- Declaring Pointers
- Pointers to Functions
- Pointer to void
- Global and static pointers

Pointer Size and Types

- Memory Models
- Predefined Pointer-Related Types
- Understanding size_t
- Using the sizeof operator with pointers
- Using intptr_t and uintptr_t
- Pointer Arithmetic
- Multiple Levels of Indirection
- Constants and Pointers

Dynamic Memory Allocation

- Memory Leaks
- Using the malloc Function
- Using the calloc Function
- Using the realloc Function

Deallocating Memory Using the free Function

- Assigning NULL to a Freed Pointer
- Double Free
- The Heap and System Memory
- Freeing Memory upon Program Termination

Dangling Pointers

- Dangling Pointer Examples

Dynamic Memory Allocation Technologies

- Garbage Collection in C

- Resource Acquisition Is Initialization
- Using Exception Handlers

Chapter 3 - Pointers and Functions

Program Stack and Heap

- Program Stack
- Organization of a Stack Frame

Passing and Returning by Pointer

- Passing Data Using a Pointer
- Passing a Pointer to a Constant
- Pointers to Local Data
- Passing Null Pointers
- Passing a Pointer to a Pointer

Function Pointers

- Declaring Function Pointers
- Using a Function Pointer
- Passing Function Pointers
- Returning Function Pointers
- Using an Array of Function Pointers
- Comparing Function Pointers
- Casting Function Pointers

Chapter 4 - Pointers and Arrays

Quick Review of Arrays

- One-Dimensional Arrays
- Two-Dimensional Arrays
- Multidimensional Arrays

Pointer Notation and Arrays

Using malloc to Create a One-Dimensional Array

Using the realloc Function to Resize an Array

- Using Array Notation
- Using Pointer Notation

Using a One-Dimensional Array of Pointers

- Allocating Potentially Non-contiguous Memory

String Fundamentals

- String Declaration
- String Initialization
- Comparing Strings
- Copying Strings
- Concatenating Strings

Passing Strings

- Passing a Simple String
- Passing a String to Be Initialized
- Passing Arguments to an Application

Returning Strings

- Returning the Address of a Literal
- Returning the Address of Dynamically Allocated Memory

Function Pointers and Strings

Chapter 6 - Pointers and Structures

Introduction

Structure Deallocation Issues

Avoiding malloc/free Overhead

Using Pointers to Support Data Structures

- Single-Linked List

Pointer Declaration and Initialization

- Improper Pointer Declaration
- Failure to Initialize a Pointer Before It Is Used
- Dealing with Uninitialized Pointers

Pointer Usage Issues

- Test for NULL
- Misuse of the Dereference Operator
- Dangling Pointers
- Accessing Memory Outside the Bounds of an Array
- Calculating the Array Size Incorrectly
- Misusing the sizeof Operator
- Always Match Pointer Types
- Bounded Pointers
- String Security Issues
- Pointer Arithmetic and Structures
- Function Pointer Issues

Memory Deallocation Issues

- Double Free

Chapter 8 - Odds and Ends

Casting Pointers

- Accessing a Special Purpose Address
- Accessing a Port
- Accessing Memory using DMA
- Determining the Endianness of a Machine
- Using a Union to Represent a Value in Multiple Ways
- Strict Aliasing
- Using the restrict Keyword

Threads and Pointers

- Sharing Pointers Between Threads
- Using Function Pointers to Support Callbacks

Object-Oriented Techniques

- Creating and Using an Opaque Pointer
- Polymorphism in C