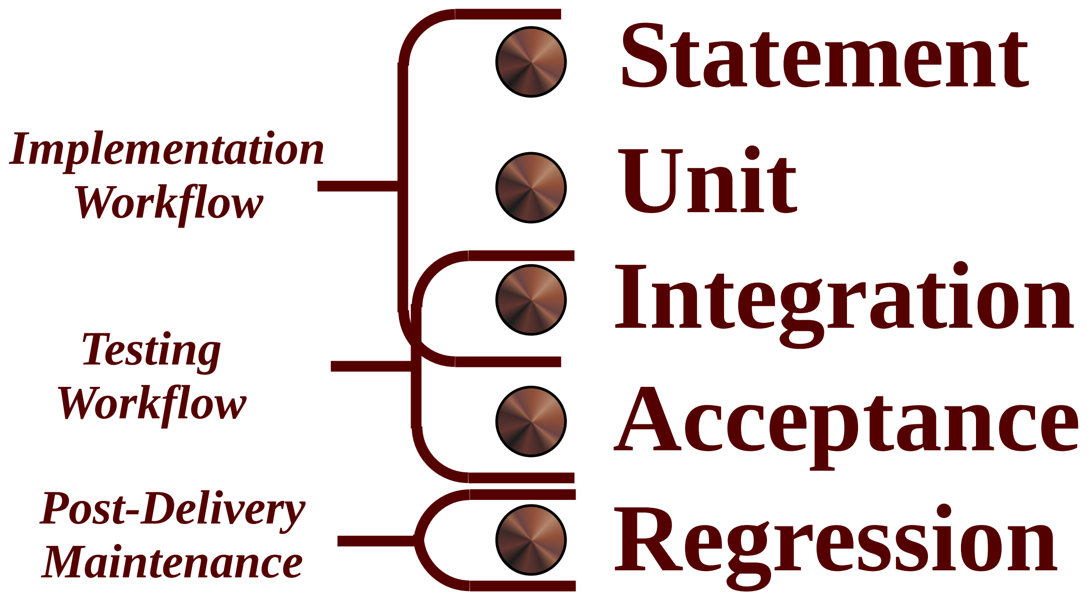
# Object Oriented Software Testing

# Object Oriented Software Testing

## Five Types of Testing



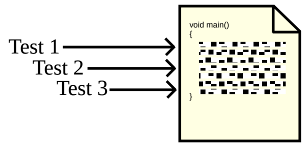
# Object Oriented Software Testing

*In the Implementation Workflow*

- **Statement**
- Unit
- Integration
- Acceptance
- Regression

*A.k.a.*

**Coverage Testing**



**Statement**

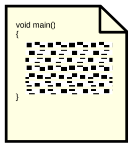
*Testing single lines or blocks of lines*

- Use drivers to pass known values in as arguments.
- Use Debugger to check validity of variables.
- Use special output statements to check states.
- Test each statement or set of statements as they are added to a function.

# Object Oriented Software Testing

*In the Implementation Workflow*

- Statement
- **Unit**
- Integration
- Acceptance
- Regression



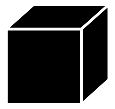
**Unit**

*Testing single functions*

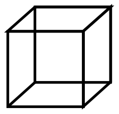
*A.k.a. or includes:*

**Method Testing**
**Path Testing**
**Boundary Value Testing**
**Black box testing**
**White/Clear box testing**

*Black box testing*



*White/Clear box testing*

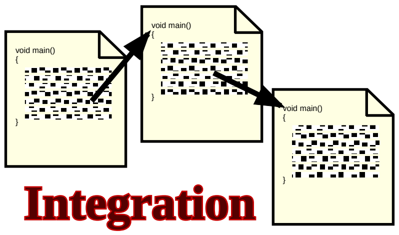


- Use drivers to pass known values in as arguments.
- Use stubs to test function calling actions.
- Use separate tests for each possible path
- Test for occurrence of fault conditions.
- Use Debugger to check validity of variables.

# Object Oriented Software Testing

*In the Implementation and Testing Workflows*

- Statement
- Unit
- **Integration**
- Acceptance
- Regression



**Integration**

*Testing how functions work together.*

*A.k.a. or includes:*

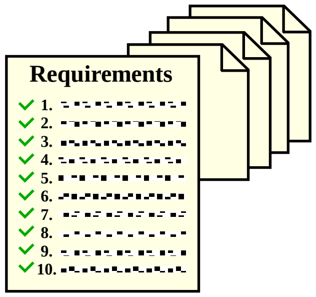
**Class Testing**
**Class Integration Testing**
**Component Testing**

- Replace all drivers and stubs with real functions.
- Use debug statements to trace call sequences.
- Use debug statements to check variable validity.
- Use the Debugger to trace path of execution.

# Object Oriented Software Testing

*In the Testing Workflow*

- Statement
- Unit
- Integration
- **Acceptance**
- Regression



**Acceptance**

*Testing against stated software requirements.*

*A.k.a. or includes:*

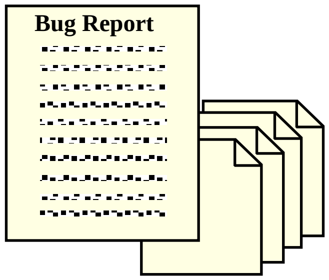
**Installation Testing**
**Stress Testing**
**User Interface Testing**
**User acceptance testing**

- Use Requirements Specification Document as guide.
- Devise tests to verify meeting all requirements:
  - External interface requirements (User, Network, etc.)
  - Functional requirements (What does it do?)
  - Performance requirements (Speed, efficiency, etc.)
  - Design constraints (Cost, time, technology available, etc.)
  - Logical database requirements (Size, access, data types, etc.)
  - Software System attributes (CM, QA, etc.)
  - Other requirements (Everything else)

# Object Oriented Software Testing

*In the Post-Delivery Maintenance Workflow*

- Statement
- Unit
- Integration
- Acceptance
- **Regression**



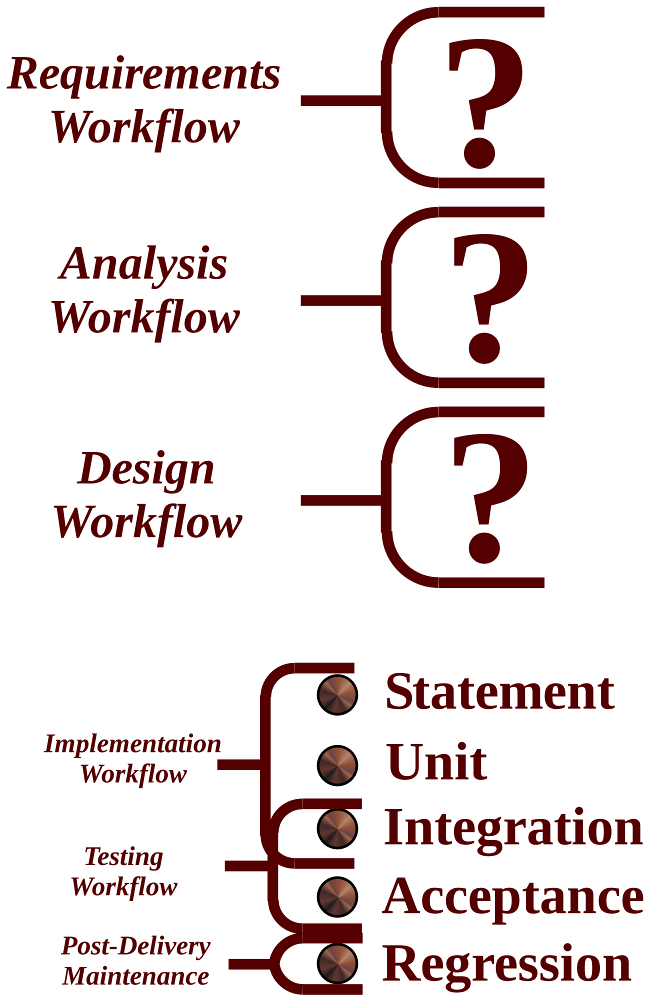
**Regression**

*Testing after fixing bugs or adding new features.*

- Perform full unit testing on new code.
- Perform full integration testing on all sections of the application affected by the new code.
- Perform acceptance testing to verify new code meets requirements to fix bug or add new feature.

# Object Oriented Software Testing

## What about the Requirements, Analysis, and Design workflows?



# Object Oriented Software Testing

## Types of testing in the Requirements, Analysis, and Design workflows?

Model reviews.
Usage scenario testing.
Prototype walk-throughs.
Prove it with code.
Peer reviews
Technical reviews.
Design review.

# Object Oriented Software Testing

## Other types of testing in the Implementation, Testing, and Post-delivery Maintenance workflows

Code Peer Reviews.
System Testing.
Alpha testing.
Beta testing.
User Acceptance Testing.

# Object Oriented Software Testing

## *In the Requirements Workflow*

### *What are you doing?*

*Understand the application domain.*
*Meet with the customer to discuss needs.*
*Define the constraints.*
*Write the Requirements Definition Document.*
*Prepare a rough draft of the Software Test Plan.*

### *What and how are you testing?*

**Model reviews.**
**Usage scenario testing.**
**Prototype walk-throughs.**
**Prove it with code.**

# Object Oriented Software Testing

*In the Analysis Workflow*

*What are you doing?*

*List requirements.*
*Determine deliverables.*
*List major milestones.*
*Determine the budget.*
*Prepare an Architectural Design.*
*Write the Software Development Plan.*
*Write the Requirements Specification Document.*
*Prepare a preliminary set of UML diagrams.*
*Revise the Software Test Plan.*

*What and how are you testing?*

**Model reviews.**
**Usage scenario testing**
**Prototype walk-throughs.**
**Prove it with code.**
**Technical reviews.**
**Peer reviews**

# Object Oriented Software Testing

*In the Design Workflow*

*What are you doing?*

*Revise the Architectural Design.*
*Plan the Detailed Design.*
*Keep meticulous records.*
*Write the Software Design Plan*
*Finish the Software Test Plan.*

*What and how are you testing?*

**Model reviews.**
**Prototype walkthroughs.**
**Prove it with code.**
**Technical review.**
**Design review.**

# Object Oriented Software Testing

## *In the Implementation Workflow*

### *What are you doing?*

*Assign modules to team members.*
*Write code.*
*Integrate code from all team members.*
*Revise code as required while testing.*

### *What and how are you testing?*

**Statement testing.**
**Unit testing.**
**Integration testing.**
**Code Peer Reviews.**

# Object Oriented Software Testing

***In the Testing Workflow***

***What are you doing?***

***Perform all final testing of the product.***

***What and how are you testing?***

**Integration Testing.**
**Acceptance Testing.**
**System Testing.**
**Alpha testing.**
**Beta testing.**
**User Acceptance Testing.**

# Object Oriented Software Testing

## *In the Post-Delivery Maintenance Workflow*

***What are you doing?***

***Fix bugs.***
***Add new features.***
***Document changes.***

***What and how are you testing?***

**Regression testing.**

# Object Oriented Software Testing

***Some***

***testing suggestions***

***for programming assignments***

***during the Implementation Workflow!***

# Object Oriented Software Testing

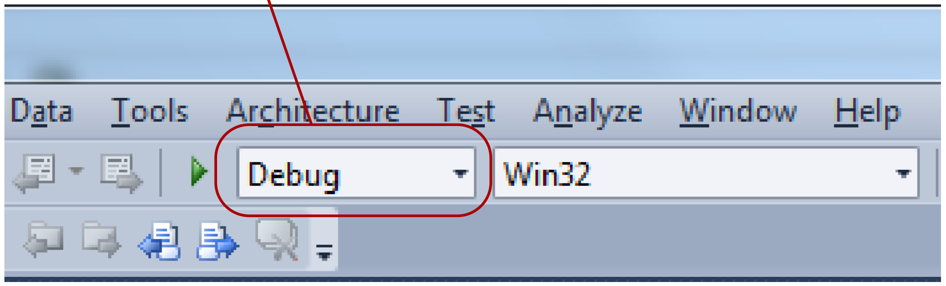
## *Debugging Statements*

```
//==============================================
// MySource.cpp
//==============================================
#include "Whatever.h"

#define DEBUG_1     // Comment out when finished debugging
#define DEBUG_2     // Comment out when finished debugging

--- Bunches of lines of code here ---

#ifdef DEBUG_1
cout << "Step 1 in function ABC().  Var X = " << X << endl;
#endif
--- More lines of code here ---
#ifdef DEBUG_1
cout << "Step 2 in function ABC().  Var Y = " << Y << endl;
#endif

--- Still more lines of code here ---

#ifdef DEBUG_2
cout << "About to call function XYZ() with  Var Y = " << Y << endl;
cout << "      and Var X = " << X << endl;
#endif
```

# Object Oriented Software Testing

## *Using the Visual Studio Debugger*

**Make sure Debug is selected when compiling.**

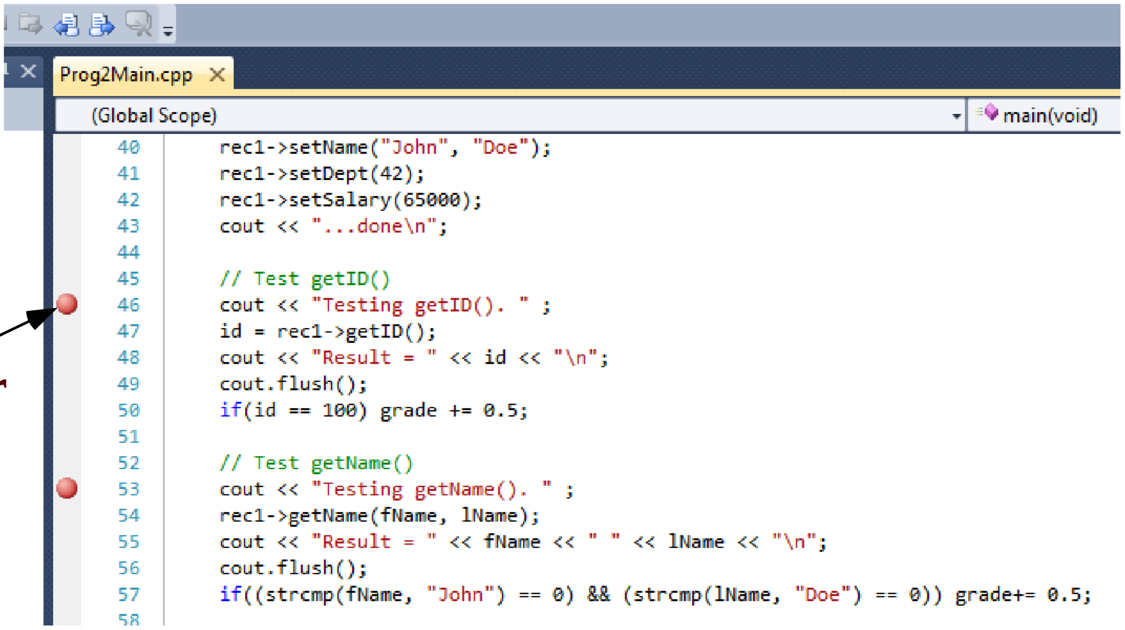


**BTW: You will need to set this to Release if you plan on giving your executable to someone else to run on their machine.**

# Object Oriented Software Testing

## *Using the Visual Studio Debugger*

- Set break points in source code before running.



```
rec1->setName("John", "Doe");
rec1->setDept(42);
rec1->setSalary(65000);
cout << "...done\n";

// Test getID()
cout << "Testing getID(). " ;
id = rec1->getID();
cout << "Result = " << id << "\n";
cout.flush();
if(id == 100) grade += 0.5;

// Test getName()
cout << "Testing getName(). " ;
rec1->getName(fName, lName);
cout << "Result = " << fName << " " << lName << "\n";
cout.flush();
if((strcmp(fName, "John") == 0) && (strcmp(lName, "Doe") == 0)) grade+= 0.5;
```

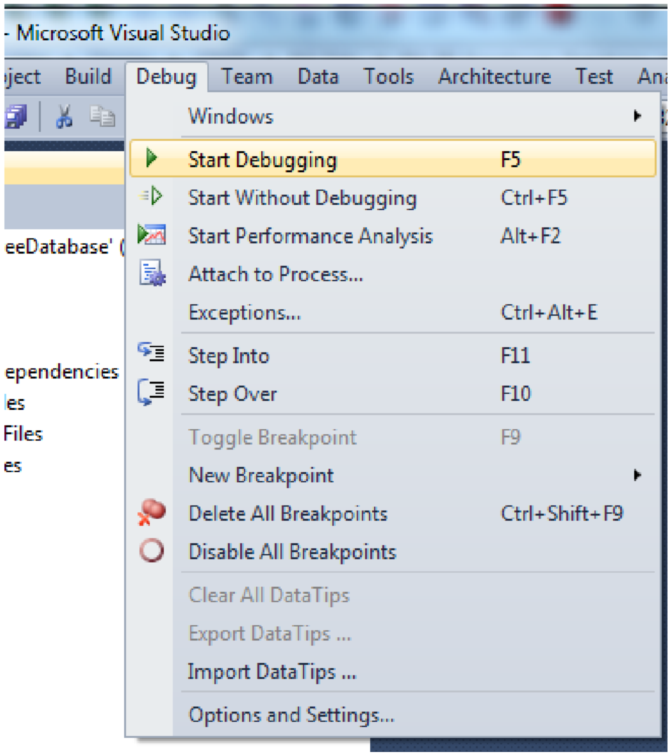
Click to set/clear a break point at a line of code.

You can also set/clear more break points while the application is running in the debugger.

# Object Oriented Software Testing

## *Using the Visual Studio Debugger*

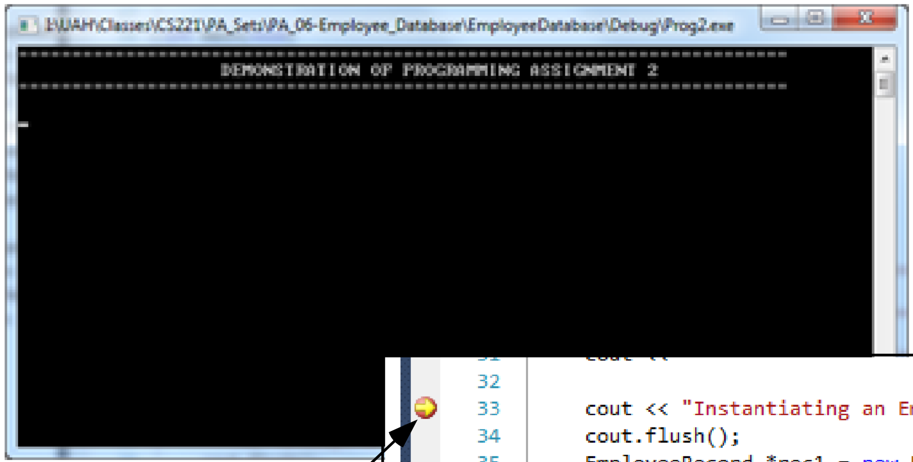
- **Compile the project**
- **Select Debug->Start Debugging**



# Object Oriented Software Testing

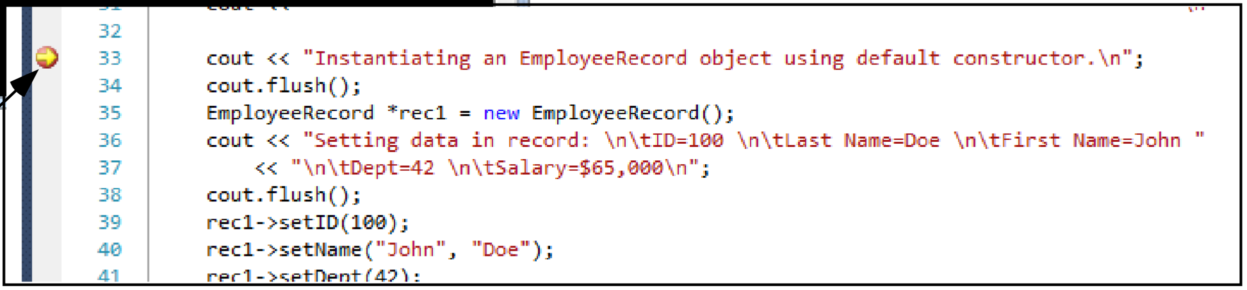
## *Using the Visual Studio Debugger*

**Once started the program will run until it comes to a break point.**


E:\UAH\Classes\CS221\PA_Sets\PA_06-Employee_Database\EmployeeDatabase\Debug\Prog2.exe

```
=======================================================================
                 DEMONSTRATION OF PROGRAMMING ASSIGNMENT 2
=======================================================================
_
```

**DOS window opens in which program will run.**



```
    cout << "Instantiating an EmployeeRecord object using default constructor.\n";
    cout.flush();
    EmployeeRecord *rec1 = new EmployeeRecord();
    cout << "Setting data in record: \n\tID=100 \n\tLast Name=Doe \n\tFirst Name=John "
        << "\n\tDept=42 \n\tSalary=$65,000\n";
    cout.flush();
    rec1->setID(100);
    rec1->setName("John", "Doe");
    rec1->setDept(42);
```

**Program pauses when a break point is reached.**

- **Press F5 to run to the next break point.**
- **Press F10 to step to the next statement.**
- **Press F11 to step into a function if current code statement is a function call.**

# Object Oriented Software Testing

## *Using the Visual Studio Debugger*

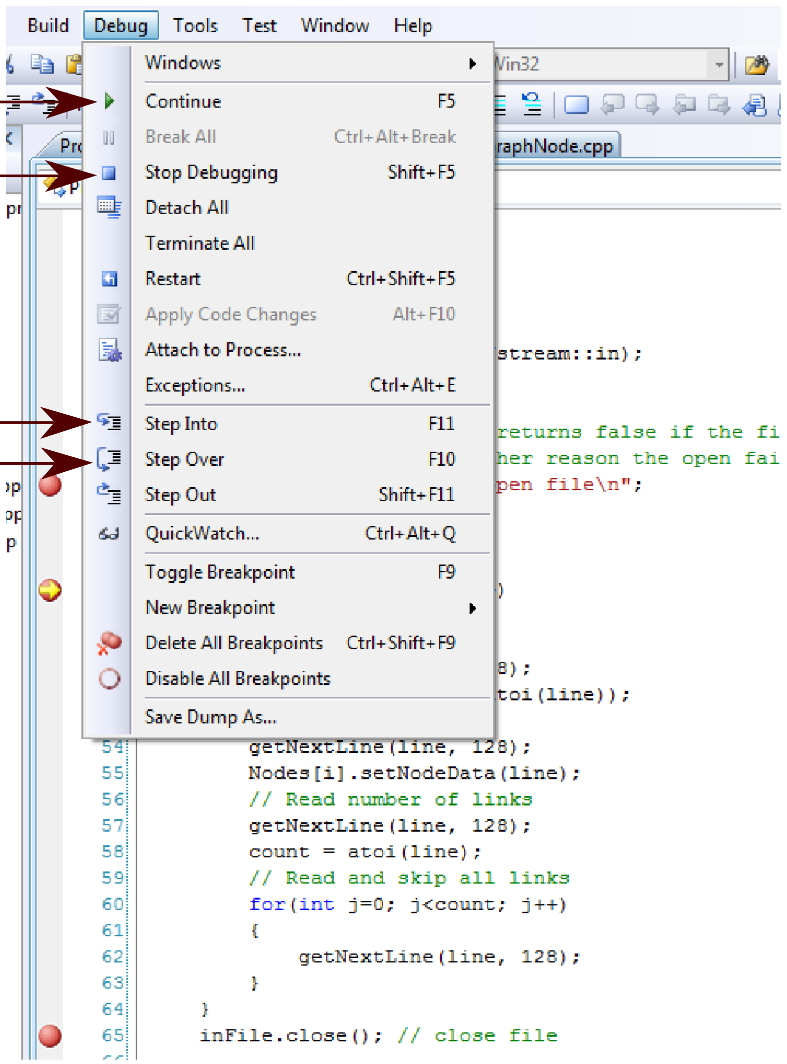
**The Debug menu when running the debugger...**

**Continue-Same as pressing F5**

**Stops the debugger returns to normal file editing mode.**

**Step Into-Same as pressing F11**

**Step Over-Same as pressing F10**

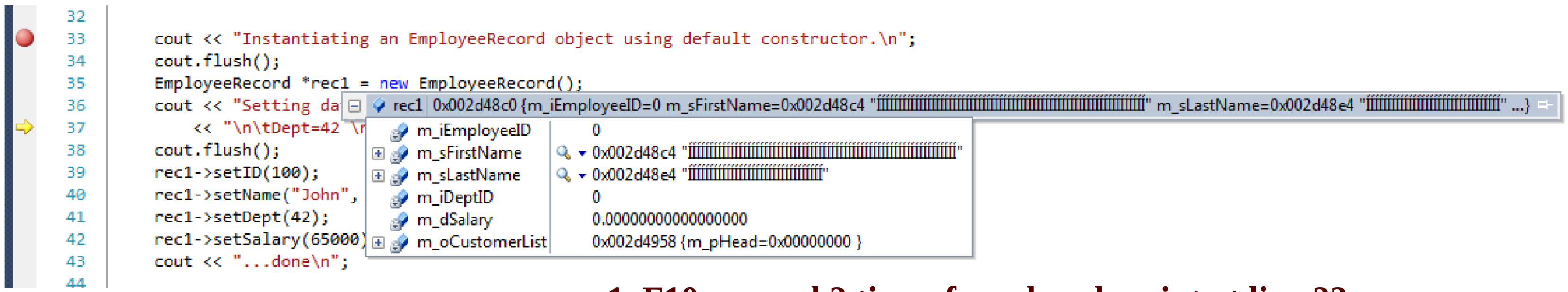


```
stream::in);

returns false if the fi
her reason the open fai
pen file\n";

)

8);
toi(line));

            getNextLine(line, 128);
            Nodes[i].setNodeData(line);
            // Read number of links
            getNextLine(line, 128);
            count = atoi(line);
            // Read and skip all links
            for(int j=0; j<count; j++)
            {
                getNextLine(line, 128);
            }
        }
        inFile.close(); // close file
```

# Object Oriented Software Testing

## *Using the Visual Studio Debugger*

### Checking values or the status of objects

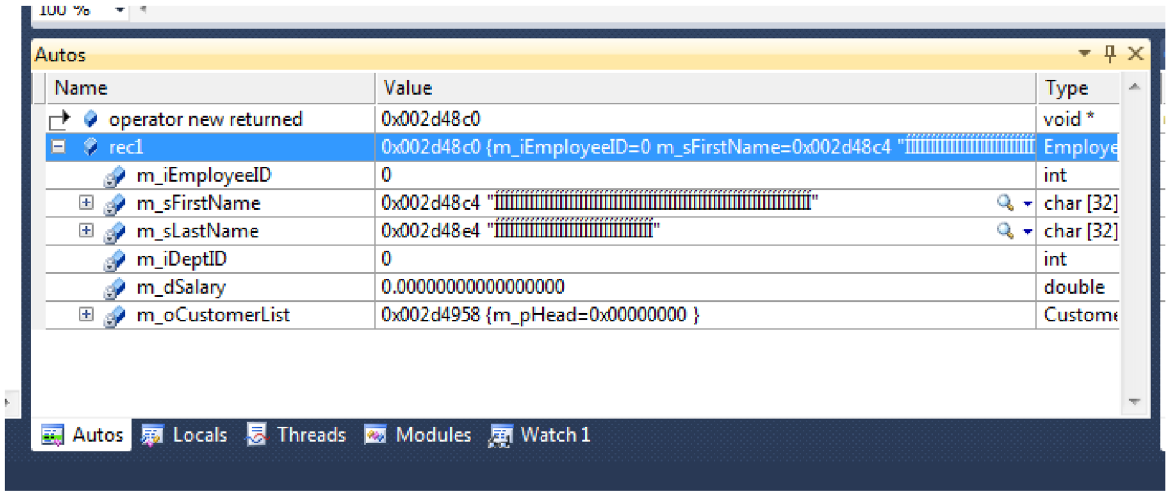


```
32
33      cout << "Instantiating an EmployeeRecord object using default constructor.\n";
34      cout.flush();
35      EmployeeRecord *rec1 = new EmployeeRecord();
36      cout << "Setting da
37          << "\n\tDept=42 \r
38      cout.flush();
39      rec1->setID(100);
40      rec1->setName("John",
41      rec1->setDept(42);
42      rec1->setSalary(65000)
43      cout << "...done\n";
44
```

```
⊟ rec1  0x002d48c0 {m_iEmployeeID=0 m_sFirstName=0x002d48c4 "ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ" m_sLastName=0x002d48e4 "ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ" ...}
     m_iEmployeeID      0
   ⊞ m_sFirstName       0x002d48c4 "ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ"
   ⊞ m_sLastName        0x002d48e4 "ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ"
     m_iDeptID          0
     m_dSalary          0.00000000000000000
   ⊞ m_oCustomerList    0x002d4958 {m_pHead=0x00000000 }
```

**1. F10 pressed 3 times from break point at line 33.**
**2. Mouse curser hovered over *rec1 to display first line.**
**3. Mouse curser moved slowly to ⊞ next to rec1 on overlay.**
**4. Dropdown display appeared and ⊞ changed to ⊟.**

***Note all fields of the class instance have invalid values.***



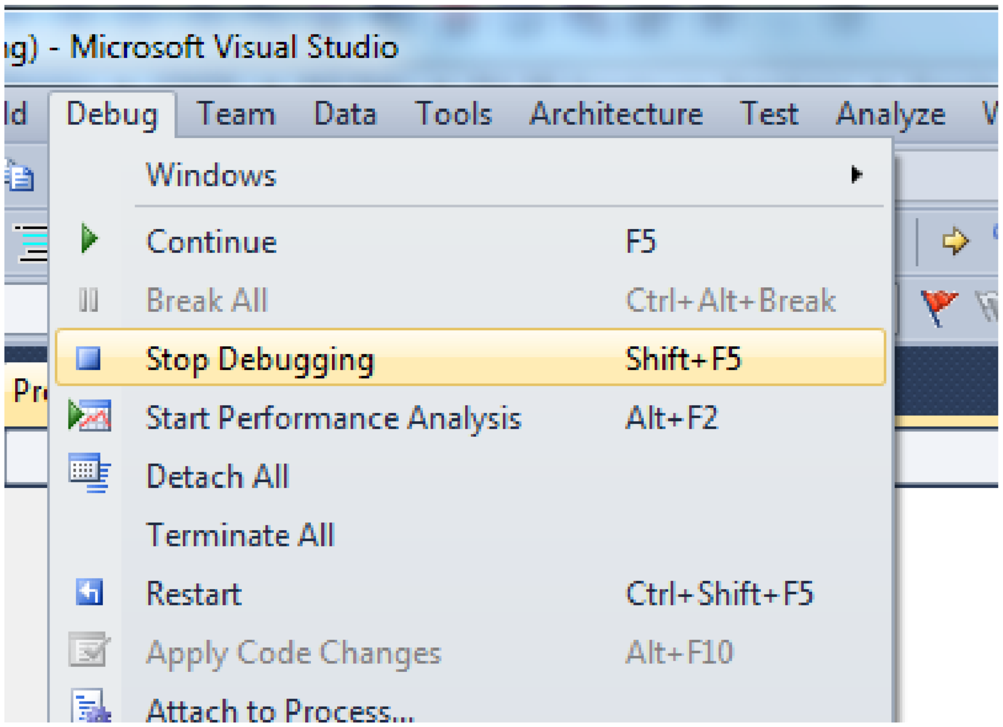
| Name | Value | Type |
|---|---|---|
| operator new returned | 0x002d48c0 | void * |
| ⊟ rec1 | 0x002d48c0 {m_iEmployeeID=0 m_sFirstName=0x002d48c4 "ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ | Employe |
| m_iEmployeeID | 0 | int |
| ⊞ m_sFirstName | 0x002d48c4 "ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ" | char [32] |
| ⊞ m_sLastName | 0x002d48e4 "ÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌÌ" | char [32] |
| m_iDeptID | 0 | int |
| m_dSalary | 0.00000000000000000 | double |
| ⊞ m_oCustomerList | 0x002d4958 {m_pHead=0x00000000 } | Custome |

***Table at the bottom of the window lists all variables that are currently in scope with values.***

# Object Oriented Software Testing

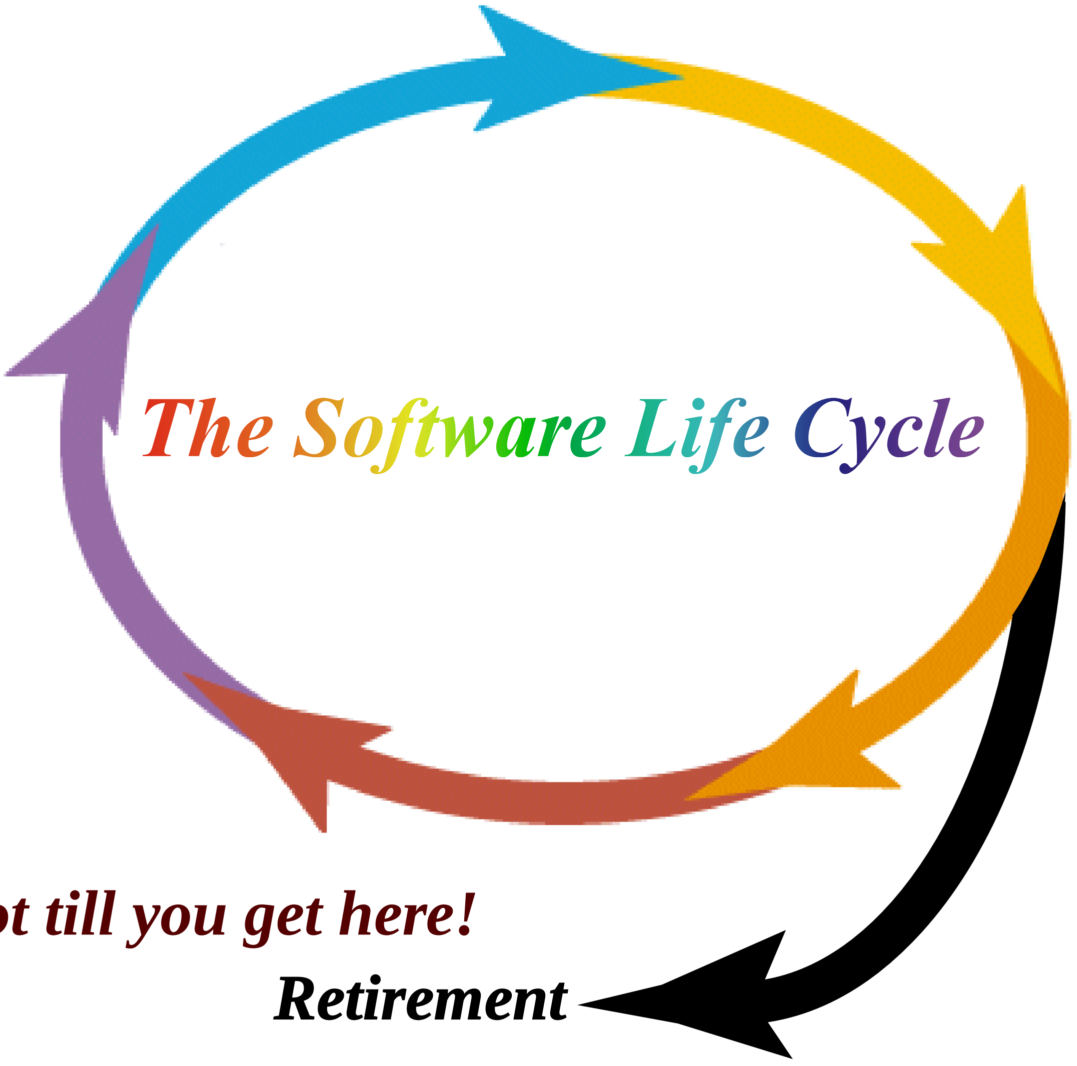
## *Using the Visual Studio Debugger*

**To stop debugging and kill the process select Debug->Stop Debugging.**



# Object Oriented Software Testing

## *When are you done testing?*


*The Software Life Cycle*
*Not till you get here!*
*Retirement*