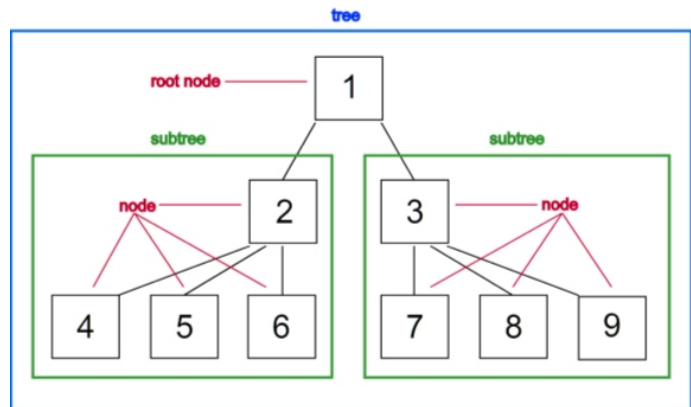


C & C++



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



Introduction to C - A 5-day Course

Synopsis

This is an intensive introduction to ANSI C programming using the GNU C compiler. At the end of the course attendees should be reasonably confident in reading and writing C programs and have a good practical understanding of advanced topics such as the use of C pointers, the relationship between pointers and arrays, as well as dynamic memory allocation and memory management. In addition, they will be able to understand makefiles and version control using RCS and CVS. Throughout, the course will emphasise a disciplined and structured approach to C programming. Debugging and code testing techniques will also be covered extensively, including the use of the GNU debugging tools. The course is not based on any particular Integrated Development Environment (IDE), and adopts a more traditional approach where files are created using a suitable editor (such as EMACS) and programs are compiled either by issuing commands at the commandline, or by invoking make on an appropriate makefile.

Suitable for

Attendees are expected to have some programming experience - e.g. Basic, Fortran or Pascal. The course is also suitable for recent engineering and computer science graduates who need to "cure" any bad (programming) habits they might have picked up at University.

Prerequisites

Drive and determination are important pre-requisites - the course aims to produce competent working programmers, as is a good sense of humour (vital when working under pressure). Good programmers think hard, but also have a strong obsessive streak a need to get a program fully debugged and working at all costs.

Key Skills

Master the GNU C compiler

Learn both the syntax and usage of C

Enter the exotic world of dynamic data structures and algorithms

Be able to save data to files and read data from files

Understand the intricacies of makefiles and make

Appreciate the need for version control and be able to work with RCS and CVS

Master the techniques for writing well structured modular code in C

Know how to use the GNU debugger

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.

The course also includes numerous mini-challenges and code examples for private study, and to build on the skills acquired during the course.

Compiling a simple C program

- Compiling an application containing several C modules (.c files)
- Creating data structures and applications that use them
- Traversing and processing information in one dimensional and multi-dimensional arrays
- Finding your way round the GNU documentation and help manuals
- Reading and writing text and binary files
- Creating and using indexed data structures
- Contents
- First steps
- How C became
- Getting to know the GNU C compiler
- The basic anatomy of a C program
- Data types, operators and expressions (the basics)
- Base data types and their sizes
- Constants and declarations
- Variable names
- Arithmetic operators
- Relational and logical operators
- Increment and decrement operators
- Increment and decrement operators
- Assignment operators
- Expressions
- Operator precedence
- Casting and type conversion
- Program flow control
- if - else
- switch
- while , do-while and for-loops
- Statements and blocks
- Functions
- Functions and function prototypes
- Returning values from functions
- External variables and scope rules
- Static variables
- call by value and recursion
- Pointers and arrays
- Pointers and addresses
- Pointers as arguments to functions
- Arrays
- The relationship of pointers and arrays
- Pointer arithmetic
- How C deals with multi-dimensional arrays
- Arrays of pointers - their uses and initialisation
- Strings and string functions
- Structures
- Fundamentals of structures
- Passing structures as arguments to functions
- Returning structures as return values from functions

Arrays of structures

Pointers to structures

Using pointers to structures to pass values to a function and return values from a function

Typedef and its uses in developing well structured and maintainable code

An introduction to data structures and algorithms

Sorting arrays of records

Allocating and freeing memory (malloc and free)

Linked lists and queues

Indexing (table lookup, binary trees, hashing)

Input-output

Standard input and standard output

Formatted output - printf

Formatted input - scanf

The dangers of scanf

File access and file I/O

An overview of input-output in a windowing environment

Further topics

The MACRO pre-processor and how to use it wisely

Unions

Bit fields

Date and Time functions in the C Standard Library

Diagnostics- the assert macro

Makefiles, Libraries, RCS and CVS

Simple make files

MACROS in make files

Use of dummy targets

Recursive makefiles

Building and using libraries

The importance of version control

RCS and CVS