Faculty of Computing

CS110: Fundamentals of Computer Programming

Class: BESE-16B

Lab 02: Basic I/O, Variables, Data Types, and Operators

CLO 1	**Understand** the syntax and semantics of different programming constructs
CLO 3	**Build** a program and associated documentation using appropriate IDE and supplementary tools

Date: 16th September, 2025

Time: 2:00pm-5:00pm

Instructor: Dr. Momina Moetesum
Lab Engineer: Mr. Nadeem Nawaz

Introduction:
This lab introduces students to the fundamental structure of a C++ program and its execution workflow. Through a set of guided tasks, students will practice writing simple C++ programs, explore the role of operators and expressions, and analyze how memory and bitwise operations work in real-world contexts. The lab also emphasizes the use of Visual Studio IDE to compile, run, and debug programs, providing hands-on familiarity with the programming environment.

Learning Objectives:
After completing this section, you will be able to:

- **Illustrate** the use of basic C++ syntax elements such as `cout`, escape sequences, and operator precedence.
- **Identify** and assign appropriate data types to variables in C++.
- **Develop** and **execute** programs in Visual Studio IDE to perform memory analysis and bitwise operations.
- **Demonstrate** program output with proper documentation and screenshots in a structured report.

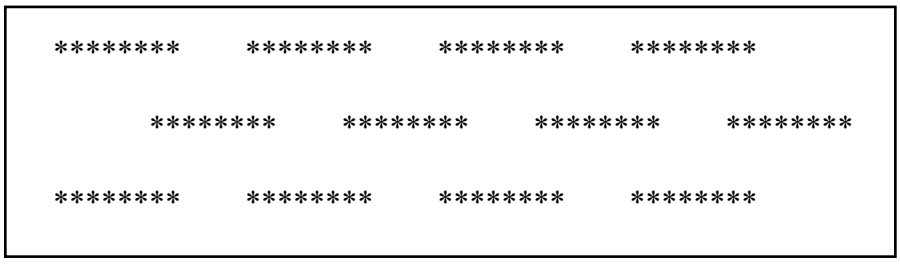
Tools/Software Requirement:

- Microsoft Visual Studio (any version)
- C++ Compiler (integrated within Visual Studio)
- Word Processor (MS Word or equivalent for compiling deliverables)

Task 1 [CLO 1]:

Illustrate the use of **`cout`** along with escape sequences (**`\t`** and **`\n`**) by reproducing the following pattern on the command line. **Explain** in comments how escape sequences control the output format. Paste your C++ code and the screenshot of the output pattern.

Note: Outer rectangle is not part of the pattern.



Task 2 [CLO 1]:

Illustrate your understanding of **operator precedence** in C++ by writing a simple program that computes the following equation.

$$\frac{4}{3(r + 34)} - 9(a + bc) + \frac{3 + d(2 + a)}{a + bd}$$

1. **Identify** and assign appropriate data types to each variable (`a, b, c, d, r`).
2. **Explain** in comments how operator precedence affects the evaluation of the expression.
3. **Test** the program with the input values: `a = 5, b = 3, c = 4.5, d = 6, r = 6.12`.
4. **Summarize** your observation about how C++ evaluates the expression step by step.

Task 3 [CLO 3]:

Develop and **execute** a C++ program to analyze memory usage in a financial application. The program should:

1. **Declare** variables of type `int`, `float`, `double`, `char`, and `bool`.
2. **Use** the `sizeof()` operator to display the memory size (in bytes) of each data type.
3. **Calculate** the total memory consumed by storing account data with the following fields:
   - `int` for account number
   - `float` for transaction amounts
   - `char` for account status (A, D, or S)
   - `bool` for active status
4. **Extend** the calculation to 10,000 accounts and **convert** the memory consumption into KB or MB if it exceeds 1024 bytes.
5. **Demonstrate** results with code and output screenshots in your lab report.

Task 4 [CLO 3]:

Construct and **implement** a C++ program that simulates a simple data compression operation using bitwise operators. The program should:

1. **Prompt** the user to input two 8-bit integers.
2. **Perform** and **display** the following operations in binary format:
   - Bitwise AND (`&`)
   - Bitwise OR (`|`)
   - Bitwise XOR (`^`)
   - Left shift (`<<`) of the first number by 2 bits
   - Right shift (`>>`) of the second number by 3 bits
3. **Demonstrate** the working of the program with the code and execution screenshot.

Note: An **8-bit integer** is a number that can be represented using **8 binary digits (bits)**.

- **Unsigned 8-bit integer** → range **0 to 255** (e.g., `25 = 00011001`).
- **Signed 8-bit integer** → range **–128 to +127** (e.g., `-5 = 11111011` in two's complement).

For this lab, assume **unsigned 8-bit integers (0–255)** when entering values.

Deliverables:

Compile a single Word document as displayed in solution/answer part and submit this Word file on LMS.

Lab Rubrics:

Your Lab 2 will be graded out of 5 for each rubric according to the following rubrics.

Lab Rubrics for Lab 2					
Sr. No.	**Assessment**	**Unacceptable (0 Marks)**	**Does Not Meet Expectations (1/2 Marks)**	**Meets Expectations (3/4 Marks)**	**Exceeds Expectations (5 Marks)**
1	**Illustrating the basic understanding of semantics and syntax (CLO1, PLO1)**	The student did not submit any work. OR The student plagiarized the solution and/or used unfair means.	The student is unable to demonstrate the understanding of syntax of C++ language and is unable to write an executable code. The student is not able to understand the structure of a program at all.	The student demonstrates some understanding of syntax of C++ language and is able to write a code with few errors. The student is able to understand the structure but still learning the syntax.	The student demonstrates good understanding of syntax of C++ language and is able to write executable code without help The student is able to understand the structure and is able to identify problems in the code when introduced
2	**Software Tool Usage (CLO3-PLO5)**		The student demonstrates a lack of understanding of tool usage. Implementation has syntax/semantic/runtime errors, and the student is unable to debug and correct the errors. The code has inadequate comments and variable names and does not adhere to the coding standards. No Error handling has been performed. Documentation is poorly structured.	The student demonstrates some understanding of tool usage. The codes are correct in terms of their syntax, however, the program output is not always correct in all test cases. The code has limited comments and inconsistent variable names and may not adhere to the coding standards. Some Error handling has been performed. Documentation is adequately structured.	The student demonstrates a good understanding of tool usage. Furthermore, his/her coding is complete and functional, and the program output is correct in all test cases. The code has sufficient comments and consistent variable names and reasonably adhere to the coding standards. Adequate Error handling has been performed. Documentation is well structured.