

Course Aims:

This course is design to provide non-C programmers with the essential skills and knowledge necessary to allow them to become competent in developing ANSI-compliant C programs suitable for use in embedded applications.

Course Outline:

<i>The History of C Programming</i> <i>The Structure of the C Program:</i> • Compiling a C Program <i>Variables & Constants:</i> • Variable Names; Data Types; Declarations • Assignment Operator; Constants • Initialisation of Variables; Enumeration Types • printf(); getchar() <i>Expressions & Operators</i> • Arithmetic Operators; Expression Evaluation • Increment & Decrement Operators • Bitwise Logical Operators • Compound Assignment Operators • Type Conversion; Comma Operator • Order of Precedence <i>Flow Control of a Program</i> • Types of Flow Control Statements • Logic States; Relational Operators • Boolean Logical Operators; if-else Statement • Nested if Statements • Conditional Operator; Compound Statements • Switch Statement; While loop; Do-while Loop; • For Loop; For Loop Variations • Break & Continue Statements • exit() Statement; goto Statement <i>Arrays</i> • Theory of Arrays; Range Checking; Initialisation • Unsized Array Initialisation <i>The Standard I/O Library</i> • getchar() & putchar(); gets() & puts() • printf(); scanf() <i>Functions</i> • The Definition of a Function; Function Parameters • Non-Returning Functions; Void Functions • The Return Statement • Functions Returning Non-integers • Passing Arrays to Functions; Recursive Functions <i>Scoping</i> • Storage Class Types; Automatic Variables • Register, Static & External Variables • Initialisation Rules for the Storage Classes	<i>Pointers</i> • The & and * Operators • Pointers & Function Arguments • Pointer Arithmetic; Pointers & Arrays • Arrays as Function Arguments • Pointer Initialisation; Arrays of Pointers • Pointer to Pointers; Command Line Arguments • Pointers to Functions <i>File Handling</i> • The File Pointer; fopen() • stdin, stdout & stderr File Pointers; fclose() • fgetc(); fputc(); ungetc(); fgets(); fputs() • fprintf(); fscanf(); fread() & fwrite() • fseek(); ftell(); rewind(); fflush(); File Status <i>Structures & Unions</i> • Members of a Structure Variable • Pointers to Structures • Structure Initialisation & Assignment • Structures & Functions; Nested Structures • Bit Fields; Unions; sizeof(); typedef; <i>Dynamic Memory</i> • Memory Leakage • malloc() & free() <i>The C Preprocessor</i> • The Pre-processor Operation • #include; Defined Constants • Defined Macros; Macro Side Effects • Deleting a Definition; Conditional Compilation • #ifdef & #ifndef; #line <i>Embedded Specifics</i> • Use of pointers • IO Ports • Interrupts <i>Real Time Issues</i> • Scheduling • Concurrency & resource protection • Semaphores • Threads <i>Testing & Debugging Embedded Systems</i> • Emulation • Test harnesses • Target Specific Techniques
--	---

Target Audience:

Systems and applications programmers who will be developing embedded systems in C. Anyone who wants a practical understanding of C will benefit from this course. It is suitable for hardware and software engineers who want to expand their knowledge in a powerful all-purpose language, technical managers who want to manage C programming projects.

Assumed Knowledge:

Participants should have a knowledge of basic programming techniques and preferably at least one programming language.