C Programming Syllabus

Module 1: Introduction to C

1.1 Overview of C

- **Description**: Introduction to the C programming language, its history, and significance.
- **Learning Outcomes**: Understand the basic structure of a C program and the compilation process.

1.2 First C Program

- **Code Example**:

```
c6 lines

Click to expand

#include <stdio.h>

...
```

- **Practical Exercise**: Write a program to print your name and age.
-

Module 2: Data Types and Operators

2.1 Data Types

- **Description**: Understanding basic data types (int, float, char, etc.) and their usage.
- **Learning Outcomes**: Declare and initialize variables of different data types.

2.2 Operators

- **Description**: Overview of arithmetic, relational, logical, and bitwise operators.
- **Practical Exercise**: Create a simple calculator that performs basic arithmetic operations

Module 3: Control Structures

3.1 Conditional Statements

- **Description**: Using if, if-else, and switch statements to control program flow.
- **Learning Outcomes**: Implement decision-making in programs.

3.2 Loops

- **Description**: Understanding for, while, and do-while loops.
- **Practical Exercise**: Write a program to find the factorial of a number using loops.

Module 4: Functions

4.1 Function Basics

- **Description**: Function declaration, definition, and calling conventions.
- **Learning Outcomes**: Create modular programs using functions.

4.2 Recursion

- **Description**: Understanding recursive functions and their applications.
- **Practical Exercise**: Implement a recursive function to calculate Fibonacci numbers.

Module 5: Arrays and Strings

5.1 Arrays

- **Description**: Single and multidimensional arrays, array operations.
- **Learning Outcomes**: Manipulate arrays effectively.

5.2 Strings

- **Description**: String handling functions (strlen, strcpy, strcat).
- **Practical Exercise**: Write a program to reverse a string.

Module 6: Pointers

6.1 Pointer Basics

- **Description**: Understanding pointers, pointer arithmetic, and their applications.
- **Learning Outcomes**: Use pointers for dynamic memory management.

6.2 Pointers and Arrays

- **Description**: Relationship between arrays and pointers.
- **Practical Exercise**: Create a program that swaps two numbers using pointers.

Module 7: Structures and Unions

7.1 Structures

- **Description**: Declaring and using structures to group related data.
- **Learning Outcomes**: Create and manipulate structures.

7.2 Unions

- **Description**: Understanding unions and their differences from structures.
- **Practical Exercise**: Implement a program to store student records using structures.

Module 8: File Handling

8.1 File Operations

- **Description**: Opening, reading, writing, and closing files.
- **Learning Outcomes**: Perform file I/O operations in C.

8.2 Practical Exercise

- Create a program to read and write student records to a file.

C++ Programming Syllabus

Module 1: Introduction to C++

1.1 Overview of C++

- **Description**: Introduction to C++, its features, and its applications.
- **Learning Outcomes**: Understand the basic structure of a C++ program.

1.2 First C++ Program

- **Code Example**:

```cpp
7 lines

Click to expand

#include <iostream>

using namespace std;

...
```

- **Practical Exercise**: Write a program to display your favorite quote.

Module 2: Object-Oriented Programming

2.1 Classes and Objects

- **Description**: Understanding the concepts of classes and objects in C++.
- **Learning Outcomes**: Create and use classes and objects.

2.2 Inheritance

- **Description**: Implementing inheritance and its types (single, multiple).

- **Practical Exercise**: Create a class hierarchy for different types of vehicles.

Module 3: Polymorphism

3.1 Function Overloading

- **Description**: Understanding function overloading and its benefits.
- **Learning Outcomes**: Implement function overloading in C++.

3.2 Operator Overloading

- **Description**: Overloading operators for user-defined types.
- **Practical Exercise**: Implement a complex number class with operator overloading.

Module 4: Advanced C++ Features

4.1 Templates

- **Description**: Understanding function and class templates for generic programming.
- **Learning Outcomes**: Create template functions and classes.

4.2 Exception Handling

- **Description**: Using try, catch, and throw for error handling.
- **Practical Exercise**: Write a program that handles division by zero using exceptions.

Module 5: Standard Template Library (STL)

5.1 Overview of STL

- **Description**: Introduction to STL and its components (containers, algorithms).
- **Learning Outcomes**: Use STL containers like vector, list, and map.

5.2 Practical Exercise

- Implement a program to manage a list of students using STL.

Module 6: File I/O in C++

6.1 File Operations

- **Description**: Reading from and writing to files using fstream.
- **Learning Outcomes**: Perform file I/O operations in C++.

6.2 Practical Exercise

- Create a program to read and write employee records to a file.

Java Programming Syllabus

Module 1: Introduction to Java

1.1 Overview of Java

- **Description**: Introduction to Java, its features, and platform independence.
- **Learning Outcomes**: Understand the basic structure of a Java program.

1.2 First Java Program

- **Code Example**:

java5 lines

Click to close

```
public class HelloWorld {

public static void main(String[] args) {

...
```

- **Practical Exercise**: Write a program to display your favorite number.

Module 2: Object-Oriented Programming in Java

2.1 Classes and Objects

- **Description**: Understanding classes, objects, and constructors in Java.
- **Learning Outcomes**: Create and use classes and objects.

2.2 Inheritance

- **Description**: Implementing inheritance and its types (single, multilevel).
- **Practical Exercise**: Create a class hierarchy for different types of animals.

Module 3: Interfaces and Abstract Classes

3.1 Interfaces

- **Description**: Understanding interfaces and their implementation.
- **Learning Outcomes**: Use interfaces to achieve abstraction.

3.2 Abstract Classes

- **Description**: Understanding abstract classes and their usage.
- **Practical Exercise**: Implement an abstract class for shapes.

Module 4: Exception Handling in Java

4.1 Exception Basics

- **Description**: Understanding try-catch blocks and exception types.
- **Learning Outcomes**: Handle exceptions gracefully in Java applications.

4.2 Practical Exercise

- Create a program that handles user input errors.

Module 5: Java Collections Framework

5.1 Overview of Collections

- **Description**: Introduction to the Java Collections Framework and its components.
- **Learning Outcomes**: Use collections like List, Set, and Map.

5.2 Practical Exercise

- Implement a program to manage a list of books using ArrayList.

Module 6: Multithreading in Java

6.1 Thread Basics

- **Description**: Understanding threads and their lifecycle.
- **Learning Outcomes**: Create and manage threads in Java.

6.2 Synchronization

- **Description**: Using synchronized blocks for thread safety.
- **Practical Exercise**: Create a program that demonstrates thread synchronization.

Data Structures and Algorithms (DSA) Syllabus

Module 1: Introduction to DSA

1.1 Importance of DSA

- **Description**: Understanding the significance of data structures and algorithms in programming.
- **Learning Outcomes**: Analyze the efficiency of algorithms.

1.2 Time and Space Complexity

- **Description**: Introduction to Big O notation and complexity analysis.
- **Practical Exercise**: Analyze the time complexity of simple algorithms.

Module 2: Arrays and Strings

2.1 Array Operations

- **Description**: Insertion, deletion, and searching in arrays.
- **Learning Outcomes**: Perform operations on arrays efficiently.

2.2 String Manipulation

- **Description**: Common string algorithms (searching, sorting).
- **Practical Exercise**: Implement a program to find the longest substring without repeating characters.

Module 3: Linked Lists

3.1 Singly Linked List

- **Description**: Implementation and operations (insertion, deletion).

- **Learning Outcomes**: Manipulate linked lists effectively.

3.2 Doubly Linked List

- **Description**: Understanding doubly linked lists and their advantages.
- **Practical Exercise**: Create a program to reverse a doubly linked list.

Module 4: Stacks and Queues

4.1 Stack Operations

- **Description**: Implementing stack using arrays and linked lists.
- **Learning Outcomes**: Use stacks for expression evaluation.

4.2 Queue Operations

- **Description**: Implementing queue using arrays and linked lists.
- **Practical Exercise**: Create a program to simulate a ticket booking system using queues.

Module 5: Trees

5.1 Binary Trees

- **Description**: Understanding binary trees and their traversal methods (inorder, preorder, postorder).
- **Learning Outcomes**: Implement tree data structures.

5.2 Binary Search Trees (BST)

- **Description**: Properties and operations of BSTs.
- **Practical Exercise**: Create a program to perform search, insert, and delete operations in a BST

Module 6: Graphs

6.1 Graph Representations

- **Description**: Adjacency matrix vs. adjacency list.
- **Learning Outcomes**: Represent graphs using different methods.

6.2 Graph Traversal Algorithms

- **Description**: Implementing BFS and DFS algorithms.
- **Practical Exercise**: Create a program to find the shortest path in an unweighted graph using BFS.

Module 7: Sorting Algorithms

7.1 Common Sorting Algorithms

- **Description**: Overview of bubble sort, selection sort, insertion sort, quicksort, and mergesort.
- **Learning Outcomes**: Implement and analyze sorting algorithms.

7.2 Practical Exercise

- Compare the performance of different sorting algorithms on large datasets.

Module 8: Searching Algorithms

8.1 Linear Search

- **Description**: Implementing linear search and its complexity.
- **Learning Outcomes**: Use linear search for unsorted data.

8.2 Binary Search

- **Description**: Implementing binary search on sorted arrays.
- **Practical Exercise**: Create a program to find an element in a sorted array using binary search.