C 18

Data Types, Functions, Conditionals, Pointers, Functions, I/O, Structures, Pre-processor, DLLs

C is available on virtually every form of computing hardware and for every OS. C libraries are callable from almost every high-level programming language (this is needed because the API to most OSes is in C). Applications seeking close-to-the-metal performance need to be written in C. Most kernel and device driver development is in C. The syntax of custom languages for GPUs (shader languages) and FPGAs are C-like.

Hence every developer should know C and know it well. What you learn on this intensive training course will be applicable on all these target platforms.

In this course all the main C programming concepts are covered. This includes flow control constructs, pointers, functions, the pre-processor and typedefs along with the importance of data-types, type safety and custom structures. Compiling, linking and debugging multi-file applications are covered with demo code that evolves from simple ten-line utilities to large-scale projects involving multiple developers (e.g. libraries & header files). As well as covering the syntax of the C language, we also cover how to use the important C standard library (e.g. I/O) and explain how to use C effectively in app development.

Target Audience

This training course is aimed at software engineers who need to quickly get up to speed developing in C.

Prerequisites

Attendees must have a programming background but no experience of C required.

Contents of One-Day Training Course

Overview of C

C evolution – K&R, ANSI, C99, C11, C18
Whirlwind tour of features of C
Hardware issues (e.g. volatile)

A Simple Program

Freeform language
Data types and variables
Constants, functions, comments, layout

Variables and Data Types

Simple data types, naming and size
Strings - char and wchar
Security and buffer overflow issues
Enumerators
User defined structures and unions
Static and global variables
Visibility of global variables

Intro to C Functions

Splitting features into functions
Small and self-contained chunks of code
A multi-function C program

Operators

Arithmetic & relational operators
'=' and '= ='
Increment and decrement operators
Bitwise operators

Flow of control

if / if else / for loop
switch/case
while
break and continue

Non-Local Jumps

Practical uses of non-local jumps
setjmp and longjmp

Functions in Detail

Function prototypes
Variable arguments lists
Functions in separate files (.c and .h files)

Pointers and Arrays

Handling arrays and array arithmetic
Pointers to standard data & custom structs
Function pointers (defining/setting/calling)
void vs. void *, use of restricted modifier

Pre-processor

#include, #define, #ifdef and #endif
Concatenation/stringizing/varidic macros

Dynamic Memory

malloc, free, realloc; costs of memory ops
Examples of dynamic memory usage

C Standard Library

Standard I/O, file I/O
Assert, math, conversions, time

Compiling, Linking, Debugging

Role of modules and compilation units
Constructing apps and libraries from code in multiple files
Building, linking and debugging

C11 [major]/C18 [minor]

Tour of new features of latest standard
Anonymous structs
New C threads and stdatomics header files

Project

Object-based C - let's build OO in C with inheritance (tree of anonymous structs), encapsulation (redefining structs), exceptions (using setjmp/longjmp), namespaces, RTTI, generics, etc.