C Programming Syllabus Design

A Structured Academic Curriculum for C Programming

Author: D Charan Jeet

Curriculum Designer & Technical Author

Target Audience: B.Tech / Computer Science Students
Prerequisites: Basic Logic & Mathematics
Format: Modular Progressive Learning

January 29, 2026

Contents

Chapter 1

Level I: BASICS

Building the Foundation of Computational Thinking

Module 1: Basics of C Programming

Foundation | 10 Hours

Learning Objectives

· Understand the history and significance of the C language.

· Master the compilation process and development environment.

· Gain proficiency in basic data types, variables, and operators.

Conceptual Topics

- **Introduction to C & Programming Concepts**
  - History & Features of C (Speed, Portability, Structure).
  - Structure of a C Program (Header, Main, Body).
  - Compilation Process: Preprocessing, Compiling, Assembling, Linking (GCC, IDEs).
- **Fundamentals of Syntax**
  - Keywords, Identifiers, Constants, and Variables.
  - Data Types: Primitive (`int`, `float`, `char`, `double`, `void`).
  - Input & Output Functions: `printf()`, `scanf()`, Format Specifiers.
- **Operators in C**
  - Arithmetic, Relational, Logical.
  - Assignment, Increment/Decrement, Bitwise (Intro).
  - Expressions & Operator Precedence.

Practical Labs & Basic Programs

- "Hello World" execution.
- Simple Calculator (Arithmetic operations).
- Area & Perimeter calculation programs.
- Temperature conversion (Celsius to Fahrenheit).

Expected Outcome: Students will be able to write, compile, and execute basic C programs and understand how data is stored in variables.

Chapter 2

Level II: CORE CONCEPTS

Mastering Logic, Structure, and Memory Management

Module 2: Control Flow & Decision Making *Core Concept 1 | 12 Hours*

Learning Objectives

· Develop logical thinking capabilities.

· Control the flow of program execution based on conditions.

· Implement iterative logic using loops.

Conceptual Topics

- **Conditional Statements**
  - Simple `if`, `if-else`, Nested `if`.
  - The `else-if` ladder for multiple conditions.
  - `switch-case`: Menu-driven programming.
- **Looping Statements**
  - Entry Controlled Loops: `for`, `while`.
  - Exit Controlled Loop: `do-while`.
- **Jump Statements**
  - `break`, `continue`, `goto`, `return`.

Real-world Applications

- Password validation logic.
- ATM menu systems.
- Pattern generation (Stars, Pyramids).

Expected Outcome: Ability to solve logical problems involving conditions and repetitive tasks efficiently.

Module 3: Functions & Modular Programming

Core Concept 2 | 12 Hours

Learning Objectives

· Understand the "Divide and Conquer" strategy.

· Learn to write reusable and modular code.

· Grasp the concepts of recursion and variable scope.

Conceptual Topics

- **Functions in C**
  - Difference between Library & User-defined functions.
  - Function Declaration (Prototype), Definition, and Function Call.
- **Types of Functions**
  - Arguments vs. No Arguments.
  - Return values vs. Void.
  - Call by Value vs. Call by Reference (Intro).
- **Advanced Function Concepts**
  - **Recursion:** Base case and recursive step (Factorial, Fibonacci).
  - **Scope & Lifetime:** Local, Global, Static variables.
  - **Header Files:** Creating custom header files.
  - Command Line Arguments (argc, argv).

Expected Outcome: Students can decompose complex problems into smaller, manageable functions and understand the memory stack during function calls.

Module 4: Arrays, Strings & Pointers

Core Concept 3 | 15 Hours

Learning Objectives

· Handle collections of data efficiently using Arrays.

· Manipulate text data using Strings.

· Master direct memory access using Pointers.

Conceptual Topics

- **Arrays**
  - Declaration and Initialization.
  - 1D Arrays (Traversing, Sorting, Searching).
  - 2D Arrays (Matrices operations).
- **String Handling**
  - Character arrays vs String literals.
  - Standard Library Functions: `strlen`, `strcpy`, `strcmp`, `strcat`.
- **Pointers**
  - Definition, Address-of (&) and De-reference (*) operators.
  - Pointer Arithmetic.
  - Relationship between Pointers and Arrays.
- **Composite Data Types & Files**
  - Structures vs Unions.
  - `typedef` keyword.
  - Basic File I/O: `fopen`, `fclose`, `fprintf`, `fscanf`.

Expected Outcome: Proficient handling of datasets, text processing, and understanding the memory layout of C programs.

Chapter 3

Level III: ADVANCED TOPICS

Introduction to System-Level Programming Concepts

Module 5: Advanced C Features

Light Advanced | 8 Hours

Learning Objectives

· Manage memory dynamically at runtime.

· Utilize bit-level operations for system programming.

· Understand preprocessor directives for conditional compilation.

Conceptual Topics

- **Dynamic Memory Allocation**
  - Memory segments: Stack vs Heap.
  - Functions: `malloc()`, `calloc()`, `realloc()`.
  - Memory De-allocation: `free()` and preventing memory leaks.
- **Enumerations (enum)**
  - Defining named integer constants.
  - Improving code readability with Enums.
- **Bitwise Operators**
  - AND (&), OR (|), XOR (^), NOT (~).
  - Shift operators (<<, >>) and their applications.
- **The Preprocessor**
  - Macro substitution: `#define`.
  - File inclusion: `#include`.
  - Conditional Compilation: `#ifdef`, `#ifndef`, `#endif`.

Real-world Applications

- Creating dynamic arrays that grow with user input.
- Setting flags in embedded systems (Bitwise).
- Cross-platform code compilation macros.

Expected Outcome: Students gain insight into how C interacts closely with hardware and operating system memory management.

End of Syllabus – Designed by D Charan Jeet