Beginner Programming Curriculum

Month 1: Introduction and Basics

Week 1: Introduction to Programming

- **Lesson 1.1:** What is Programming?
  - Overview of programming concepts and terminology.
- **Lesson 1.2:** Introduction to Your Chosen Language (Java, C/C++, C#, Python)
  - Basics of syntax and structure.
- **Project 1.1: Hello World Program**
  - **Step 1:** Write a program to print "Hello, World!".
  - **Step 2:** Understand the structure of the program.
  - **Step 3:** Run and troubleshoot the program.
- **Project 1.2: Basic Calculator**
  - **Step 1:** Create a program to perform basic arithmetic operations.
  - **Step 2:** Implement functions for addition, subtraction, multiplication, and division.
  - **Step 3:** Test and validate the calculator.
- **Project 1.3: Temperature Converter**
  - **Step 1:** Design a program to convert temperatures between Celsius and Fahrenheit.
  - **Step 2:** Implement conversion functions.
  - **Step 3:** Test with different temperature inputs.

Week 2: Variables and Data Types

- **Lesson 2.1:** Understanding Variables
  - Declaration, initialization, and types.
- **Lesson 2.2:** Basic Data Types (int, float, char, etc.)
  - Usage and operations.
- **Project 2.1: Simple Age Calculator**
  - **Step 1:** Create a program to calculate age based on birth year.
  - **Step 2:** Implement input and calculation logic.
  - **Step 3:** Display the result.
- **Project 2.2: Basic Unit Converter**
  - **Step 1:** Design a program to convert units (e.g., meters to kilometers).
  - **Step 2:** Implement conversion functions.
  - **Step 3:** Test with different unit values.
- **Project 2.3: Simple Banking System**
  - **Step 1:** Implement a program to manage basic bank account operations.
  - **Step 2:** Create functions to deposit and withdraw funds.

    - **Step 3:** Display account balance.

Week 3: Control Structures

- **Lesson 3.1:** Conditional Statements (if, else, switch)
    - Syntax and usage.
- **Lesson 3.2:** Loops (for, while, do-while)
    - Iteration and control.
- **Project 3.1: Grade Checker**
    - **Step 1:** Implement a program to determine grades based on scores.
    - **Step 2:** Use conditional statements to assign grades.
    - **Step 3:** Display the result.
- **Project 3.2: Number Guessing Game**
    - **Step 1:** Create a program where the user guesses a number.
    - **Step 2:** Implement loops and conditional statements.
    - **Step 3:** Provide feedback on guesses.
- **Project 3.3: Basic Countdown Timer**
    - **Step 1:** Design a countdown timer using loops.
    - **Step 2:** Implement time decrement and display.
    - **Step 3:** Test with different time intervals.

Week 4: Functions

- **Lesson 4.1:** Defining and Calling Functions
    - Syntax and purpose.
- **Lesson 4.2:** Function Parameters and Return Values
    - Passing data and returning results.
- **Project 4.1: Simple Interest Calculator**
    - **Step 1:** Implement a function to calculate simple interest.
    - **Step 2:** Use parameters for principal, rate, and time.
    - **Step 3:** Display the interest.
- **Project 4.2: Factorial Calculator**
    - **Step 1:** Create a function to compute the factorial of a number.
    - **Step 2:** Implement recursive and iterative solutions.
    - **Step 3:** Test with different inputs.
- **Project 4.3: Fibonacci Sequence Generator**
    - **Step 1:** Design a function to generate Fibonacci numbers.
    - **Step 2:** Implement both iterative and recursive approaches.
    - **Step 3:** Display the sequence.

Month 2: Data Structures

Week 5: Arrays and Lists

- **Lesson 5.1:** Introduction to Arrays
  - Declaring, initializing, and accessing arrays.
- **Lesson 5.2:** Introduction to Lists
  - Basics of lists and their operations.
- **Project 5.1: Array Operations**
  - **Step 1:** Create and manipulate an array of integers.
  - **Step 2:** Implement functions to find maximum, minimum, and average.
  - **Step 3:** Display results.
- **Project 5.2: List-Based To-Do List**
  - **Step 1:** Implement a to-do list using lists.
  - **Step 2:** Create functions to add, remove, and display tasks.
  - **Step 3:** Test with different tasks.
- **Project 5.3: Basic Address Book**
  - **Step 1:** Design an address book using arrays/lists.
  - **Step 2:** Implement functions to add, search, and display contacts.
  - **Step 3:** Handle user input for address management.

Week 6: Basic Sorting and Searching

- **Lesson 6.1:** Introduction to Sorting Algorithms
  - Bubble sort, selection sort.
- **Lesson 6.2:** Introduction to Searching Algorithms
  - Linear search, binary search.
- **Project 6.1: Sorting Numbers**
  - **Step 1:** Implement a program to sort an array of numbers.
  - **Step 2:** Use bubble sort or selection sort algorithms.
  - **Step 3:** Display sorted numbers.
- **Project 6.2: Basic Search Engine**
  - **Step 1:** Create a program to search for a value in an array.
  - **Step 2:** Implement linear and binary search methods.
  - **Step 3:** Display search results.
- **Project 6.3: Simple Contact List Search**
  - **Step 1:** Implement search functionality in a contact list.
  - **Step 2:** Use searching algorithms to find contacts.
  - **Step 3:** Display search results.

Week 7: Introduction to Object-Oriented Programming (OOP)

- **Lesson 7.1:** Basic OOP Concepts
  - Classes, objects, and encapsulation.
- **Lesson 7.2:** Creating and Using Classes
  - Syntax and object management.
- **Project 7.1: Simple Banking System**
  - **Step 1:** Define a `BankAccount` class.
  - **Step 2:** Implement methods for deposit and withdrawal.
  - **Step 3:** Test and validate account operations.
- **Project 7.2: Basic Contact Management System**
  - **Step 1:** Define a `Contact` class with attributes (name, phone number).
  - **Step 2:** Implement methods to manage contacts.
  - **Step 3:** Test and refine the contact management system.
- **Project 7.3: Rectangle Class**
  - **Step 1:** Create a `Rectangle` class with methods for area and perimeter.
  - **Step 2:** Implement getters and setters for dimensions.
  - **Step 3:** Test with different dimensions.

Week 8: More OOP Concepts

- **Lesson 8.1:** Inheritance and Polymorphism
  - Understanding base and derived classes.
- **Lesson 8.2:** Encapsulation and Access Modifiers
  - Managing access levels with private, protected, and public.
- **Project 8.1: Advanced Banking System**
  - **Step 1:** Enhance the banking system with inheritance (e.g., `SavingsAccount`, `CheckingAccount`).
  - **Step 2:** Implement polymorphism for different account types.
  - **Step 3:** Test and refine the system.
- **Project 8.2: Advanced Contact Manager**
  - **Step 1:** Create a base `Contact` class with derived classes (e.g., `PersonalContact`, `BusinessContact`).
  - **Step 2:** Implement inheritance and polymorphism.
  - **Step 3:** Test and integrate features.
- **Project 8.3: Employee Management System**
  - **Step 1:** Develop a base `Employee` class with subclasses (`FullTimeEmployee`, `PartTimeEmployee`).
  - **Step 2:** Implement payroll calculations and employee details.
  - **Step 3:** Test and validate employee management functionalities.

Month 3: String Manipulation

Week 9: Basic String Operations

- **Lesson 9.1:** Introduction to Strings
  - Creating and manipulating strings.
- **Lesson 9.2:** String Methods
  - Using methods like `length`, `substring`, `toUpperCase`, `toLowerCase`.
- **Project 9.1: Name Formatter**
  - **Step 1:** Create a program to format names (capitalize first letters).
  - **Step 2:** Implement string manipulation methods.
  - **Step 3:** Display formatted names.
- **Project 9.2: Basic Text Analyzer**
  - **Step 1:** Implement functions to count characters, words, and sentences.
  - **Step 2:** Use string methods for text analysis.
  - **Step 3:** Display analysis results.
- **Project 9.3: Simple Password Validator**
  - **Step 1:** Design a program to validate passwords based on length and character types.
  - **Step 2:** Implement validation checks.
  - **Step 3:** Test with different passwords.

Week 10: Advanced String Manipulation

- **Lesson 10.1:** Regular Expressions
  - Basics of pattern matching.
- **Lesson 10.2:** String Parsing and Formatting
  - Advanced string operations and formatting.
- **Project 10.1: Regex-Based Validator**
  - **Step 1:** Create a program to validate email addresses using regular expressions.
  - **Step 2:** Implement regex patterns.
  - **Step 3:** Test with various email formats.
- **Project 10.2: Log File Analyzer**
  - **Step 1:** Develop a program to parse and analyze log files.
  - **Step 2:** Implement string parsing and regex matching.
  - **Step 3:** Display log analysis results.
- **Project 10.3: Simple Text-Based Game**
  - **Step 1:** Design a text-based game with string manipulation.
  - **Step 2:** Implement game logic and user interaction.
  - **Step 3:** Test and refine the game.

Week 11: Introduction to File Handling

- **Lesson 11.1:** Basic File Operations
  - Reading from and writing to files.
- **Lesson 11.2:** Handling File Exceptions
  - Managing file-related errors.
- **Project 11.1: Basic Note-Taking Application**
  - **Step 1:** Implement a program to save and load notes.
  - **Step 2:** Use file operations for storing notes.
  - **Step 3:** Test file handling.
- **Project 11.2: Simple Data Logger**
  - **Step 1:** Create a data logger to append and read log entries.
  - **Step 2:** Implement file operations and error handling.
  - **Step 3:** Test and validate data logging.
- **Project 11.3: To-Do List with File Storage**
  - **Step 1:** Enhance the to-do list application to use file storage.
  - **Step 2:** Implement save and load functionalities.
  - **Step 3:** Test file operations.

Week 12: Review and Integration

- **Lesson 12.1:** Review of Concepts
  - Recap of key topics covered.
- **Project 12.1: Comprehensive Address Book**
  - **Step 1:** Design an address book with advanced features.
  - **Step 2:** Implement file handling and OOP concepts.
  - **Step 3:** Test and refine the application.
- **Project 12.2: Mini Project: Personal Finance Manager**
  - **Step 1:** Develop an application integrating multiple concepts.
  - **Step 2:** Implement features for managing finances and transactions.
  - **Step 3:** Test and refine the application.
- **Project 12.3: Final Integration Project**
  - **Step 1:** Create a final project that combines all learned concepts.
  - **Step 2:** Present and review the project.
  - **Step 3:** Complete a self-assessment.

Month 4: Intermediate Topics and Advanced Concepts

Week 13: Advanced OOP Concepts

- **Lesson 13.1:** Understanding Abstract Classes and Interfaces
  - Defining and implementing abstract classes and interfaces.
- **Lesson 13.2:** Exception Handling
  - Basics of try-catch blocks and custom exceptions.
- **Project 13.1: Abstract Class Example**
  - **Step 1:** Create an abstract class with abstract methods.
  - **Step 2:** Implement derived classes and override methods.
  - **Step 3:** Test abstract class functionality.
- **Project 13.2: Exception Handling in Banking System**
  - **Step 1:** Enhance the banking system with exception handling.
  - **Step 2:** Implement custom exceptions for invalid operations.
  - **Step 3:** Test error handling scenarios.
- **Project 13.3: Interface-Based Calculator**
  - **Step 1:** Design a calculator using interfaces.
  - **Step 2:** Implement different operations as interface methods.
  - **Step 3:** Test and refine the calculator.

Week 14: Collections and Generics

- **Lesson 14.1:** Introduction to Collections
  - Lists, Sets, and Maps.
- **Lesson 14.2:** Using Generics
  - Understanding and implementing generics.
- **Project 14.1: Contact List with Collections**
  - **Step 1:** Implement a contact list using collections (e.g., ArrayList).
  - **Step 2:** Implement add, remove, and search functionalities.
  - **Step 3:** Test with different contact entries.
- **Project 14.2: Simple Task Manager with Generics**
  - **Step 1:** Create a task manager using generics.
  - **Step 2:** Implement task management features (add, remove, list).
  - **Step 3:** Test and validate task management.
- **Project 14.3: Data Storage System**
  - **Step 1:** Design a data storage system using collections.
  - **Step 2:** Implement data management functionalities.
  - **Step 3:** Test and refine the system.

Week 15: File Handling and Serialization

- **Lesson 15.1:** File Serialization
  - Basics of serializing and deserializing objects.
- **Lesson 15.2:** Working with Different File Formats
  - Handling JSON, XML, and CSV.
- **Project 15.1: Serialized Object Storage**
  - **Step 1:** Implement a program to serialize and deserialize objects.
  - **Step 2:** Create a file-based storage system.
  - **Step 3:** Test object persistence.
- **Project 15.2: JSON Data Handler**
  - **Step 1:** Design a program to read and write JSON data.
  - **Step 2:** Implement JSON parsing and formatting.
  - **Step 3:** Test with various JSON files.
- **Project 15.3: XML Data Storage**
  - **Step 1:** Create a program to handle XML data.
  - **Step 2:** Implement XML parsing and generation.
  - **Step 3:** Test XML data handling.

Week 16: Advanced Data Structures

- **Lesson 16.1:** Introduction to Advanced Data Structures
  - Trees, Graphs, and Hash Tables.
- **Lesson 16.2:** Implementing and Using Advanced Data Structures
  - Basics of implementation and usage.
- **Project 16.1: Simple Tree Structure**
  - **Step 1:** Implement a basic tree structure.
  - **Step 2:** Create functionalities for adding and traversing nodes.
  - **Step 3:** Test tree operations.
- **Project 16.2: Graph Representation**
  - **Step 1:** Design a graph representation.
  - **Step 2:** Implement graph traversal algorithms (e.g., DFS, BFS).
  - **Step 3:** Test graph functionalities.
- **Project 16.3: Hash Table Implementation**
  - **Step 1:** Create a basic hash table.
  - **Step 2:** Implement hashing functions and collision handling.
  - **Step 3:** Test hash table operations.

Month 5: Advanced Programming Concepts

Week 17: Algorithm Design and Analysis

- **Lesson 17.1:** Basics of Algorithm Design
  - Understanding algorithm complexity and performance.
- **Lesson 17.2:** Common Algorithms and Their Use Cases
  - Sorting, searching, and optimization algorithms.
- **Project 17.1: Sorting Algorithm Comparisons**
  - **Step 1:** Implement and compare different sorting algorithms.
  - **Step 2:** Analyze performance and efficiency.
  - **Step 3:** Display and compare results.
- **Project 17.2: Search Algorithm Analysis**
  - **Step 1:** Implement and compare different search algorithms.
  - **Step 2:** Analyze performance and efficiency.
  - **Step 3:** Test with various data sets.
- **Project 17.3: Basic Optimization Problem**
  - **Step 1:** Design a program to solve an optimization problem.
  - **Step 2:** Implement and test optimization algorithms.
  - **Step 3:** Analyze and present results.

Week 18: Introduction to Multithreading

- **Lesson 18.1:** Basics of Multithreading
  - Understanding threads and concurrency.
- **Lesson 18.2:** Implementing Multithreading
  - Creating and managing threads.
- **Project 18.1: Simple Multithreaded Application**
  - **Step 1:** Create a basic multithreaded application.
  - **Step 2:** Implement thread management and synchronization.
  - **Step 3:** Test and refine the application.
- **Project 18.2: Concurrent Data Processing**
  - **Step 1:** Design an application for concurrent data processing.
  - **Step 2:** Implement multithreading to process data in parallel.
  - **Step 3:** Test and validate performance.
- **Project 18.3: Multithreaded Game Simulation**
  - **Step 1:** Develop a simple game simulation with multiple threads.
  - **Step 2:** Implement game logic and thread management.
  - **Step 3:** Test and refine the simulation.

Week 19: Network Programming Basics

- **Lesson 19.1:** Introduction to Network Programming
  - Understanding basic networking concepts.
- **Lesson 19.2:** Creating Networked Applications
  - Basics of client-server communication.
- **Project 19.1: Simple Chat Application**
  - **Step 1:** Implement a basic chat application using sockets.
  - **Step 2:** Create server and client programs.
  - **Step 3:** Test and refine communication.
- **Project 19.2: Basic File Transfer Application**
  - **Step 1:** Design a program to transfer files over a network.
  - **Step 2:** Implement file sending and receiving functionalities.
  - **Step 3:** Test with different file sizes.
- **Project 19.3: Networked Game**
  - **Step 1:** Develop a basic networked game.
  - **Step 2:** Implement network communication for game state updates.
  - **Step 3:** Test and refine the game.

Week 20: Review and Final Integration

- **Lesson 20.1:** Review of Advanced Topics
  - Recap of advanced concepts and projects.
- **Project 20.1: Final Capstone Project**
  - **Step 1:** Develop a comprehensive application integrating all learned concepts.
  - **Step 2:** Implement advanced features and functionality.
  - **Step 3:** Present and review the final project.
- **Project 20.2: Portfolio and Resume Preparation**
  - **Step 1:** Compile projects and experiences into a portfolio.
  - **Step 2:** Prepare a resume highlighting skills and projects.
  - **Step 3:** Review and finalize the portfolio and resume.
- **Project 20.3: Mock Interviews and Skill Assessment**
  - **Step 1:** Participate in mock interviews to assess skills.
  - **Step 2:** Review and refine responses.
  - **Step 3:** Final self-assessment and goal setting.

Month 6: Additional Topics and Industry Readiness

Week 21: Introduction to Databases

- **Lesson 21.1:** Basics of Databases
  - Understanding relational databases and SQL.
- **Lesson 21.2:** Database Operations
  - CRUD operations (Create, Read, Update, Delete).
- **Project 21.1: Simple Database Application**
  - **Step 1:** Design a simple database schema.
  - **Step 2:** Implement CRUD operations using SQL.
  - **Step 3:** Test and validate database functionality.
- **Project 21.2: Basic Inventory System**
  - **Step 1:** Create an inventory management system with a database.
  - **Step 2:** Implement inventory tracking and reporting.
  - **Step 3:** Test and refine the system.
- **Project 21.3: User Authentication System**
  - **Step 1:** Develop a user authentication system with a database.
  - **Step 2:** Implement registration, login, and password management.
  - **Step 3:** Test and validate authentication features.

Week 22: Introduction to Web Development

- **Lesson 22.1:** Basics of Web Development
  - Understanding HTML, CSS, and basic JavaScript.
- **Lesson 22.2:** Creating a Simple Web Page
  - Designing and coding a static web page.
- **Project 22.1: Personal Portfolio Website**
  - **Step 1:** Design and implement a personal portfolio website.
  - **Step 2:** Use HTML, CSS, and basic JavaScript for interactivity.
  - **Step 3:** Test and refine the website.
- **Project 22.2: Basic Blog Application**
  - **Step 1:** Develop a simple blog application with static content.
  - **Step 2:** Implement features for creating and displaying blog posts.
  - **Step 3:** Test and validate blog functionalities.
- **Project 22.3: Web Form with Validation**
  - **Step 1:** Design a web form with input validation.
  - **Step 2:** Implement client-side validation using JavaScript.
  - **Step 3:** Test and refine form functionalities.

Week 23: Introduction to Version Control

- **Lesson 23.1:** Basics of Version Control
  - Understanding Git and GitHub.
- **Lesson 23.2:** Using Git for Project Management
  - Basic Git commands and workflows.
- **Project 23.1: Git-Based Project Repository**
  - **Step 1:** Initialize a Git repository for a project.
  - **Step 2:** Implement version control for project files.
  - **Step 3:** Test and manage project versions.
- **Project 23.2: Collaborative Project Management**
  - **Step 1:** Collaborate on a project using GitHub.
  - **Step 2:** Implement branching, merging, and pull requests.
  - **Step 3:** Test and manage collaborative contributions.
- **Project 23.3: Version Control for Portfolio**
  - **Step 1:** Use Git to manage your portfolio project.
  - **Step 2:** Implement version control for ongoing updates.
  - **Step 3:** Test and review version history.

Week 24: Final Review and Industry Preparation

- **Lesson 24.1:** Comprehensive Review
  - Recap of all concepts and projects.
- **Project 24.1: Final Capstone Project**
  - **Step 1:** Develop a comprehensive project incorporating all skills learned.
  - **Step 2:** Implement advanced features and demonstrate proficiency.
  - **Step 3:** Present and review the final project.
- **Project 24.2: Job Preparation**
  - **Step 1:** Finalize portfolio and resume.
  - **Step 2:** Prepare for job interviews and technical assessments.
  - **Step 3:** Set career goals and next steps.
- **Project 24.3: Networking and Industry Engagement**
  - **Step 1:** Engage with industry professionals and communities.
  - **Step 2:** Participate in networking events and online forums.
  - **Step 3:** Explore job opportunities and internships.