Introduction to C Programming

Course Summary

Description

This course uses a practical, problem-solving approach to provide a thorough introduction to programming in the C language. This course is not platform specific, so it can be taught on a variety of platforms, including Windows, UNIX, and Mainframe.

Topics

- Introduction
- Operators in C
- Control flow
- Functions
- Pointers
- Arrays in C
- Structures in C
- Reading C declarations
- Handling files in C
- Miscellaneous things
- C and the heap

Audience

This course is designed for programmers who wish to learn to program in C.

Prerequisites

The student should have a good working knowledge of programming techniques. No prior use of C is assumed, but limited exposure to the product would help.

Duration

Five days

Introduction to C Programming

Course Outline

I. Introduction

A. Welcome to C
B. Course objectives
C. Practical exercises
D. Features of C
E. The history of C
F. Standard C vs. K&R C
G. A C program
H. The format of C
I. Another example
J. Variables
K. printf and scanf
L. Integer types in C
M. Integer example
N. Character example
O. Integers with different bases
P. Real types in C
Q. Real example
R. Constants
S. Warning!
T. Named constants
U. Preprocessor constants
V. Take care with printf and scanf!

II. Operators in C

A. Operators in C
B. Arithmetic operators
C. Using arithmetic operators
D. The cast operator
E. Increment and decrement
F. Prefix and postfix
G. Truth in C
H. Comparison operators
I. Logical operators
J. Bitwise operators
K. Bitwise example
L. Assignment
M. Other assignment operators
N. sizeof operator
O. Conditional expression operator
P. Precedence of operators
Q. Associativity
R. Precedence/associativity table

III. Control Flow

A. Control flow
B. Decisions if then
C. if then else
D. Nesting ifs
E. switch
F. More about switch
G. while loop
H. Semicolon warning!
I. while, not until!
J. do while
K. for loop
L. for is not until either!
M. Stepping with for
N. Extending the for loop
O. break
P. continue

IV. Functions

A. The rules
B. Writing a function
C. Calling a function
D. Prototypes
E. Prototyping is not really optional
F. Writing prototypes
G. Take care with semicolons
H. Example prototypes
I. Example calls
J. Rules of visibility
K. Call by value
L. C and the stack
M. Stack example
N. Storage
O. auto
P. static
Q. register
R. Global variables

Introduction to C Programming

Course Outline (cont'd)

V. Pointers
- A. Pointers
- B. Declaring pointers
- C. Example pointer declarations
- D. The "&" operator
- E. Rules
- F. The "*" operator
- G. Writing down pointers
- H. Initialization warning!
- I. Initialize pointers!
- J. NULL
- K. Fill in the gaps
- L. Type mismatch
- M. Call by value
- N. Reminder
- O. Call by reference
- P. Pointers to pointers

VI. Arrays in C
- A. Declaring arrays
- B. Accessing elements
- C. Array names
- D. Passing arrays to functions
- E. Using pointers
- F. Pointers to go backwards too
- G. Pointers may be subtracted
- H. Using pointers
- I. * and ++
- J. Which notation?
- K. Strings
- L. Printing strings
- M. Null really does mark the end!
- N. Assigning to strings
- O. Pointing to strings
- P. Multidimensional arrays

VII. Structures in C
- A. Concepts
- B. Setting up the template
- C. Creating instances
- D. Initializing instances
- E. Structures within structures
- F. Accessing members
- G. Unusual properties
- H. Instances may be assigned
- I. Passing instances to functions
- J. Pointers to structures
- K. Why (*p).name?
- L. Using p->name
- M. Pass by reference
- N. Returning structure instances
- O. Linked lists
- P. Example
- Q. Printing the list

VIII. Reading C Declarations
- A. Introduction
- B. SOAC
- C. typedef

IX. Handling Files in C
- A. Introduction
- B. Streams
- C. What is a stream?
- D. Why stdout and stderr?
- E. stdin is line buffered
- F. Opening files
- G. Dealing with errors
- H. File access problem
- I. Copying files
- J. Convenience problem
- K. Accessing the command line
- L. Useful routines
- M. Binary files

Introduction to C Programming

Course Outline (cont'd)

X. Miscellaneous Things

A. Unions
B. Remembering
C. Enumerated types
D. Using different constants
E. The preprocessor
F. Including files
G. Pathnames
H. Preprocessor constants
I. Avoid temptation!
J. Preprocessor macros
K. A debugging aid
L. Working with large projects
M. Data sharing example
N. Data hiding example
O. Use header files
P. Getting it right

XI. C and the Heap

A. What is the heap?
B. How much memory?
C. Dynamic arrays
D. Using dynamic arrays
E. calloc/malloc example
F. realloc example
G. realloc can do it all
H. Allocating arrays of arrays
I. Dynamic data structures
J. Linking the list