Phone: +91 8882618533
Email: info@geekinstitute.org
Website: www.geekinstitute.org

THE GEEK INSTITUTE OF CYBER SECURITY

Regd. By: E-Max India
Centre Code: EMAX/EK80606
(Recognized By Govt. Of India)

(Building Futures Through Digital Knowledge and Innovation)

Data Structures and Algorithms (DSA) in C++ (06 Months)

Syllabus

1. Foundations of C++ Programming & Algorithm Basics

- **Introduction to DSA:**
  - Importance of data structures and algorithms in software development
  - Overview of problem-solving techniques and algorithmic thinking
- **Setting Up Environment: IDE (Visual Studio,** Code::Blocks), GCC compiler basics
- **C++ Basics for DSA:**
  - Syntax, variables, data types (int, float, char, bool, string)
  - Constants, literals, and type conversions
  - Operators: arithmetic, relational, logical, bitwise
- **Control Flow:**
  - Conditional statements (if, if-else, nested if, switch-case)
  - Loops: for, while, do-while (nested loops and loop control statements: break, continue)
- **Functions and Recursion:**
  - Function declaration, definition, call, parameter passing (by value, by reference, pointers)
  - Function overloading, inline functions, default arguments
  - Recursion fundamentals: base case, recursive case, tail recursion, examples (factorial, Fibonacci)
  - Recursive backtracking introduction
- **Arrays:**
  - Declaration, initialization, multi-dimensional arrays
  - Array traversal and manipulation (insertion, deletion)
  - Passing arrays to functions
- **Time Complexity Basics:**
  - Introduction to algorithm efficiency
  - Understanding Big O notation through simple examples

2: Pointers, Dynamic Memory & Linear Data Structures

- **Pointers in Depth:**
  - Pointer declaration, initialization, dereferencing
  - Pointer arithmetic and arrays
  - Pointer to pointer, pointers with functions (pass by pointer)
- **Dynamic Memory Management:**
  - new and delete operators
  - Dynamic arrays, memory leaks, and best practices
- **Strings and String Handling in C++:**
  - C-style strings vs. std::string
  - Common string operations and functions
- **Structures and typedef:**
  - Defining and using structs
  - Nested structures and pointers to structures
- **Linked Lists:**
  - Concept, advantages over arrays
  - Singly Linked List: node structure, insertion (beginning, end, middle), deletion, traversal, searching

- Doubly Linked List: node structure, bidirectional traversal, insertion, deletion
- Circular Linked List: implementation and use cases
- Memory management in linked lists

- **Stack Data Structure:**
  - Concept and real-world use cases
  - Stack implementation using arrays and linked lists
  - Stack operations: push, pop, peek, isEmpty
  - Applications: expression evaluation (infix, postfix, prefix), balanced parentheses checking
- **Queue Data Structure:**
  - Concept and types of queues (simple queue, circular queue, priority queue, deque)
  - Implementation using arrays and linked lists
  - Queue operations: enqueue, dequeue, front, rear
  - Real-world examples and applications

3: Trees - Theory, Implementation & Traversal Techniques

- **Tree Basics:**
  - Tree terminology: nodes, edges, root, leaves, height, depth, degree
  - Types of trees: general trees, binary trees
- **Binary Trees:**
  - Node structure and properties
  - Recursive and iterative tree traversals: inorder, preorder, postorder
  - Level order traversal using queue (Breadth-First Search on trees)
- **Binary Search Trees (BST):**
  - BST property and operations: insertion, deletion (cases: leaf, one child, two children), searching
  - Finding minimum, maximum, successor, predecessor
  - BST validation and usage
- **Balanced Trees (Conceptual):**
  - Introduction to AVL trees: rotations (left, right, left-right, right-left)
  - Importance of balance factor
- **Heaps:**
  - Definition and types: Min-Heap, Max-Heap
  - Heap property, array representation
  - Heap operations: insertion, deletion, heapify
  - Applications: priority queues, heap sort overview
- **Trie (Prefix Tree):**
  - Trie data structure and applications (autocomplete, spell checking)
  - Implementation basics
- **Memory and Efficiency Considerations in Trees**

4: Sorting, Searching, and Algorithm Analysis

- **Sorting Algorithms (Detailed Implementation & Analysis):**
  - Bubble Sort: optimization and time complexity
  - Selection Sort and Insertion Sort: in-place sorting and stability
  - Merge Sort: divide and conquer strategy, recursive implementation, time & space complexity
  - Quick Sort: partitioning schemes (Lomuto, Hoare), average vs worst-case complexity, tail recursion
  - Heap Sort: building heap, sorting process
  - Comparison of sorting algorithms (when to use what)
- **Searching Algorithms:**
  - Linear search: implementation and use cases

- Binary search: iterative and recursive approaches, prerequisites, applications

- **Algorithm Complexity and Analysis:**
  - Time complexity (Big O, Big Theta, Big Omega)
  - Space complexity
  - Best, average, and worst-case analysis
  - Amortized analysis basics
- **Introduction to Algorithm Design Techniques:**
  - Divide and Conquer
  - Greedy approach (intro)
  - Dynamic Programming (intro)

5: Graph Theory and Hashing Techniques

- **Graph Fundamentals:**
  - Definitions: graph terminology (vertex, edge, degree, weighted/unweighted, directed/undirected)
  - Representations: adjacency matrix, adjacency list, edge list
  - Graph traversal: DFS and BFS detailed implementation (recursive and iterative)
- **Applications of Graph Traversals:**
  - Connected components
  - Cycle detection in directed and undirected graphs
- **Minimum Spanning Tree (MST) (Theory and Algorithm Concepts):**
  - Prim's algorithm
  - Kruskal's algorithm
- **Shortest Path Algorithms (Overview):**
  - Dijkstra's algorithm (working principle and implementation)
  - Bellman-Ford algorithm (conceptual)
- **Hashing:**
  - Hash functions: properties and examples
  - Collision handling techniques: chaining, open addressing (linear probing, quadratic probing, double hashing)
  - Design and implementation of a hash table in C++
  - Applications of hashing

6: Advanced Algorithmic Techniques and Practical Project

- **Dynamic Programming (DP):**
- Key concepts: overlapping subproblems, optimal substructure
- Memoization vs tabulation
- **Classic problems:**
  - Fibonacci sequence
  - 0/1 Knapsack problem
  - Coin change problem
  - Longest Common Subsequence (LCS)
- **Greedy Algorithms:**
- Understanding greedy choice property and optimal substructure
- Problem-solving with greedy algorithms:
  - Activity selection
  - Huffman coding (concept and implementation)
- **Backtracking Algorithms:**
  - Concept and problem-solving approach
  - Classic problems: N-Queens, Sudoku Solver, Subset Sum
- **Complexity Classes and Theory (Overview):**
  - Introduction to P, NP, and NP-Complete problems
  - Importance and impact on algorithm design
- **Final Project:**
  - Students design and implement a complete software project that integrates multiple data structures and algorithms
  - Real-world problem solving (e.g., contact management system, basic search engine, simple game with algorithmic logic)
  - Code optimization and documentation
  - Project presentation, peer review