

# Object Oriented Programming Lab

**FALL - 2022**

**LAB 01**



FAST National University of  
Computer and Emerging Sciences

## **Learning Outcomes**

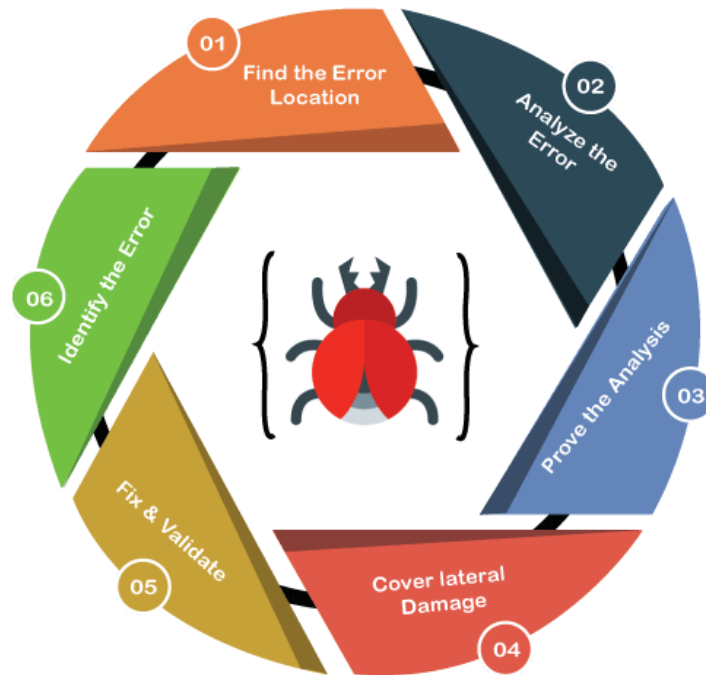
In this lab you are expected to learn the following:

- Debugging
- G-Testing

# Debugging

## Debugging:

An error in the computer program is called a bug. Debugging is the process of finding and resolving bugs within computer programs.



## Part 1: Debugging Task

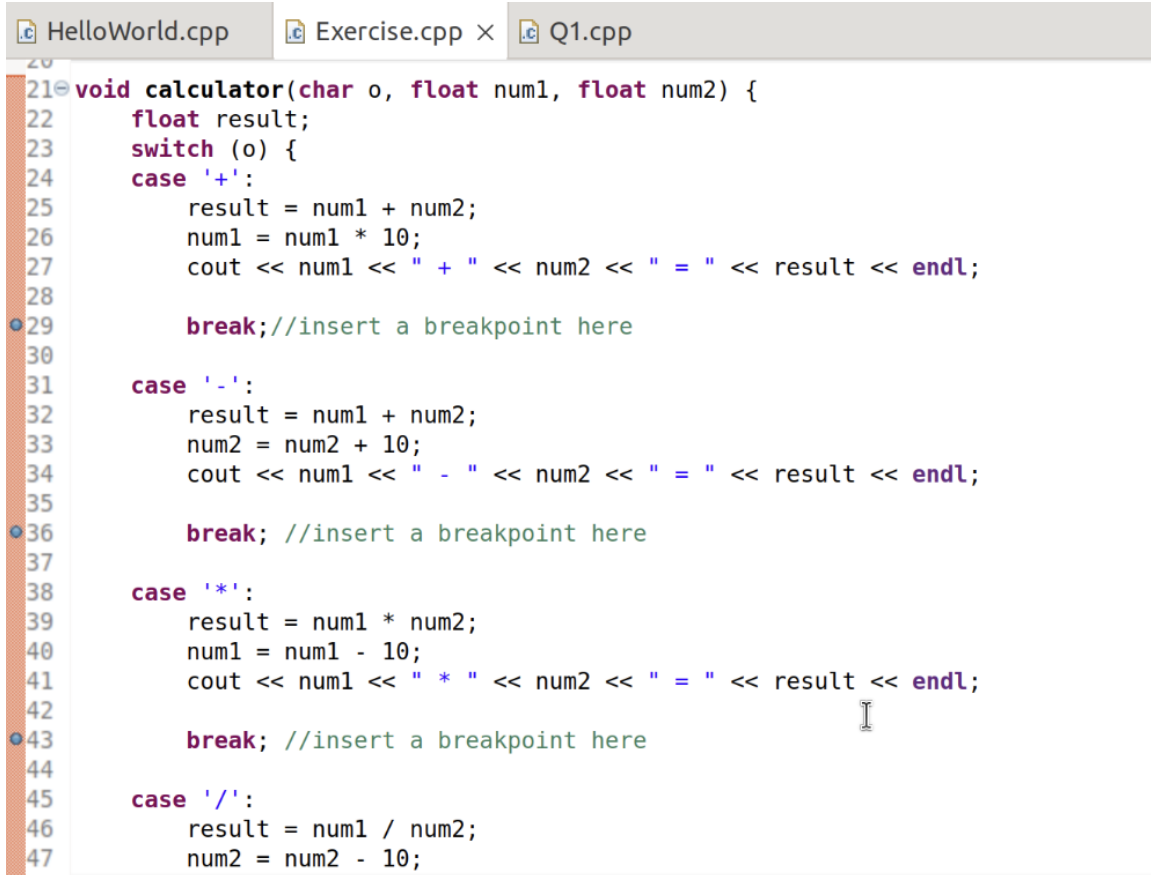
[5 Marks]

1. Open Exercise.cpp file from Debug folder uploaded on the Google Classroom
2. Now open eclipse and create a new C++ project.
3. Name your project as DebugExercise.
4. Copy code from Exercise.cpp file and paste it in your new project file.

There is a calculator function in the Exercise.cpp file:

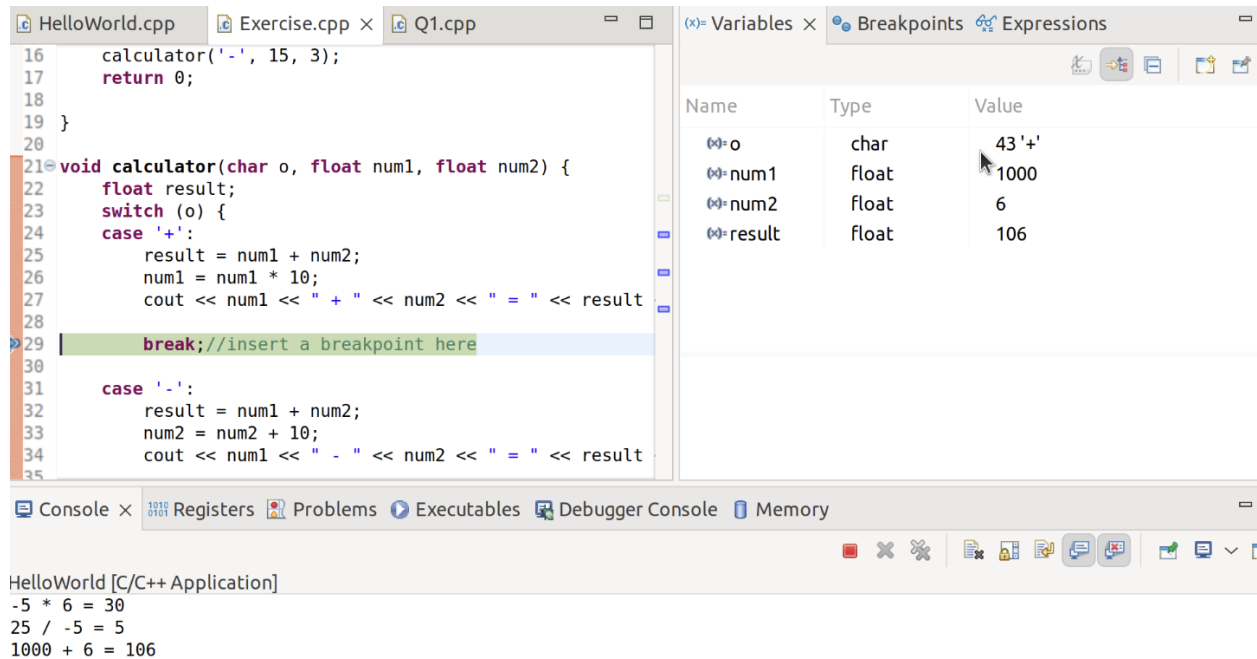
**`calculator(char, float, float);`**

5. Now add breakpoints in the code.
6. For example in the following figure, a code snippet of function calculator() is shown, in which breakpoints are inserted at line 29, 36, 43.

The image shows a screenshot of a C++ IDE with three tabs: 'HelloWorld.cpp', 'Exercise.cpp x', and 'Q1.cpp'. The 'Q1.cpp' tab is active, displaying a C++ program. The program defines a function 'calculator' that takes a character 'o' and two floats 'num1' and 'num2'. It uses a switch statement to handle different operators: '+', '-', '\*', and '/'. Each case performs a calculation, updates 'num1' (e.g., multiplying by 10 for addition and subtraction), and prints the result. Breakpoints are marked with blue dots on the left margin at lines 29, 36, 43, and 47. The code is as follows:

```
20
21 void calculator(char o, float num1, float num2) {
22     float result;
23     switch (o) {
24     case '+':
25         result = num1 + num2;
26         num1 = num1 * 10;
27         cout << num1 << " + " << num2 << " = " << result << endl;
28
29         break; //insert a breakpoint here
30
31     case '-':
32         result = num1 + num2;
33         num2 = num2 + 10;
34         cout << num1 << " - " << num2 << " = " << result << endl;
35
36         break; //insert a breakpoint here
37
38     case '*':
39         result = num1 * num2;
40         num1 = num1 - 10;
41         cout << num1 << " * " << num2 << " = " << result << endl;
42
43         break; //insert a breakpoint here
44
45     case '/':
46         result = num1 / num2;
47         num2 = num2 - 10;
```

7. After adding breakpoints, start debugging the project by clicking on Debug Icon in the Quick Access toolbar.
8. Use step in and step out to check the value of each declared variable inside all functions. For example in the following figure, the debugger stops at breakpoint inserted at line 29, and the values of all variables are shown in the variable window.



9. For each function, at all given breakpoints, you need to report the values of each local variable declared.

**Note:** You need to comment the values after each break point.

## G-Testing

### Installation and Usage of Gtest:

1. Open TestCases\_Tutorial folder uploaded on the Google Classroom
2. Unzip and open the folder
3. Open the terminal of your system and change directory to the current folder.
4. run the following command  

```
bash install-libraries.sh
```
5. The `tests.cpp` file contains the test cases which are to be checked. The `tests.cpp` contains the following code:

```
#include "What_to_test.cpp"
#include <gtest/gtest.h>

TEST(SquareRootTest, PositiveNos) {
    ASSERT_EQ(6.0, squareRoot(36.0));
    ASSERT_EQ(18.0, squareRoot(324.0));
}
```

```

    ASSERT_EQ(25.4, squareRoot(645.16));
    ASSERT_EQ(0, squareRoot(0.0));
}
TEST(SquareRootTest, NegativeNos) {
    ASSERT_EQ(-1.0, squareRoot(-15.0));
    ASSERT_EQ(-1.0, squareRoot(-0.2));
}
int main(int argc, char **argv) {
    testing::InitGoogleTest(&argc, argv);
    return RUN_ALL_TESTS();
}

```

6. The file named *What to test.cpp* contains the function which we want to check in the test cases.

```

#include <math.h>

double squareRoot(const double a) {
    double b = sqrt(a);
    if(b != b) { // nan check
        return -1.0;
    }else{
        return sqrt(a);
    }
}

```

8. After successful installation of libraries, you can run the tests.cpp by the following g++ command.

```
g++ tests.cpp -lgtest -lpthread -o test
```

7. This will create the test executable `test` in the folder. Now you can run the test `test` by issuing a command

```
./test
```

This will show the summary of the test cases which are passed and which are failed.

```

sheherbano@jammyfish:~/Downloads/Lab1/TestCases_Tutorial$ g++ tests.cpp -lgtest -lpthread -o test
sheherbano@jammyfish:~/Downloads/Lab1/TestCases_Tutorial$ ./test
[=====] Running 2 tests from 1 test suite.
[-----] Global test environment set-up.
[-----] 2 tests from SquareRootTest
[ RUN     ] SquareRootTest.PositiveNos
[ OK      ] SquareRootTest.PositiveNos (0 ms)
[ RUN     ] SquareRootTest.NegativeNos
[ OK      ] SquareRootTest.NegativeNos (0 ms)
[-----] 2 tests from SquareRootTest (0 ms total)

[-----] Global test environment tear-down
[=====] 2 tests from 1 test suite ran. (0 ms total)
[ PASSED ] 2 tests.
sheherbano@jammyfish:~/Downloads/Lab1/TestCases_Tutorial$

```

## Part 2: G-Testing Task

[10 Marks]

**Note:** You are supposed to pass the test cases for all functions.

Q1) Write a program that has an integer array having n elements. The program should have a function that can receive the array and then return the sum of all the elements of the array.

**Function Prototype:** `int sumArray(int arr[ ],int size)`

Q2) Write a function that reverses the array using a single loop. The program should have a function that can receive the array and then return the reverse array.

**Function Prototype:** `int *reverseArray(int arr[ ], int size)`

### Submission Instructions:

1. Create a single cpp file containing all the functions of the problems and main function.
2. Save the **cpp** file with the task number  
**e.g. Q1.cpp**
3. Now create a new folder with name *ROLLNO\_SEC\_LAB01* **e.g. i22XXXX\_A\_LAB01**
4. You need to display your roll no and name before the output of each question.
5. Move all of your **.cpp files (without the main function i.e., comment out the main function)** to this newly created directory and compress it into a **.zip file**.
6. Now you have to submit this zipped file on Google Classroom.
7. If you don't follow the above-mentioned submission instruction, you will be marked **zero**.
8. Plagiarism in the Lab Task will result in **zero** marks in the whole category.