# Lab Exercise 1: Basic errors Debugging in C++

## Objectives

This lab introduces debugging through simple arithmetic operations. Students will identify bugs in arithmetic functions and systematically fix them, committing each fix to Git with documentation.

## Expected Learning Outcomes

- Recognize simple programming faults in arithmetic functions.  
- Document and fix each bug clearly with ***<BUG\_x>*** numbering.  
- Apply Git workflow for incremental commits after each bug fix.  
- Verify correctness through provided test cases.

## Identify and Fix

1. Open ***Student\_Lab1.cpp*** and review the code. Each intentional bug is marked with //***<BUG\_x>***.  
2. Run the program and observe incorrect outputs.  
3. Fix one bug at a time, leaving the //***<BUG\_x>*** marker in place and adding a short explanatory comment, when you fix it.  
4. After each fix, commit changes:  
 ***git add Student\_Lab1.cpp***  
 ***git commit -m "Fixed BUG\_x: <short explanation>"***  
5. Re-run program until all outputs are correct.

## (Student\_Lab1.cpp)

The following code contains intentional bugs:

// Student\_Lab1.cpp  
/\*  
Module: COMP H4204 - Games Engineering II  
Lab 1: Arithmetic Operations – Debugging and Testing  
  
Objectives:  
- Implement simple arithmetic functions in C++ using two classes.  
- Practice identifying faults (defects) and fixing them systematically.  
- Document bug fixes clearly with <BUG\_x> numbers in comments.  
- Use Git to track fixes: commit after fixing each bug with a message explaining the fix.  
  
Expected learning outcomes:  
- Understand how simple arithmetic functions can contain subtle faults.  
- Gain experience in systematic debugging and documenting fixes.  
- Use test cases to confirm correctness and prevent regressions.  
  
--- Identify and Fix Instructions ---  
1) Each bug in this code is marked with a placeholder comment like //<BUG\_1>.  
2) When you fix a bug, leave the <BUG\_1> tag and add a short explanation comment.  
 Example: //<BUG\_1 fixed: changed int to double to avoid integer division>  
3) After each fix, commit to Git with a message like:  
 git add Student\_Lab1.cpp  
 git commit -m "Fixed BUG\_1: changed int to double to avoid truncation"  
4) Repeat for all bugs until program runs correctly.  
\*/  
  
#include <iostream>  
using namespace std;  
  
class Arithmetic1 {  
public:  
 // Addition  
 int add(int a, int b) {  
 return a - b; // <BUG\_1>:   
 }  
  
 // Subtraction  
 int subtract(int a, int b) {  
 return a + b; // <BUG\_2>:   
 }  
};  
  
class Arithmetic2 {  
public:  
 // Multiplication  
 int multiply(int a, int b) {  
 return a / b; // <BUG\_3>:   
 }  
  
 // Division  
 double divide(int a, int b) {  
 return a / b; // <BUG\_4>:   
 }  
};  
  
int main() {  
 Arithmetic1 ar1;  
 Arithmetic2 ar2;  
  
 int x = 10, y = 5;  
 cout << "Addition (10+5): " << ar1.add(x,y) << endl;  
 cout << "Subtraction (10-5): " << ar1.subtract(x,y) << endl;  
 cout << "Multiplication (10\*5): " << ar2.multiply(x,y) << endl;  
 cout << "Division (10/5): " << ar2.divide(x,y) << endl;  
  
 return 0;  
}