PROGRAMMING FUNDAMENTALS

COURSE DESCRIPTION

This course aims to provide a basic understanding of computer programming. Students will get introduced to the fundamentals of programming including algorithmic approaches, various types of programming, and problem-solving techniques.

It includes an introduction to programming, data structures, recursion, searching and sorting algorithms, trees, graphs, types of algorithms and approaches, a detailed understanding of backtracking and greedy algorithms, dynamic programming, problem-solving, and code optimization techniques. Participants in the course will get to learn about the components and the process to design a computer program. Developers should have the essential knowledge to learn and understand the programming concepts to become skilled in Computer Programming.

The course will help to introduce the required concept, application, and its types, how to write the given program, relevant problem statements, and assessments at the end of each learning resource.

LEARNING OUTCOMES

By the end of this course, participants will gather thorough knowledge of critical areas and components of computer programming.

We have given a special focus on algorithms and the various approaches on which a computer program runs. The course is divided into 11 modules with an outline as mentioned below: -

- Introduction to Data Structures including the linear and non-linear types
- Understanding of Recursion and its examples
- Searching and Sorting Algorithms
- Introduction and Types of Trees
- Introduction to Graphs
- Algorithmic Approaches and various types of algorithms
- Introduction to Greedy Algorithms
- Application of Backtracking
- Dynamic Programming
- Introduction to Problem Solving and Application of Problem-Solving Techniques
- Introduction to Code Optimization and Application of Code Optimization Techniques

PREREQUISITE SKILLS

A background in GitHub is useful but not a mandate.

COURSE TOPICS

DATA STRUCTURES

- Introduction to Data Structures
- Creation, Insertion, Deletion & Traversal
- Linear & Non-Linear Data Structures

RECURSION

- Introduction to Recursion
- Importance of Recursion
- Examples of Recursion

SEARCHING AND SORTING ALGORITHMS

- Understanding of Algorithms
- Introduction to Searching Algorithms
  - Linear
  - Binary
- Introduction to Sorting Algorithms
  - Bubble Sort
  - Radix Sort
  - Selection Sort
  - Heap Sort
  - Insertion Sort
  - Merge Sort
  - Quick Sort

TREES

- Introduction of Trees
- Types of Trees
- Application of Trees Data Structure

GRAPHS

- Introduction to Graphs
- Comparing Graphs vs Trees
- Application of Graphs
- Traversals of a Graph

TYPES OF ALGORITHMS

- Introduction to Algorithms
- Types of Algorithms
- Application of Algorithms with Examples –
  - Simple recursive algorithms
  - Backtracking algorithms
  - Divide and conquer algorithms
  - Dynamic programming algorithms
  - Greedy algorithms
  - Branch and bound algorithms
  - Brute force algorithms
  - Randomized algorithms

GREEDY ALGORITHMS

- Introduction to Greedy Algorithms
- Application of Greedy Techniques
  - Knapsack Problem
  - Huffman Coding & Decoding

- Dijkstra's Algorithm
- Kruskal's Minimum Spanning Tree
- Prim's Minimum Spanning Tree

BACKTRACKING ALGORITHMS

- Introduction to Backtracking
- Application of Backtracking
  - The Knight-Tour Problem
  - Rat in a maze
  - N-Queen's Problem
  - Hamiltonian Cycle

DYNAMIC PROGRAMMING

- Introduction to Dynamic Programming
- Application of Dynamic Programming
  - Recursion vs Dynamic Programming
  - Fibonacci Numbers
  - Binomial Coefficients
  - Longest Common Subsequence
  - Subset Sum Problem
  - Knapsack Problem

PROBLEM-SOLVING TECHNIQUES

- Introduction to Problem Solving
- Steps in Problem-Solving
  - Problem Analysis
  - Program Design - Algorithm, Flowchart, and Pseudocode
  - Coding
  - Compilation and Execution
  - Debugging and Testing
  - Program Documentation
- Application of Problem-Solving Techniques

CODE-OPTIMIZATION TECHNIQUES

- Introduction to Code-Optimization
- Importance of Code-Optimization
- Application of Code-Optimization
- Code Optimization Techniques
  - Compile Time Evaluation
  - Common sub-expression elimination
  - Dead Code Elimination
  - Code Movement
  - Strength Reduction