Trainer Name: Er. Manish Singh [M.Tech(CSE), B.Tech(CSE)]
Trainer Experience: 12+ Years of IT Industry

Course Overview:

Current C Standard: C23
Training Mode: Hands-on Offline classroom program
Training Duration: Regular 45 Days (2 Hours Per Day)
Training Fees: ₹2500
Training Deliverables:

- Daily Practical Assignment
- Per Day Lab Sessions
- Interview Oriented Programs
- One Complete Real-Time Working Project
- Course Completion Certificate from Naomika Computer Consultancy

About C Language:

The C programming language is the mother of all the programming languages exist today. The C Language is the first computer language that students learn to develop thinking capability to solve any real world problem with the help of computer system. It's a general purpose programming language that can be used to develop system software, applications software, and can be used widely in embedded system. It has a rich history that dates back to the early 1970s. It was developed by Dennis Ritchie at Bell Labs as a successor to the B language, which was itself derived from BCPL (Basic Combined Programming Language). The primary goal of C was to create a language that could be used to write the UNIX operating system, which was also being developed at Bell Labs at the time.

Objective of C Language Course:

The main objective of the C language course is to make the students learn about fundamentals of programming and to develop thinking capability about different programming logics. After Completion of the course, students can be able to attend any MNC Company interview and can solve the technical rounds both theoretically and Practically.

Why Is C Still Relevant Today?

There is no any interview without C language. As mentioned above, C is the mother of all the modern languages. After completing the C language course, a student can learn any other programming language like C++, Java, Python etc. quickly. C is still used today for developing high performance software like in gaming, networking, GUI programming, device driver, embedded system etc.

Course Contents:

Module 1: Introduction

- Computer Program
- Computer Programming
- Programming Language
- Introduction to C23
    - History and Evolution of the C Language
        - ✓ Origin of the C Language
        - ✓ Evolution of C
        - ✓ Modern C: C99, C11, C17, C23
        - ✓ The Role of C in Modern Programming
    - The Importance of C in Modern Programming
        - ✓ C as the Foundation of Modern Computing
        - ✓ Why C is Still Relevant Today
        - ✓ C in Modern Domain
        - ✓ Learning C as Foundation
    - Setting Up Your Development Environment
        - ✓ Choosing a Compiler with C23 Support
        - ✓ Installing an Integrated Development Environment
- Introduction to Compiler, Interpreter, and Assembler

Module 2: Fundamentals of C23

- The Basic Syntax and Program Structure
    - The Structure of a C Program
    - Basic Syntax Rule
    - Writing Your First C Program
    - Introduction to Input and Output Functions (printf, scanf)
    - Common Pitfalls and Best Practices
- Understanding Build Process(Preprocessing→Compilation→Linking→Execution)
- Data Types and Variables
    - Data Types in C23
    - Identifiers and Naming Rules
    - How does the data get stored in Computer Memory?
    - Variables
        - ✓ Variable Declaration
        - ✓ Variable Initialization
        - ✓ Variable Assignment
    - Keywords
    - Constants
    - Type Modifiers
    - Type Conversion
- Operators and Expressions
    - What are Operators and Expressions?
    - Types of operators in C23
    - Operator Precedence and Associativity
    - Evaluation of Expressions
    - Practical Examples
- Control Flow: Conditionals and Loops
    - Conditional Statements
        - ✓ if
        - ✓ if-else
        - ✓ Nested if-else
        - ✓ If-else-if Ladder

        - ✓ switch Statement
    - Loop
        - ✓ while Loop
        - ✓ for
        - ✓ do-while
    - Jump Statement
        - ✓ break
        - ✓ continue
    - Control Flow Best Practices
    - Practical Examples
- Input and Output in C23
    - Standard Input and Output
    - Formatted Output
    - File Input and Output
    - Error Handling
    - Practical Examples

Module 3: Array and String

- Working with Arrays
    - What is an Array?
    - Declaring Arrays
    - Initializing Arrays
    - Accessing Array Elements
    - Modifying Array Elements
    - Common Operations on Arrays
    - Common Pitfalls with Arrays
    - Best Practices for Working with Arrays
    - Practical Examples
- Multidimensional Arrays
    - What are Multidimensional Arrays?
    - Declaring Multidimensional Arrays
    - Initializing Multidimensional Arrays
    - Accessing Elements in Multidimensional Arrays
    - Modifying Elements in Multidimensional Arrays
    - Common Operations on Multidimensional Arrays
    - Common Pitfalls in Multidimensional Arrays
    - Best Practices for Working with Multidimensional Arrays
    - Practical Examples
- Strings and String Manipulation
    - What is a String?
    - Declaring and Initializing Strings?
    - Accessing String Elements
    - Modifying Strings
    - Common String Operations
    - Common Pitfalls with Strings
    - Best Practices for Working with Strings
    - Practical Examples
- Common String Functions in C23
    - String Length(strlen)
    - String Copy(strcpy,strncpy)
    - String Concatenation (strcat, strncat)
    - String Comparison (strcmp, strncmp)
    - String Search (strstr, strchr)
    - String Tokenization(strtok)
    - String to Number Conversion (atoi, textttatof, strtol, strtod)

Module 4: Pointers and Memory Management

- Understanding Pointers
  - Understanding Memory Address
  - What is a Pointer?
  - Declaring and Initializing Pointers
  - Accessing the Value Pointed to by a Pointer
  - Pointer Arithmetic
  - Pointers and Arrays
  - Pointers to Pointers
  - Void Pointers
  - Null Pointers
  - Common Pitfalls with Pointers
  - Best Practices for Using Pointers
  - Practical Examples
- Pointer Arithmetic and Operations
  - Basics of Pointer Arithmetic
  - Pointer Subtractions (Between Two Pointers)
  - Pointer Comparison
  - Pointer Dereferencing
  - Pointer Arithmetic with Arrays
  - Pointer Arithmetic with Strings
  - Pointer Arithmetic with Dynamic Memory
  - Common Pitfalls with Pointer Arithmetic
  - Best Practices with Pointer Arithmetic
  - Practical Examples
- Dynamic Memory Allocation (malloc, calloc, realloc, free)
  - Why Do We Use Dynamic Memory Allocation?
  - The malloc Function
  - The calloc Function
  - The realloc Function
  - The free Function
  - Common Pitfalls with Dynamic Memory Allocation
  - Best Practices for Dynamic Memory Allocation
  - Practical Examples
- Smart Pointers in C23
  - What are Smart Pointers?
  - Implementing Smart Pointers in C23
  - Advantage of Smart Pointers
  - Limitation of Smart Pointers in C
  - Practical Examples

Module 5: Functions in C23

- Defining and Calling a Function
  - What is Function?
  - Advantages of Using a Function
  - Defining a Function
  - Function Parameters and Arguments
  - Return Values
  - Calling a Function
    - ✓ Call by Value and Call by Reference
  - Function Prototypes
  - Nested Functions
  - Functions Returning Pointers
  - Passing 1-D Arrays to Functions
  - Passing 2-D Arrays to Functions

- Pointer to Function
- Scope
- Best Practices for Defining and Calling Functions

- Recursion
  - What is Recursion?
  - Structure of Recursive Function
  - Iteration vs Recursion
  - Tail Recursion
  - Advantage and Disadvantage of Recursion
  - Practical Example
- Inline Function in C23
  - What is Inline Function
  - Syntax of Inline Functions
  - How Inline Functions Work
  - Advantages and Disadvantages of Inline Function
  - Inline Functions vs Macros
  - Best Practices for Using Inline Functions

Module 6: Structures and Unions

- Defining and Using Structures
  - What is a Structure?
  - Define a Structure
  - Declaring Structure Variables
  - Initializing Structures
  - Accessing Structure Members
  - Modifying Structure Members
  - Nested Structures
  - Array of Structures
  - Common Pitfalls with Structures
  - Best Practices for Using Structures
  - Practical Examples
- Pointers to Structures
  - What is a Pointer to a Structure?
  - Declaring Pointers to Structures
  - Initializing Pointers to Structures
  - Accessing Structure Members Via Pointers
  - Modifying Structure Members via Pointers
  - Dynamic Memory Allocation for Structures
  - Passing Structures to Functions
  - Function Returning Structures
  - Common Pitfalls with Pointers to Structures
  - Best Practices for Using Pointers to Structures
  - Practical Examples
- Union and Their Applications
  - What is a Union?
  - Defining a Union
  - Declaring Union Variables
  - Initializing Unions
  - Accessing Union Members
  - Modifying Union Members
  - Common Applications of Unions
  - Common Pitfalls with Unions
  - Best Practices for Using Unions
  - Practical Examples

- New Features for Structures and Unions in C23
    - Enhanced Designated Initializers
    - Improved Type Safety
    - New Attributes for Structures and Unions
    - Anonymous Structures and Unions
    - Flexible Array Members

Module 7: Storage Classes in C23

- Pre-existing Storage Classes
    - auto
    - register
    - static
    - extern
- New Storage Classes Specifier
    - constexpr
    - thread_local

Module 8: Pre-processor Directives

- Key Pre-processor Directives in C23
    - #include
    - #define
    - #undef
    - #error
    - #pragma
    - #embed
- Conditional Compilation Directives
    - #if
    - #else
    - #elif
    - #endif
    - #ifdef
    - #ifndef
    - #elifdef
    - #elifndef

Module 9: File Handling

- Opening and Closing Files
    - What is File Handling?
    - Opening a File
    - Closing a File
    - Common Pitfalls with Opening and Closing Files
    - Best Practices for Opening and Closing Files
    - Practical Examples
- Reading and Writing Files
    - Reading from Files
    - Writing to Files
    - Common Pitfalls with Reading and Writing Files
    - Best Practices for Reading and Writing Files
    - Practical Examples
- Error Handling in File Operations
    - Importance of Error Handling in Files Operations
    - Common File Operations Error
    - Error Handling Techniques
    - Best Practices for Error Handling
- Working with Binary Files

- Understanding Binary Files
- Opening and Closing Binary Files
- Reading from Binary Files
- Writing to Binary Files
- Working with Structs in Binary Files
- Random access in Binary Files
- Error Handling in Binary File Operations
- Best Practices for Working with Binary Files