C Programming

- Data Types, Storage Classes, & Scope
- Pointers, Arrays, and Dynamic Allocation
- Compiler Directives, Pre processor, and Macros
- Operators and Expressions
- Functions and C Runtime Library
- Structures and Unions

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 viragos 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 performed throughout training each day to reinforce key concepts.

Course Prerequisites: Understanding of fundamental programming concepts.

Overview of C

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

Compiler Directives and the C Pre processor

- Compile-Time Directives
- Use of typed of
- C Pre processor Syntax

Fundamental Data Types, Storage Classes, and Scope

- Fundamental Data Types and Qualifiers
- Constants and Strings
- Storage Classes
- Scope and Block Structure
- Scope and Data Hiding
- Data Initialization

Macros

- Functions versus In lining
- 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

Pointers and Dynamic Allocation

- Advantages of Pointers
- User of Pointers
- Pointer and Address Arithmetic
- Dynamic Storage Allocation
- Size of 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 c trace
- The Purpose of Debuggers
- How Not to Use Debuggers
- Symbolic Debuggers

Operators and Expressions

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

Functions (Subroutines)

- Purpose of Functions
- Functions versus 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
- Casting the Return Value of a Function
- Recursion and Re-entrancy

Advanced Structures and Unions

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

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

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

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

Advanced Programming Consideration

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

#1, C.V.R Complex, Singaravelu St, T.Nagar, Chennai - 17,(Behind BIG BAZAAR) Tamilnadu, India.

Mobile : +91 9962 067 067, 9176 499 499 www.spiroita,com, training@spiroita.com