

# Programming In C

Duration:	4 days
Type:	beginner

## Description

This course provides a comprehensive introduction to programming in the C language. All the features of modern C are covered, as defined by the C99 standard, plus a preview of the new features arriving in C1X. By the end of the course delegates will be able to write programs that efficiently implement algorithms, create and manipulate data structures, manage heap memory and perform file and console based I/O. The Make build tool and GCC compiler suite are incrementally introduced as the course progresses. Extra topics can be introduced as required, including signal handling, threading with Pthreads and networking.

## Prerequisites

The course can be delivered to delegates with no previous programming experience, in which case a 5 day duration is recommended. Typically delegates will have already used one or more of the C family of languages, such as C++, Java, C# or Perl. If the course is delivered in a UNIX environment familiarity with the Korn shell and VI text editor is required.

## List of Modules

### Introduction to C

- The evolution of C and its successors
- Key characteristics of the C language
- Examples of where C should and should not be used

### Structuring C Applications

- Performing basic console I/O
- Declaring and working with variables
- The pre-processor, compiler and linker
- Standard options when using the GCC compilers
- Understanding pre-processor directives
- Placing declarations in header files
- Placing definitions in source files
- Compilation and linkage errors
- Using guards in header files

## Basic Programming in C

- Integer and floating point types
- Enumerations and unions
- Basic arrays and structures
- Operators, expressions and precedence
- Conditional statements and iteration
- Managing control flow with *break* and *continue*
- Understanding and using *goto* safely

## Dependency Management using Make

- Introducing the concept of dependency trees
- The structure of a simple makefile
- Declaring explicit and pseudo rules
- Using patterns in explicit rules
- Declaring and using variables
- Built in rules and variables
- Writing recursive makefiles
- Functions provided by make

## Writing and Calling Functions

- Function prototypes and definitions
- Matching invocations to prototypes
- The purpose and structure of a call stack
- Making use of recursive functions
- Declaring functions as *inline*
- Using variable argument lists via *stdarg.h*

## Pointers and Storage Management

- Storage class specifiers in C
- Understanding variable length arrays
- Declaring and dereferencing pointers
- The *NULL* macro and 0 address
- The purpose of the void pointer type
- Working with pointers to arrays
- Working with pointers to structures
- Practical uses of pointers to pointers
- The *const* keyword and pointers
- Differentiating 'const pointers' and 'pointer to const'
- Declaring pointers to functions
- Simplifying function pointers via *typedef*
- Using function pointers to implement callbacks
- Dynamically allocating memory from the heap
- Handling out of memory conditions

## String Manipulation in C

- Making sense of strings as arrays of *char*
- Understanding how string literals are compiled and used
- Built in functions for manipulating strings
- Using *sprintf* to format output
- Common problems when working with strings
- Support for wide characters and strings

## File Access in C

- Safely opening and closing files
- Performing random access within a file
- Reading and writing plain and formatted data