Complete Roadmap for C Programming with Project.

-SolitaryHorkos

C Programming Roadmap – Expert Level (Mastery)

🎯 Goals:

- ✅ Master system programming, memory management, and concurrency.
- ✅ Deep dive into advanced data structures and algorithms.
- ✅ Learn operating system concepts and kernel development.
- ✅ Explore embedded systems and real-time programming.
- ✅ Gain expertise in performance optimization and security.
- ✅ Work with networking, cryptography, and reverse engineering.
- ✅ Contribute to large-scale C projects and open-source development.

🔰 Phase 1: Advanced Data Structures & Algorithms

📌 Advanced Data Structures

- 🔹 AVL trees and Red-Black trees.
- 🔹 Heaps (Min Heap, Max Heap).
- 🔹 Hash tables and hash functions.

📌 Advanced Algorithms

- 🔹 Dynamic programming (Memoization & Tabulation).
- 🔹 Greedy algorithms.
- 🔹 Backtracking (Sudoku solver, N-Queens problem).

📌 Algorithm Optimization

- 🔹 Profiling and benchmarking (gprof, perf).
- 🔹 Writing cache-friendly code for performance.

🔰 Phase 2: System Programming & Operating Systems

Complete Roadmap for C Programming with Project.

-SolitaryHorkos

📌 Operating System Concepts
- 🔹 Processes, threads, and inter-process communication (IPC).
- 🔹 Memory management (paging, segmentation).
- 🔹 System calls (fork(), exec(), wait()).

📌 Low-Level System Programming
- 🔹 Working with signals (kill, signal, sigaction).
- 🔹 Writing daemon processes.
- 🔹 Low-level file operations (open(), read(), write()).

📌 Concurrency & Multi-threading
- 🔹 Advanced POSIX threads (Pthreads).
- 🔹 Synchronization techniques (mutexes, semaphores, condition variables).
- 🔹 Lock-free programming strategies.

🔰 Phase 3: Embedded Systems & Kernel Development

📌 Embedded C Programming
- 🔹 Programming microcontrollers (AVR, ARM).
- 🔹 Real-time operating systems (RTOS).
- 🔹 Interfacing C with hardware (GPIO, I2C, SPI).

📌 Kernel Development
- 🔹 Writing Linux kernel modules.
- 🔹 Device driver development (character device drivers).
- 🔹 Understanding Linux kernel internals.

📌 Real-Time Systems
- 🔹 Hard vs. soft real-time constraints.
- 🔹 Implementing priority-based scheduling.

Complete Roadmap for C Programming with Project.

-SolitaryHorkos

🔰 Phase 4: Security & Cryptography

📌 Secure Coding Practices

- 🔹 Preventing buffer overflows and memory corruption.
- 🔹 Secure input handling and validation.
- 🔹 Writing sandboxed applications.

📌 Cryptography

- 🔹 Encryption & decryption techniques.
- 🔹 Implementing hashing algorithms (SHA, MD5).
- 🔹 Working with OpenSSL for secure communication.

📌 Reverse Engineering

- 🔹 Understanding binary exploitation.
- 🔹 Debugging compiled binaries (gdb, objdump).
- 🔹 Writing disassemblers and decompilers.

🔰 Phase 5: Advanced Networking & Protocol Implementation

📌 Network Programming in C

- 🔹 Raw sockets and socket programming.
- 🔹 TCP/UDP client-server communication.
- 🔹 Implementing network protocols (HTTP, FTP).

📌 Multi-threaded Network Programming

- 🔹 Creating concurrent servers using threads.
- 🔹 Load balancing and socket multiplexing (select(), poll()).

Complete Roadmap for C Programming with Project.

-SolitaryHorkos

📌 Custom Protocol Implementation
- 🔹 Implementing a simple messaging protocol.
- 🔹 Writing packet sniffers and network analyzers.

🔰 Phase 6: Compiler & Assembly Integration

📌 Writing a Compiler
- 🔹 Lexical analysis and tokenization.
- 🔹 Parsing and syntax trees.
- 🔹 Code generation and optimization.

📌 Assembly Integration
- 🔹 Inline assembly in C (__asm__).
- 🔹 Interfacing C with x86 and ARM assembly.

📌 Custom Memory Management
- 🔹 Implementing memory allocators (malloc(), free()).
- 🔹 Custom garbage collection techniques.

🔰 Phase 7: Cross-Platform Development & Open-Source Contribution

📌 Cross-Platform C Development
- 🔹 Writing portable C code.
- 🔹 Handling OS-specific functionality (#ifdef, #ifndef).

📌 Contributing to Open Source
- 🔹 Understanding large codebases (Linux kernel, GCC).
- 🔹 Submitting patches and collaborating with open-source communities.

Complete Roadmap for C Programming with Project.

-SolitaryHorkos

📌 Design Patterns & Code Architecture

🔹 Applying design patterns in C.

🔹 Writing maintainable and modular codebases.

🏆 Final Skills to Master:

✅ Writing high-performance, secure, and optimized C code.

✅ Developing system-level and real-time applications.

✅ Understanding and contributing to open-source C projects.

✅ Working with embedded systems, networking, and cryptography.

✅ Mastering debugging and reverse engineering techniques.

🎯 PROJECTS:

- **Custom Compiler** – Build a simple C compiler.
- **Operating System Kernel (MiniOS)** – Implement a basic kernel.
- **Blockchain Implementation** – Develop a simple blockchain in C.
- **Keylogger (Ethical Use Only)** – Capture keystrokes for security testing.
- **AI-Based Chatbot** – Implement a chatbot using C.
- **Web Browser in C** – A minimal browser using sockets.
- **Linux Shell (Command Interpreter)** – Build your own shell like Bash
- **Machine Learning Library in C** – Implement basic ML algorithms.
- **Game Engine in C** – Build a basic 2D game engine.
- **Custom File System** – Implement your own file system in C.
- **Minimal Operating System** – Build a minimal operating system.
- **Custom Programming Language** – Design and implement a new programming language.
- **Real-Time Chat Application** – A chat app with real-time messaging.
- **Kernel Module** – Write a Linux kernel module for a custom device driver.

Complete Roadmap for C Programming with Project.

-SolitaryHorkos

- **Embedded System Project** – Build a project using Arduino or Raspberry Pi.
- **Distributed System** – Build a distributed file system.
- **Security Tool** – Create a tool for vulnerability scanning or encryption.
- **Linux Kernel Module** – Develop a loadable kernel module that extends kernel functionality.
- **Embedded System Firmware** – Write firmware for a microcontroller, interfacing with sensors or actuators.
- **Custom Dynamic Memory Allocator** – Implement malloc, free, and realloc with fragmentation handling.
- **High-performance Multi-threaded Server** – Build a server using non-blocking I/O and thread pools.
- **Mini-Compiler/Interpreter** – Create a compiler or interpreter for a domain-specific language.
- **Peer-to-Peer (P2P) Chat Application** – Develop a P2P chat system with advanced socket programming.
- **Device Driver Development** – Write a Linux device driver for custom hardware.
- **Real-time Data Acquisition System** – Capture, process, and log data from hardware sensors.
- **Performance Profiler/Debugger Tool** – Build a tool for monitoring memory usage and profiling functions.
- **Game Engine Core** – Develop core modules like event handling and rendering using SDL/OpenGL.
- **Multi-threaded Web Server** – Implement a concurrent web server.
- **Network Packet Sniffer** – Capture and analyze network packets.
- **Custom Operating System Kernel Module** – Create a simple kernel module.
- **Advanced Data Structure Library** – Implement a comprehensive data structure library.

Complete Roadmap for C Programming with Project.

-SolitaryHorkos

- **High-Performance Sorting Algorithm** – Implement and optimize a complex sorting algorithm.
- **Real-time System Simulation** – Simulate a real-time system.
- **Custom Database Engine** – Create a basic database engine.
- **Game Engine Component** – Develop a specific component like a physics engine.
- **Compiler/Interpreter Component** – Implement a specific compiler component, such as a lexer or parser.