C Programming

Duration: 35 hours

Prerequisites: Prior experience with a scripting or programming language is required.

Course Description: This hands on C programming course provides a comprehensive introduction to the ANSI C language, emphasizing portability and structured design. Students are introduced to all major language elements including fundamental data types, flow control, and standard function libraries. Thorough treatment is given to the topics of string and character manipulation, dynamic memory allocation, standard I/O, macro definition, and the C runtime library. The course explains the use of aggregate structures, unions, and pointers early on so the students can practice extensively in the hands on labs. Structured programming constructs and varargs functions are also covered. Emphasis is given to the processing of command line arguments and environment variables so students will be able to write flexible, user-friendly programs. The course also includes coverage of portability tips drawn from experienced programmers working in production environments. Comprehensive hands on exercises are integrated throughout to reinforce learning and develop real competency.

Students Will Learn

- C Language Basics and Development Environment
- Fundamental Data Types, Storage Classes and Scope
- typedef and Macros
- Operators and Expressions
- Flow Control Constructs
- Basic Character and Formatted I/O
- Pointers and Dynamic Allocation
- Functions
- Arrays
- Strings and Character Manipulation
- Structures
- Advanced Structures and Unions
- Accessing Command Line Arguments and Environment Variables
- Advanced Macro Design
- C Runtime Library Standard I/O Functions

Overview

Overview of C

- Operating System Independence
- Design Goals and Capabilities
- Flavors of C

Fundamental Data Types, Storage Classes, and Scope

- Fundamental Data Types and Qualifiers
- Constants and Strings

Compiler Directives and the C Preprocessor

- Compile-Time Directives
- Use of `typedef`
- C Preprocessor Syntax

Pointers and Dynamic Allocation

- Advantages of Pointers
- User of Pointers

- Storage Classes
- Scope and Block Structure
- Scope and Data Hiding
- Data Initialization

Macros

- Functions vs. Inlining
- Purpose of Macros
- Use of Macros
    - Making Code More Readable
    - Auto Adjustment of Compile Time Values
    - Conditional Compilation
    - Making Code Portable
    - Simplifying Complex Access Calculations
- Advanced Micro Design Tips
- Using Macros to Help Write Portable Programs
- When to Use a Macro instead of a Function
- Using Macros for Debugging

Basic Formatted I/O

- Standard I/O Library
- Character Set Encoding
- Standard Input and Output
- Character I/O Functions
- Formatted I/O Functions
- String Constants

Operators and Expressions

- Arithmetic, Logical, and Bit Operators
- Precedence and Associativity
- Assignment and Casting
- The Conditional Operator

Functions (Subroutines)

- Purpose of Functions
- Functions vs. Inlining
- Automatic Variables
- The Argument Stack
- Passing By Value
- Passing By Reference
- Declaring External Functions
- Function Prototyping
- ANSI Prototyping
- The `_NO_PROTO` Compiler Symbol
- Varargs Functions
- Passing a Function as an Argument
- Designing Functions for Reusability
- Calling a Function from Another Language
- Returning a Dynamically Allocated Value Using Double Indirection
- Pointer and Address Arithmetic
- Dynamic Storage Allocation
- `sizeof` Operator
- Double Indirection

Arrays

- Purpose of Arrays
- Declaring an Array
- Initializing an Array
- Addressing Elements
- Stepping Through an Array
- Variable Size Arrays
- Arrays of Pointers
- Arrays of Strings
- Passing an Array to a Function
- Dynamic Memory Allocation
- Multidimensional Arrays

Program Debugging

- Problem Analysis
- Instrumenting with `printif`
- Instrumenting with `ctrace`
- The Purpose of Debuggers
- How Not to Use Debuggers
- Symbolic Debuggers

Flow Control Constructs

- Conditional Constructs: `if`, `switch`
- Looping Constructs: `while`, `do`, `for`
- Programming Style

Structures

- Purpose of Structures
- Defining and Declaring Structures
- Accessing Members
- Pointers to Structures
- Dynamic Memory Allocation
- Passing a Structure to a Function
    - As a Pointer
    - Passing the Actual Structure

- Casting the Return Value of a Function
- Recursion and Reentrancy

Advanced Structures and Unions

- Nested Structures
- Arrays of Structures
- Bit Fields
- Unions
- Linked Lists

Strings and Character Manipulation

- Strings as Character Arrays
- String Library Functions
- Reading and Writing Strings

Structured Programming

- Structuring Code for Quality, Reliability, Maintainability
- Designing for Modularity and Reusability

C Runtime Library Standard Functions

- Character I/O
- Unformatted File I/O
- Formatted File I/O
- Math Functions
- Miscellaneous Functions

Accessing Command Line Arguments and Environment Symbols

- `argc` and `argv`
- Parsing Command Line Options
- Accessing the Environment Array

Advanced Programming Consideration

- Writing Portable Code
- Use of Macros
- ANSI C Limits
- Feature Test Macros
- Client/Server Design
- Performance Considerations

Related Bootcamp

Track	Duration	Price
C/C++ Programmer	63 hours	$1,790
UNIX Software Developer	98 hours	$2,685

Contact Us

Address: 6 Hemlock Drive Chelmsford, MA 01824

Phone: 978.250.4983

Mon - Thur: 9 am - 5 pm EST
Fri: 9 am - 4 pm EST

E-mail: info@developer-bootcamp.com

Copyright© 2025 Developer Bootcamp