

**Price:** R8,600.00 excl. VAT  
**Duration:** 5 days  
**Code:** AANSC

# Advanced C Programming

## Description

The Advanced C course provides a deeper look into the standard C programming language. It covers advanced topics such as debugging techniques, algorithms, memory management. The emphasis is always on portability, compiler independence and professionalism.

## Objectives

Delegates who complete the Advanced C Programming course will be able to:

- Understand advanced C programming development techniques.
- Understand and use all types of pointers.
- Use self-referencing structures and link lists.
- Manage memory within C applications.

## Intended Audience

Programmers who have developed in C for at least 6 months, and who need to learn more advanced techniques. It is also useful for engineers developing embedded applications.

## Prerequisites

Our Standard C Programming course or equivalent experience in C programming.

## Course Contents

*The lecturer reserves the right to modify the contents of the course to suit the needs of the delegates.*

**Introduction** • Professionalism. • Portability. • Modularity. • Structured programming. • Coding standards. • Standard library functions. • Non-standard functions. • Under-utilized functions.

**The C compiler** • Technical aspects. • Startup module. • Modifications for embedded / custom systems. • Translation order and tokenization. • Object modules. • Compiler switches.

**The C Preprocessor** • Conditional compilation. • Token concatenation. • Stringization. • Charization. • Compilation units. • Trigraphs. • Advanced macros.

**Expressions, Types and Variables** • Expression evaluation. • Operator association and precedence. • Sequence guarantee points. • Implicit/explicit type conversions. • Literals. • Escape characters. • Type hierarchy. • Derived types. • Bitfields. • Portability.

**Functions, Statements and Program Structure** • Overview. • Stack usage for local variables and parameters. • Function returns. • Function call operator. • Variable number of parameters. • Structured programming vs OOP techniques. • Libraries and make files.

**Arrays and Pointers** • Array names. • Array manipulation. • Multi-dimensional arrays. • Pointer variables. • Indirection. • Dynamic arrays and pointers. • Pointers to structures and functions.

**Memory management** • The C runtime memory. • Stack, heap and static data areas. • Dynamic memory. • Fragmentation of memory. • Memory management techniques.

**Data Structures** • Creating and using structures. • Self-referencing structures. • Linked lists. • Generic data structures. • Portability. • Memory management. • Information hiding.

**Debugging Techniques** • Stabilization: lexical, syntactic, execution and logic errors. • Locating errors: lexical, material and referential proximity. • Debuggers. • Memory initialization. • Structure sentinels. • Stack errors. • Conditional compilation.

**Miscellaneous** • Introduction to numerical methods, encryption, sorting and algorithm evaluation. • Introduction to C++. • Graphical User Interfaces.