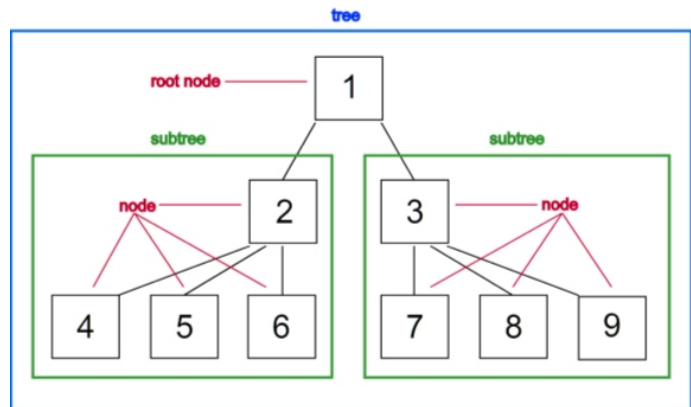


C & C++



Corporate Trainer's Profile

Corporate Trainers are having the experience of 4 to 12 years in development , working with TOP CMM level 5 companies (Project Leader /Project Manager) qualified from NIT/IIT/IIM and work exp in USA and UK.



Capability Maturity Model level Project Standard*** :-

The Capability Maturity Model (CMM) is a method for evaluating the maturity of organizations on a scale of 1 to 5.

Get the Opportunities to work on Client Projects Of US/UK, which follow the all standard of CMM level 5 Company.

Projects



Advanced C Programming for Embedded Systems A 5-day Course

Synopsis

In this course you'll learn advanced programming techniques using dynamic data structures and algorithms. Finite state machines and statecharts and their uses are covered in depth. You'll also learn techniques specific to embedded systems, such as structured interrupt handling, implementation of simple schedulers, and how to work with embedded operating systems. You'll learn the standard inter-process communication mechanisms and their uses. In addition the course overviews various advanced algorithms and discusses research oriented approaches for learning more about these algorithms and their implementation. The course will also show you how to design modules with clean interfaces to achieve disciplined and well structured code. The course emphasises the implementation of disciplined and well structured code and the design of modules with clean interfaces. Lab exercises are used to consolidate key concepts. The course uses PIC16, PIC18, and PIC24 / dsPIC processors as a platform for hands-on work. However, the techniques taught can be readily applied to other architectures. Similarly, the course uses uCOSII and CMX as example operating systems, but Inter-Process Communication techniques are covered in a generic way so that they can be applied to other architectures and operating systems.

Course Objectives

Students completing this course will considerably improve the discipline and rigour with which they design and write embedded systems applications in C. You'll be able to implement classical data structures such as circular buffers, linked lists, and trees -- and you'll know when it's appropriate to use them. You'll be exposed to a variety of advanced programming idioms and algorithms with their associated data structures, for tasks such as indexing, data compression and error detection. You'll learn to write event driven programs, to implement Finite State Machines, and to design hierarchical state machines using statecharts. You'll learn structured programming techniques for implementing multi-tasking applications, and you'll be able to work with embedded operating systems as well as the standard inter-process communication idioms such as producer-consumer, workcrew and monitors.

Key Topics

Advanced use of dynamic data structures and algorithms for manipulating them
Structured interrupt handling
Implementation of simple schedulers and operating systems
Working with embedded operating systems
Understanding standard techniques for inter-process communication, and their uses
Finite state machines, statecharts, and their uses
Advanced algorithms and research-oriented approaches
Prerequisites

Attendees should have some experience of embedded systems programming and a sound basic knowledge of the C language. Course 'Introduction to C Programming for Embedded Systems' provides suitable background.

Delivery

This is instructor led C training. Each section of the material covered by the tutor is followed by hands-on practical exercises for which worked examples of the solutions are typically provided.

Scheduled and On-site Courses

Courses in this subject are scheduled on an 'ad-hoc' basis. We can arrange a course at our Carshalton centre or on customer site for any client wishing to send two or more delegates on the same course.

Contents

An Introduction to C

Intensive overview of essential C concepts and idioms

Data types, data structures, pointers and arrays

Using pointers to search collections of data

Arrays and buffers

Circular buffers

Polygonal buffers

I/O vectors

Linked Lists in depth

Singly linked and doubly linked lists

Using lists to implements FIFO queues and LIFO queues (stacks)

Using lists of linked lists

Using linked list nodes containing void * pointers to implement heterogeneous collections of data

Using linked lists to implement resizeable arrays

Binary trees, their uses and their relations

Basic binary trees

Self-balancing binary trees (AVL, Red-Black, Splay)

Heaps and their uses

Huffman encoding

Priority queues

Error detection

CRC checksums (16 bit and 32 bit)

Implementing simple memory management schemes

Implementing simple flash memory file systems State Machines and Statecharts

Event driven programming

Basic FSMs

Pattern matching

Parsing

State driven hardware and communication protocols

Implementing FSMs using switch statements

Implementing FSMs using a table driven approach

Limitations of FSMs

Extended FSMs and hierarchical FSMs

Extending FSMs by adding variables and conditional transitions

Nesting state machines (push down automata)

Statecharts

Hierachical FSMs and extended FSMs (simple statecharts)

Orthogonal statecharts and concurrency

Active objects - linking multi-tasking, message passing and event driven programming

Basic operating systems and multi-tasking concepts

Task structures

Task life cycle

Task management

Task data structures

Task queues

Message queues

Semaphores (counting, binary, mutex)

Monitors

Pipes

Memory management services

Signals

Timers

Device drivers

Standard Inter-Process Communication Idioms

Producer - Consumer

Monitors

Readers and Writers

Workcrew