C PROGRAMMING

DURATION:

4 Days

COURSE DESCRIPTION:

This course provides students with a comprehensive study of the C programming language. Classroom lectures stress the strengths of C, which provide programmers with the means of writing efficient, maintainable, and portable code. The lectures are supplemented with non-trivial lab exercises.

WHO SHOULD ATTEND:

This course is for programmers who have had experience in any programming language or have been tasked with a C programming project, and other technical types including managers and customer support engineers who need to know C.

BENEFITS OF ATTENDANCE:

Upon completion of this course, students will be able to:

- Write C programs that are non-trivial.
- Use the variety of data types appropriate to specific programming problems.
- Utilize the modular features of the language.
- Demonstrate efficiency and readability.
- Demonstrate the use of the various control flow constructs.
- Use arrays as part of the software solution.
- Utilize pointers to efficiently solve problems.
- Include the structure data type as part of the solution.
- Create their own data types.
- Use functions from the portable C library.

PREREQUISITES:

Students should have taken the Software Development for Non-Programmers course or have experience with a programming or an assembly language.

COURSE OUTLINE:

- **CHAPTER 1: GETTING STARTED**
  1. What is C?
  2. Background
  3. Sample Program
  4. Components of a C Program
  5. Examples
  6. Data Types
  7. Variables
  8. Naming Conventions for C Variables
  9. Printing and Initializing Variables
  10. Array Examples
  11. Compiling and Executing a C Program
- **CHAPTER 2: FUNCTIONS AND OPERATORS**
  1. Examples of C Functions
  2. Functions
  3. sum Invoked from main
  4. Invoking Functions
  5. Elementary Operators
  6. The operator= Operators
  7. Operators
  8. The Conditional Operator
  9. Increment and Decrement Examples
  10. Increment and Decrement Operators
- **CHAPTER 3: CONTROL FLOW CONSTRUCTS**
  1. Examples of Expressions
  2. if
  3. if else
  4. while
  5. for
  6. Endless Loops
  7. do while
  8. break and continue
  9. switch
  10. else if
- **CHAPTER 4: THE C PREPROCESSOR**

1. #define
2. Macros
3. #include
4. Conditional Compilation
5. #ifdef
6. #ifndef

- **Chapter 5: Simple I/O**
  1. Character I/O
  2. End of File
  3. Simple I/O Examples
  4. Simple I/O Redirection
  5. I/O with Character Arrays
- **Chapter 6: More on Functions**
  1. General
  2. Function Declarations
  3. Returning a Value or Not
  4. Function Prototypes
  5. Arguments and Parameters
  6. Organization of C Source Files
  7. Extended Example
  8. The getline Function
  9. The strcmp Function
  10. The check Function
  11. The atoi Function
  12. The average Function
  13. Summary
- **Chapter 7: Bit Manipulation**
  1. Defining the Problem Space
  2. A Programming Example
  3. Bit Wise Operators
  4. Bit Manipulation Functions
  5. Circular Shifts
- **Chapter 8: Strings**
  1. Fundamental Concepts
  2. Aggregate Operations
  3. String Functions
- **Chapter 9: Higher Dimensional Arrays**
  1. Array Dimensions
  2. An Array as an Argument to a Function
  3. String Arrays
- **Chapter 10: Separate Compilation**
  1. Compiling Over Several Files
  2. Function Scope
  3. File Scope
  4. Program Scope
  5. Local static

6. register and extern
7. Object Files
8. Libraries
9. The C Loader
10. Header Files

- **Chapter 11: Pointers (Part 1)**
    1. Fundamental Concepts
    2. Pointer Operators and Operations
    3. Changing an Argument with a Function Call
    4. Pointer Arithmetic
    5. Array Traversal
    6. String Functions with Pointers
    7. Pointer Difference
    8. Prototypes for String Parameters
    9. Relationship Between an Array and a Pointer
    10. The Pointer Notation *p++
- **Chapter 12: Pointers (Part 2)**
    1. Dynamic Storage Allocation - malloc
    2. Functions Returning a Pointer
    3. Initialization of Pointers
    4. gets - a Function Returning a Pointer
    5. An Array of Character Pointers
    6. Two Dimensional Arrays vs. Array of Pointers
    7. Command Line Arguments
    8. Pointers to Pointers
    9. Practice with Pointers
    10. Function Pointers
- **Chapter 13: Structures**
    1. Fundamental Concepts
    2. Describing a Structure
    3. Creating Structures
    4. Operations on Structures
    5. Functions Returning Structures
    6. Passing Structures to Functions
    7. Pointers to Structures
    8. Array of Structures
    9. Functions Returning a Pointer to a Structure
- **Chapter 14: Structure Related Items**
    1. typedef - New Name for an Existing Type
    2. Bit Fields
    3. unions
    4. Non-Homogeneous Arrays
    5. Enumerations
- **Chapter 15: File I/O**
    1. System Calls vs. Library Calls
    2. Opening Disk Files

3. fopen
4. I/O Library Functions
5. Copying a File
6. Character Input vs. Line Input
7. scanf
8. printf
9. fclose
10. Servicing Errors - errno.h
11. feof

- **Chapter 16: Information About Files**
    1. The stat Function
    2. File Existence
    3. Telling Time - time and ctime
    4. Telling Time – localtime
- **Chapter 17: I/O With Structures**
    1. A Database Application
    2. The menu Function
    3. The fwrite Function
    4. The create_db Function
    5. The fread Function
    6. The print_db Function
    7. fseek
    8. The retrieve_db Function
    9. fflush and ftell
- **Chapter 18: Useful Library Functions**
    1. strstr
    2. strchr, strrchr
    3. system
    4. strtok
    5. strspn, strcspn
    6. Math Functions
    7. Character Testing Functions
    8. exit and atexit
    9. signal
    10. memcpy and memset
    11. qsort
    12. Binary Search - bsearch
- **Appendix A: C Language Programming**
    1. Important Header Files
    2. printf Formats
    3. C Reserved Words
    4. Conversion
    5. Precedence Chart