MICROSOFT TRAINING

Data Structures And Algorithms Course

Master data structures and algorithms with Microsoft-aligned techniques to build efficient, scalable software solutions.

DURATION
48 hours

LEVEL
Intermediate

FORMAT
Instructor-Led

- Globally Recognized Certification
- Expert-Led Hands-On Training
- Flexible Scheduling

Computer Pride | 1st Floor, ICEA Building, Kenyatta Avenue, Nairobi | www.computer-pride.co.ke

COURSE OVERVIEW

Unlock the power of efficient software design and problem-solving with Computer Pride's Microsoft-aligned Data Structures and Algorithms Course. This intermediate-level program is meticulously designed for aspiring and current software developers who want to master the fundamental building blocks of robust and scalable applications. Over 48 intensive hours, you will delve deep into core algorithmic concepts, learning how to select, implement, and analyze data structures and algorithms that drive high-performance computing.

This course transcends theoretical knowledge by focusing on practical application, utilizing industry-standard approaches relevant to Microsoft's engineering principles. You will gain proficiency in Big O notation for complexity analysis, explore various data organizations like arrays, linked lists, trees, graphs, and master essential algorithms including sorting, searching, greedy approaches, and dynamic programming. The benefits extend beyond writing better code; you'll develop a structured problem-solving mindset, critically important for tackling complex software challenges, optimizing existing systems, and excelling in technical interviews at leading tech companies. Elevate your coding skills and career prospects by building a solid foundation in computational thinking and efficient system design.

Duration	48 hours
Level	Intermediate
Vendor	Microsoft

COURSE CURRICULUM

01 Module 1: Introduction to Data Structures & Algorithmic Analysis

- Definition and Importance of Data Structures and Algorithms (DSA)
- Introduction to Algorithmic Problem Solving
- Time and Space Complexity Analysis (Big O Notation, Omega, Theta)
- Asymptotic Notations and Growth of Functions
- Best, Average, and Worst-Case Analysis
- Recurrence Relations and Master Theorem

02 Module 2: Fundamental Data Structures

- Arrays: Static vs. Dynamic Arrays, Operations, Use Cases
- Linked Lists: Singly, Doubly, and Circular Linked Lists, Operations
- Stacks: LIFO Principle, Array and Linked List Implementations, Applications (e.g., function calls)
- Queues: FIFO Principle, Array and Linked List Implementations, Applications (e.g., BFS)
- Hashing: Hash Functions, Collision Resolution Techniques (Chaining, Open Addressing)
- Hash Tables/Maps: Implementations and Performance Characteristics

03 Module 3: Trees and Heaps

- Trees: Terminology, Tree Traversals (Inorder, Preorder, Postorder, Level Order)
- Binary Trees and Binary Search Trees (BSTs): Operations, Balancing Concepts
- AVL Trees and Red-Black Trees (Introduction to self-balancing trees)
- Heaps: Min-Heap, Max-Heap, Heapify, Heap Sort Algorithm
- Priority Queues: Implementation using Heaps, Applications

04 Module 4: Graphs and Graph Algorithms

- Graphs: Terminology, Representations (Adjacency Matrix, Adjacency List)
- Graph Traversals: Breadth-First Search (BFS) and Depth-First Search (DFS)

- Spanning Trees: Minimum Spanning Tree (MST) - Prim's and Kruskal's Algorithms
- Shortest Path Algorithms: Dijkstra's Algorithm, Bellman-Ford Algorithm
- Topological Sorting for Directed Acyclic Graphs (DAGs)

05 Module 5: Sorting and Searching Techniques

- Sorting Algorithms: Bubble Sort, Selection Sort, Insertion Sort
- Efficient Sorting: Merge Sort, Quick Sort, Heap Sort
- Non-Comparison Sorts: Counting Sort, Radix Sort, Bucket Sort
- Searching Algorithms: Linear Search, Binary Search
- Algorithm Design Techniques: Brute Force, Divide and Conquer

06 Module 6: Advanced Algorithmic Paradigms & Problem Solving

- Greedy Algorithms: Principles, Classic Problems (e.g., Activity Selection, Huffman Coding)
- Dynamic Programming: Memoization vs. Tabulation, Overlapping Subproblems, Optimal Substructure
- Classic DP Problems (e.g., Fibonacci, Longest Common Subsequence, Knapsack)
- Backtracking: Principles and Applications (e.g., N-Queens, Sudoku Solver)
- Introduction to NP-Completeness and P vs NP Problem
- Practical Problem Solving and Interview Strategies

CAREER PATHWAYS

Graduates of this program are prepared for the following roles:

> **Software Development Engineer (SDE) specializing in backend systems and API development**

> **Performance Optimization Specialist focusing on identifying and resolving algorithmic bottlenecks**

> **Technical Interview Candidate for top-tier software companies requiring strong DSA proficiency**

FREQUENTLY ASKED QUESTIONS

Q: What foundational programming knowledge or prerequisites are recommended before enrolling in this Intermediate Data Structures and Algorithms course?

To get the most out of this course, participants should have a solid grasp of at least one object-oriented programming language (e.g., C#, Java, Python) including concepts like variables, data types, control structures (loops, conditionals), functions/methods, and basic object-oriented principles. Familiarity with basic debugging and code compilation/execution is also beneficial, as the course will heavily involve writing and analyzing code.

Q: How will mastering Data Structures and Algorithms through this Microsoft-aligned course specifically enhance my performance in technical interviews for software engineering roles?

This course is specifically structured to not only teach you DSA concepts but also to equip you with the problem-solving frameworks necessary for technical interviews. You'll gain extensive practice in analyzing problem constraints, choosing the most efficient data structure and algorithm for a given task, and optimizing solutions for time and space complexity. The practical exercises and module on algorithmic paradigms (like dynamic programming and greedy algorithms) directly mirror the types of challenges posed in interviews at companies, including Microsoft itself, significantly boosting your confidence and success rate.

Q: Will this course cover language-specific implementations of data structures and algorithms, or focus on

conceptual understanding, and what programming language will be primarily used for examples and exercises?

While the course primarily focuses on the conceptual understanding and mathematical analysis of data structures and algorithms, practical implementation is crucial. We will use a common industry-standard language, typically Python or C#, for demonstrating examples and hands-on exercises. This allows participants to understand the core logic which is transferable across languages, while also gaining practical coding experience with efficient implementations.

ABOUT COMPUTER PRIDE

East Africa's Premier IT Training Center

With over 15 years of excellence, Computer Pride delivers globally recognized certification programs from leading technology vendors. Located on Kenyatta Avenue in Nairobi's CBD, our state-of-the-art facilities and expert instructors have empowered thousands of professionals across East Africa.

15+	10,000+	50+
Years of Excellence	Professionals Trained	Vendor Certifications

READY TO ENROL?

Take the next step in your technology career.

Location
1st Flr, ICEA Building
Kenyatta Ave, Nairobi

Email
info@computer-pride.co.ke

Phone
+254 705 900 900

www.computer-pride.co.ke