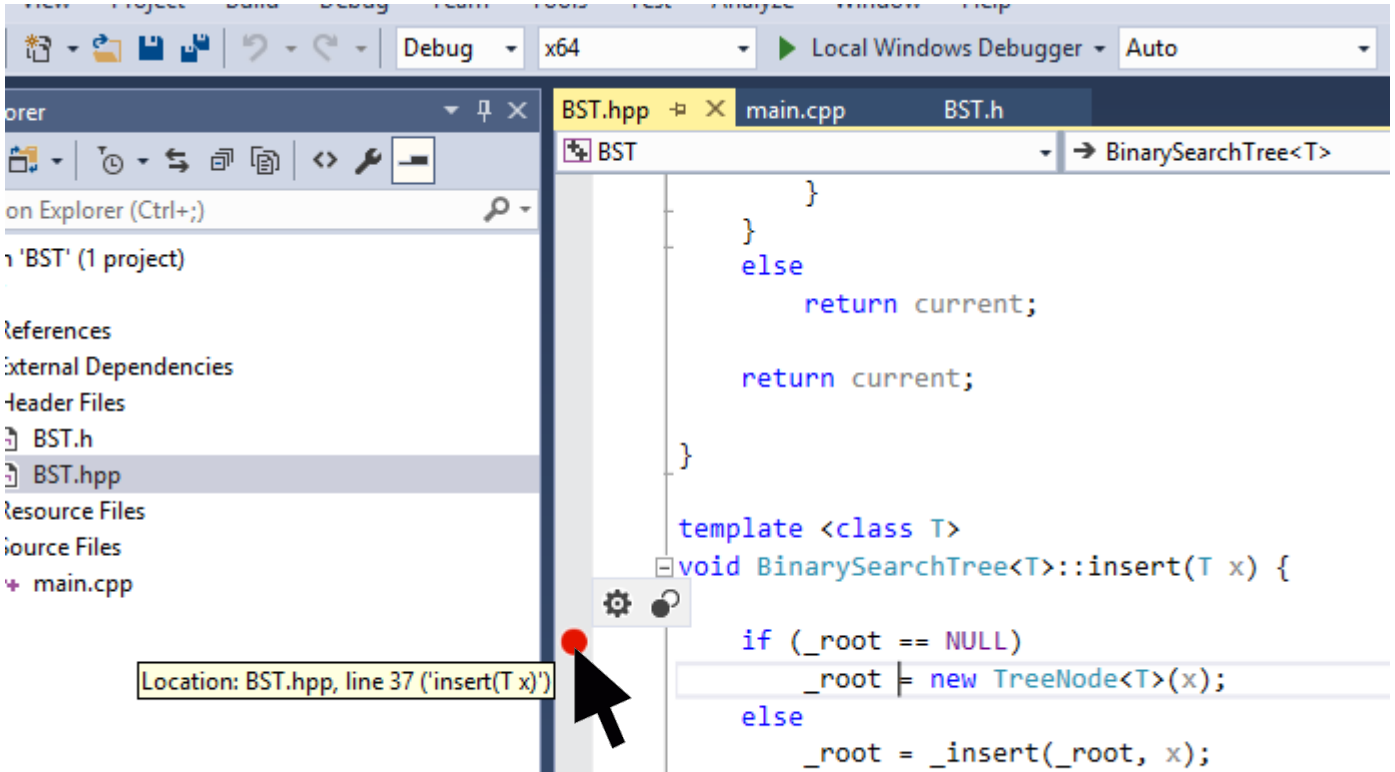
- Make sure that you compile in “Debug” profile



- Release profile will give you a faster and optimized version of executables

Setting Breakpoints

- If you wish to pause at any line of your code, simply click a red dot on the left margin of your code



Then Start Debugging

- By either:
  - pressing “F5”
  - or click “Local Windows Debugger”

Debugger

You paused at this line

BST.hpp | main.cpp | BST.h

BST → BinarySearchTree<T> → insert(T x)

```
}

template <class T>
void BinarySearchTree<T>::insert(T x) {

    if (_root == NULL)
        _root = new TreeNode<T>(x);
    else
        _root = _insert(_root, x);

}
```

110 %

Diagnostic Tools

Diagnostics session: 0 seconds (44 m...

Your call stack so far.

- main() → testInsertion1() → insert()

Summary | Events | Memory Usage

Events

Show Events (1 of 1)

Memory Usage

Autos

Name	Value	Type
_root	0x0000000000000000 <NULL>	TreeNod
_item	<Unable to read memory>	
_left	<Unable to read memory>	
_right	<Unable to read memory>	
_height	<Unable to read memory>	
this	0x0000008eaa4df978 {_size=0 _root=0	BinarySe
_size	0	int
_root	0x0000000000000000 <NULL>	TreeNod
x	7	int

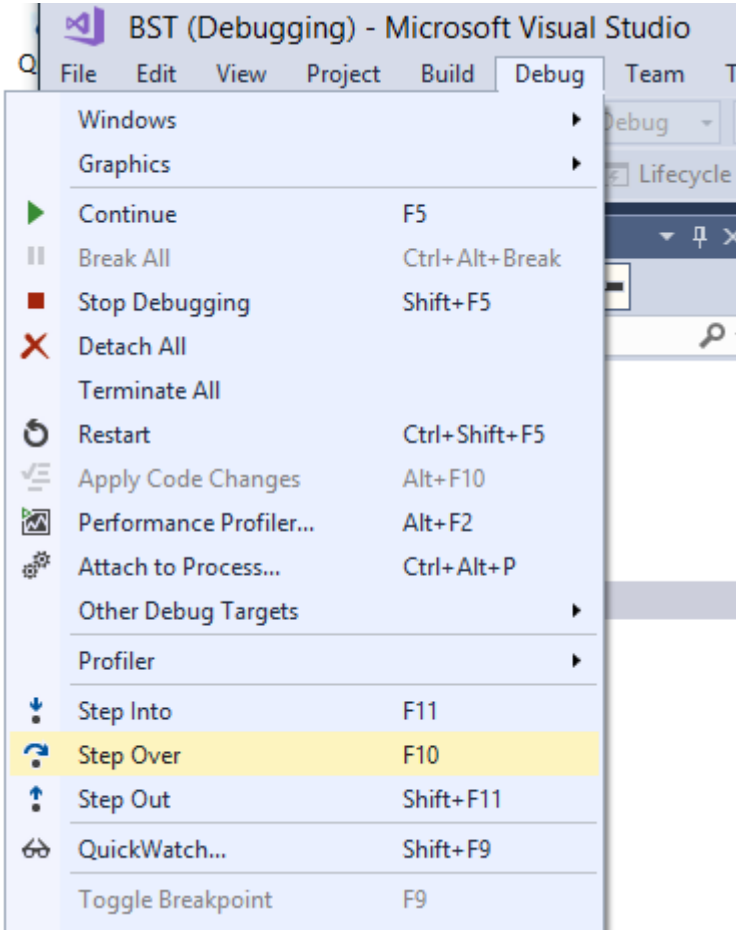
Call Stack

Name	Lang
BST.exe!BinarySearchTree<int>::insert(int x) Line 37	C++
BST.exe!testInsertion1(bool printWithHeight) Line 27	C++
BST.exe!main() Line 19	C++
[External Code]	

All the variables at this scope

To Continue

- You can go to the “next step” by
  - Step Over (F10)
  - Step Into (F11)
- Or simply
  - Continue (F5)
  - Run until the next breakpoint



Let’s Try “Step Over”

BST.hpp | main.cpp | BST.h

BST | BinarySearchTree<T> | insert(T x)

Press “F10” once
Go to the next line

```
}

template <class T>
void BinarySearchTree<T>::insert(T x) {

    if (_root == NULL)
        _root = new TreeNode<T>(x);   ≤ 1ms elapsed
    else
        _root = _insert(_root, x);

}
```

110 %

Autos

Name	Value	Type
_root	0x0000000000000000 <NULL>	TreeNod
_item	<Unable to read memory>	
_left	<Unable to read memory>	
_right	<Unable to read memory>	
_height	<Unable to read memory>	
this	0x0000006b2e4cfb58 {_size=0 _root=(	BinarySe
_size	0	int
_root	0x0000000000000000 <NULL>	TreeNod
_item	<Unable to read memory>	
_left	<Unable to read memory>	
_righ	<Unable to read memory>	
_heig	<Unable to read memory>	
x	7	int

Call Sta

Nar
BST
BST
BST
[Ext

BST.hpp | BST | insert

Press “F10” again
Go to the next line

```
}

template <class T>
void BinarySearchTree<T>::insert(T x) {

    if (_root == NULL)
        _root = new TreeNode<T>(x);
    else   ≤ 3ms elapsed
        _root = _insert(_root, x);

}
```

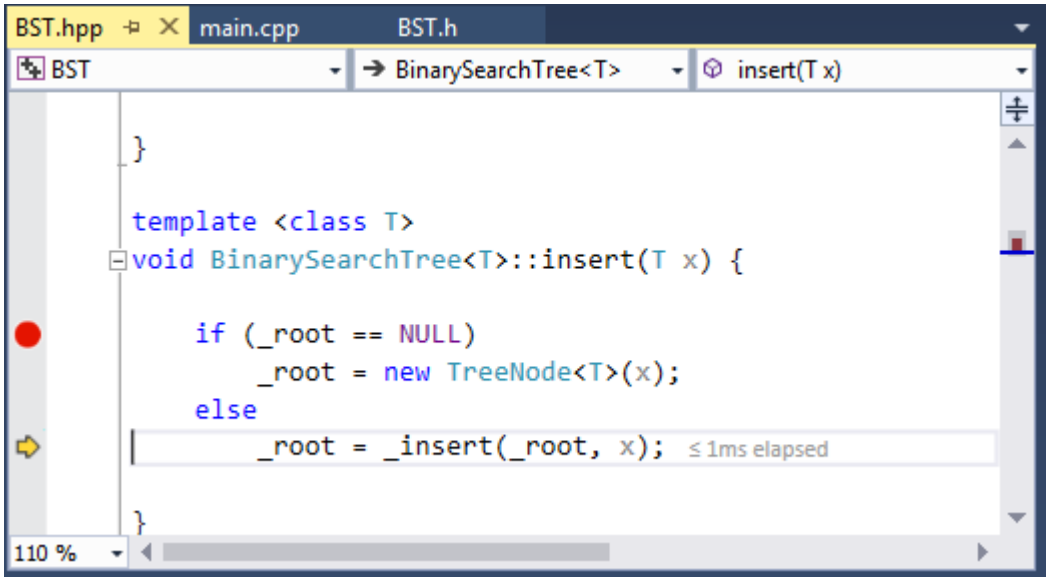
110 %

Autos

Name	Value	Type
operator ne	0x0000006b2e5388c0	void *
_root	0x0000006b2e5388c0 {_item=7 _left=(	TreeNod
_item	7	int
_left	0x0000000000000000 <NULL>	TreeNod
_right	0x0000000000000000 <NULL>	TreeNod
_height	0	int
this	0x0000006b2e4cfb58 {_size=0 _root=(	BinarySe
_size	0	int
_root	0x0000006b2e5388c0 {_item=7 _left=(	TreeNod
		int
		TreeNod
		TreeNod
		int
x		int

Notice all the changes
in the variables here
(marked red)

- At this point, you have inserted a node 7
- Press “F5” to continue to the next breakpoint
- After one “F10”, you will be here:



- Because you have inserted a node and _root is not NULL now
- Now you have a choice, “F10” or “F11”

- F11
  - Will go into the function "_insert()"

```
template <class T>
TreeNode<T>* BinarySearchTree<T>::_insert(TreeNode<T>* current, T x) {  ≤ 1ms elapsed


    if (current->_item > x)
    {
        if (current->_left)
            current->_left = _insert(current->_left, x);
        else
        {
            current->_left = new TreeNode<T>(x);
        }
    }
```

- But if you press F10 instead, it will "finish" the function _insert()

```
template <class T>
void BinarySearchTree<T>::insert(T x) {

    if (_root == NULL)
        _root = new TreeNode<T>(x);
    else
        _root = _insert(_root, x);  ≤ 1ms elapsed

}
```

After Finishing _insert()

- _left of the _root is not NULL anymore

Autos

Name	Value
BinarySearchTree<int>::_insert retur	0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e538920 _right=0x0000000000000000 ..
_root	0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e538920 {_item=3 _left=0x0000000000(
this	0x0000006b2e4cfb58 {_size=0 _root=0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e5
_size	0
_root	0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e538920 {_item=3 _left=0x0000000000(
_item	7
_left	0x0000006b2e538920 {_item=3 _left=0x0000000000000000 <NULL> _right=0x0000000000(
_right	0x0000000000000000 <NULL>
_height	0
x	3

Autos

Name	Value	Type
BinarySearchTree<int>::_insert retur	0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e538920 _right=0x0000000000000000 ..	TreeNod
_root	0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e538920 {_item=3 _left=0x0000000000(	TreeNod
this	0x0000006b2e4cfb58 {_size=0 _root=0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e5	BinarySe
_size	0	int
_root	0x0000006b2e5388c0 {_item=7 _left=0x0000006b2e538920 {_item=3 _left=0x0000000000(	TreeNod
_item	7	int
_left	0x0000006b2e538920 {_item=3 _left=0x0000000000000000 <NULL> _right=0x0000000000(	TreeNod
_item	3	int
_left	0x0000000000000000 <NULL>	TreeNod
_right	0x0000000000000000 <NULL>	TreeNod
_height	0	int

Conditional Breakpoint

- The code insert a lot of numbers

```
cout << "Insertion Test 1" << endl;
int array[] = { 7, 3, 1, 0, 2, 5, 4, 6, 11, 9, 8, 10, 13, 12, 14 };
BinarySearchTree<int> bsti;
for (int i = 0; i < 15; i++)
    bsti.insert(array[i]);
```

- If there is something wrong when I insert 13, I have to
  - F10, F11, F11, F10, F11, F10, F11, …. ?

Conditional Breakpoint

- Set a breakpoint here

```
cout << "Insertion Test 1" << endl;
int array[] = { 7, 3, 1, 0, 2, 5, 4, 6, 11, 9, 8, 10, 13, 12, 14 };
BinarySearchTree<int> bsti;
for (int i = 0; i < 15; i++)
    bsti.insert(array[i]);
```

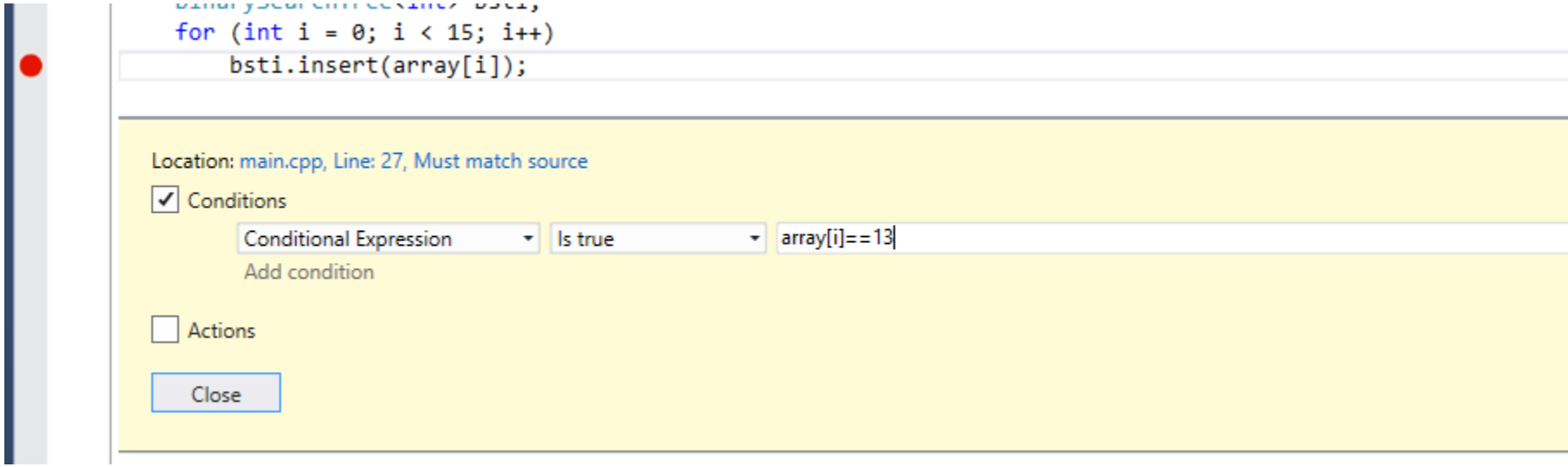
- Click “setting”

```
void testInsertion1(bool printWithHeight) {

    cout << "Insertion Test 1" << endl;
    int array[] = { 7, 3, 1, 0, 2, 5, 4, 6, 11, 9, 8, 10, 13, 12, 14 };
    BinarySearchTree<int> bsti;
    for (int i = 0; i < 15; i++)
        bsti.insert(array[i]);
```

Setting Condition to Break

- Then you can check “conditions”
  - Add “array[i]==13”
  - Then “close”



When You Run

- The program will pause at array[i] == 13
  - Unless you haven't remove the previous breakpoint

```
void testInsertion1(bool printWithHeight) {

    cout << "Insertion Test 1" << endl;
    int array[] = { 7, 3, 1, 0, 2, 5, 4, 6, 11, 9, 8, 10, 13, 12, 14 };
    BinarySearchTree<int> bsti;
    for (int i = 0; i < 15; i++)
        bsti.insert(array[i]);

    bsti.printTree(false);
```

110 %

Autos

Name	Value
bsti	{_size=0 _root=0x0000005517868b00 {_item=7 _left=0x0000005517868800 {_iten
i	12