**Software Engineering 1 – Lab 1 Tasks**

Welcome to the lab sessions for the Software Engineering 1 module.

In this lab, you will be completing various tasks that will help you understand and practice important programming concepts in C++.

All tasks should be completed individually. You may refer to the lab guidance for additional reading or raise your hand if you need help from a tutor.

For each task:

* Write your code in C++.
* Make sure to comment on your code explaining each step.
* Run the program and ensure that it works correctly before submitting.

Good luck, and remember, programming requires practice, so keep experimenting with the code!

**Task 1: Declaring and Initializing Variables**

Objective: Practice declaring and initializing variables of different types in C++.

Instructions:

1. Write a C++ program that declares and initializes variables of the following types:
   1. int, float, char, and string.
2. Display the values of these variables on the console.
3. Change the values of the variables after initializing them and display the updated values.

**Example Output:**

Initial values:

Name: Alice, Age: 20, Height: 1.65, Grade: B

Updated values:

Name: Bob, Age: 22, Height: 1.75, Grade: A

#include <iostream>

#include <string>

using namespace std;

int main() {

    int age = 25; // Declaring a variable that has the data type integer and contains the inital value of 25

    float grade = 99.5f; // Declaring a variable that has the data type float and contains the inital value of 99.5

    char character = 'a'; // Declaring a variable that has the data type char and contains the inital value of a

    string outputString = "This is a test string"; // Declaring a variable that has the data type string and contains the inital value of 'This is a test string'

//Initalising a new value in variables

    age = 50;

    grade = 40.5f;

    character = 'c';

    outputString = "New Test String";

 // Outputting all the variables to the console, adding a new line at the end of each (endl is a new line at the end of each statement)

    cout << age << endl;

    cout << grade << endl ;

    cout << character << endl;

    cout << outputString << endl;

}

**Task 2: Arithmetic Operations with User Input**

Objective: Practice performing basic arithmetic operations.

Instructions:

1. Write a C++ program that accepts two integers from the user.
2. Perform addition, subtraction, multiplication, division (using type casting for float division), and modulus on the two integers.
3. Display the results of each operation.

Example Output:

Enter the first number: 12

Enter the second number: 5

* Addition: 17
* Subtraction: 7
* Multiplication: 60
* Division: 2.4
* Modulus: 2

#include <iostream>

using namespace std;

int main() {

    int number1;

    int number2;

    cout << "Input the first number: ";

    cin >> number1;

    cout << "Input the second number: ";

    cin >> number2;

    int addNumbers = number1 + number2;

    int subtractNumbers = number1 - number2;

    int multiplyNumbers = number1 \* number2;

    float divideNumbers = float(number1) / float(number2);

    int modulusNumbers = number1 % number2;

    cout << "Addition: " << addNumbers << endl;

    cout << "Subtraction: " << subtractNumbers << endl;

    cout << "Multiplication: " << multiplyNumbers << endl;

    cout << "Division: " << divideNumbers << endl;

    cout << "Modulus" << modulusNumbers << endl;

}

**Task 3: Variable Scope and Lifetime**

Objective: Understand the scope and lifetime of variables in C++.

Instructions:

1. Write a program that demonstrates the use of local, global, and static variables.
2. Create a function that increments a local variable, a static variable, and a global variable.
3. Call this function multiple times and observe the behavior of each type of variable.

#include <iostream>

using namespace std;

int globalVariable = 1;

void function() {

    int localVariable = 1;

    static int staticVariable = 1;

    localVariable++;

    staticVariable++;

    globalVariable++;

    cout << " Local: " << localVariable << " Global: " << globalVariable << " Static: " << staticVariable << endl;

}

int main() {

    function();

    function();

    function();

    return 0;

}

**Task 4: Type Casting and Conversion**

Objective: Understand implicit and explicit type casting in arithmetic operations.

Instructions:

1. Write a program that accepts two integers and performs the following operations:
   1. Implicit type conversion in division.
   2. Explicit type conversion (using static\_cast) to cast one of the integers to a float before division.
2. Display the results of both operations and explain the difference in your comments.

Example Output:

Enter the first number: 15

Enter the second number: 2

* Result without type casting (integer division): 7
* Result with type casting (float division): 7.5

#include <iostream>

using namespace std;

int main() {

    int number1, number2;

    cout << "Enter the first number: ";

    cin >> number1;

    cout << "Enter the second number: ";

    cin >> number2;

    int divisionNormal = number1 / number2;

    cout << "Result with implicit casting: " << divisionNormal << endl;

    float divisionCasting = static\_cast<float>(number1) / number2;

    cout << "Result with explicit casting: " << divisionCasting << endl;

    return 0;

}

**Task 5: Complex Arithmetic Expression and Precedence**

Objective: Practice writing complex arithmetic expressions while understanding precedence and associativity.

Instructions:

1. Write a program that calculates the result of a complex arithmetic expression using multiple operators (+, -, \*, /, and %).
2. Ensure the expression uses parentheses to change the default precedence order.
3. Display the result of the expression with and without parentheses and explain the difference.

Example:

* Without parentheses: 5 + 3 \* 2 / 4 - 1
* With parentheses: (5 + 3) \* (2 / (4 - 1))

Example Output:

* Without parentheses: Result = 5
* With parentheses: Result = 5.33

#include <iostream>

using namespace **std**;

int **main**() {

    int withoutPrecedence = 10 + 8 \* 4 / 8 - 4;

    cout **<<** "Result without precedence: " **<<** withoutPrecedence **<<** **endl**;

    int withPrecedence = (10 + 8) \* (4 / (8 - 4));

    cout **<<** "Result with precedence: " **<<** withPrecedence **<<** **endl**;

}

Refer to the lab guide for more details on the above tasks.