C Programming Language Syllabus

Module 1: Introduction to C Programming

1. **History and Importance of C**
   - Origin of C Programming
   - Why Learn C Programming?
   - Applications of C
2. **Setting Up the Environment**
   - Installing a Compiler (e.g., GCC)
   - Using Integrated Development Environments (IDEs) like Code::Blocks, Dev-C++, Visual Studio Code
3. **First C Program**
   - Writing, Compiling, and Running a Simple Program
   - Understanding Basic Syntax (Functions, Semicolons, Braces)

Module 2: Basic Syntax and Data Types

1. **C Program Structure**
   - Header Files, Main Function, and Statements
   - Variables and Constants
   - Data Types: int, char, float, double
2. **Input and Output**
   - Using printf() and scanf()
   - Formatted Output: Specifiers (e.g., %d, %f, %s)
3. **Operators**
   - Arithmetic, Relational, Logical Operators
   - Assignment Operators, Increment/Decrement Operators

Module 3: Control Flow

1. **Conditional Statements**
   - if, else if, and else Statements
   - Nested If Statements
   - switch Statement
2. **Loops**
   - for, while, and do-while Loops
   - Nested Loops
   - Breaking and Continuing in Loops
3. **Jump Statements**
   - break, continue, and goto

Module 4: Functions

1. **Defining and Calling Functions**
   - Function Syntax and Structure
   - Return Type and Parameters
   - Function Overloading (Optional)
2. **Function Types**
   - User-Defined Functions
   - Library Functions (e.g., sqrt(), pow(), abs())
3. **Recursion**
   - Concept of Recursion
   - Writing Recursive Functions

Module 5: Arrays and Strings

1. **Arrays**
   - Defining and Initializing Arrays
   - Accessing Array Elements
   - Multidimensional Arrays
2. **Strings**
   - Defining Strings
   - String Manipulation (e.g., strlen(), strcpy(), strcat())
   - String Input and Output

Module 6: Pointers

1. **Introduction to Pointers**
   - Pointer Declaration and Initialization
   - Dereferencing and Address Operators
   - Pointer Arithmetic
2. **Pointers and Arrays**
   - Relationship Between Arrays and Pointers
   - Passing Arrays to Functions
3. **Dynamic Memory Allocation**
   - Using malloc(), calloc(), realloc(), and free()
   - Memory Leaks and Proper Deallocation

Module 7: Structures and Unions

1. **Structures**
   - Defining and Using Structures
   - Accessing Structure Members
   - Nested Structures
2. **Unions**
   - Understanding Union and its Difference from Structures
   - Using Unions for Memory Efficiency

Module 8: File Handling

1. **File Operations**
   - Opening and Closing Files (fopen(), fclose())
   - Reading and Writing to Files (fscanf(), fprintf(), fgets(), fputs())
   - Binary File Handling (fread(), fwrite())
2. **File Pointer Operations**
   - Moving File Pointer (fseek(), ftell(), rewind())
   - Error Handling in File Operations

Module 9: Advanced Topics in C

1. **Bitwise Operations**
   - Bitwise AND, OR, XOR, NOT, Left and Right Shift Operators
   - Applications of Bitwise Operations
2. **Command Line Arguments**
   - Understanding argc and argv
   - Passing Arguments to Programs
3. **Preprocessor Directives**
   - Macros (#define, #include, #ifdef, #endif)
   - File Inclusion and Conditional Compilation
4. **Linked Lists**
   - Singly Linked List: Insertion, Deletion, Traversal
   - Doubly Linked List

Module 10: Advanced Data Structures

1. **Stacks and Queues**
   - Implementing Stacks using Arrays and Linked Lists
   - Implementing Queues (FIFO)
   - Circular Queue
2. **Trees**
   - Binary Trees: Traversals, Insertions, Deletions
   - Binary Search Trees
3. **Graphs**
   - Representation of Graphs (Adjacency Matrix, Adjacency List)
   - Depth-First Search (DFS) and Breadth-First Search (BFS)

Module 11: Algorithms

1. **Sorting Algorithms**
   - Bubble Sort, Selection Sort, Insertion Sort
   - Merge Sort, Quick Sort
2. **Searching Algorithms**
   - Linear Search, Binary Search
3. **Graph Algorithms**
   - Dijkstra’s Algorithm for Shortest Path
   - Floyd-Warshall Algorithm

Module 12: Debugging and Optimization

1. **Debugging Techniques**
   - Using Debuggers (e.g., GDB)
   - Common Errors in C and How to Resolve Them
2. **Code Optimization**
   - Time and Space Complexity Analysis
   - Optimizing Algorithms and Code

Hands-On Projects and Assignments

1. **Simple Calculator**: Implementing arithmetic operations with functions
2. **Student Record Management System**: Using structures and file handling
3. **Tic-Tac-Toe Game**: Implementing logic using arrays and functions
4. **Linked List Operations**: Implementing insertion, deletion, and traversal
5. **Bank Management System**: Using files for saving data
6. **Sorting and Searching Visualizer**: Visualizing sorting algorithms in C
7. **Library Management System**: Using linked lists and file handling

Tools and IDEs

- **Compilers**: GCC, MinGW
- **IDE/Editors**: Code::Blocks, Dev-C++, Visual Studio Code, CLion
- **Debugging Tools**: GDB (GNU Debugger), Valgrind