

# Microsoft Visual C# Step by Step

Ninth Edition

John Sharp

# Contents

*Acknowledgments*  
*About the Author*  
*Introduction*

## **PART I**    **INTRODUCING MICROSOFT VISUAL C# AND MICROSOFT VISUAL STUDIO 2017**

---

### **Chapter 1**   **Welcome to C#**

Beginning programming with the Visual Studio 2017 environment  
Writing your first program  
Using namespaces  
Creating a graphical application  
    Examining the Universal Windows Platform app  
    Adding code to the graphical application  
Summary  
Quick reference

### **Chapter 2**   **Working with variables, operators, and expressions**

Understanding statements  
Using identifiers  
    Identifying keywords  
Using variables  
    Naming variables  
    Declaring variables  
    Specifying numeric values  
Working with primitive data types  
    Unassigned local variables  
    Displaying primitive data type values  
Using arithmetic operators  
    Operators and types  
    Examining arithmetic operators  
    Controlling precedence  
    Using associativity to evaluate expressions  
    Associativity and the assignment operator  
Incrementing and decrementing variables  
    Prefix and postfix

Declaring implicitly typed local variables

Summary

Quick reference

## **Chapter 3 Writing methods and applying scope**

Creating methods

Declaring a method

Returning data from a method

Using expression-bodied methods

Calling methods

Specifying the method call syntax

Returning multiple values from a method

Applying scope

Defining local scope

Defining class scope

Overloading methods

Writing methods

Refactoring code

Nesting methods

Using optional parameters and named arguments

Defining optional parameters

Passing named arguments

Resolving ambiguities with optional parameters and named arguments

Summary

Quick reference

## **Chapter 4 Using decision statements**

Declaring Boolean variables

Using Boolean operators

Understanding equality and relational operators

Understanding conditional logical operators

Short-circuiting

Summarizing operator precedence and associativity

Using *if* statements to make decisions

Understanding *if* statement syntax

Using blocks to group statements

Cascading *if* statements

Using *switch* statements

Understanding *switch* statement syntax

Following the *switch* statement rules

Summary

Quick reference

## **Chapter 5 Using compound assignment and iteration statements**

Using compound assignment operators

Writing *while* statements

Writing *for* statements

Understanding *for* statement scope

Writing *do* statements

Summary

Quick reference

## **Chapter 6 Managing errors and exceptions**

Coping with errors

Trying code and catching exceptions

Unhandled exceptions

Using multiple catch handlers

Catching multiple exceptions

Filtering exceptions

Propagating exceptions

Using checked and unchecked integer arithmetic

Writing checked statements

Writing checked expressions

Throwing exceptions

Using *throw* exceptions

Using a *finally* block

Summary

Quick reference

## **PART II UNDERSTANDING THE C# OBJECT MODEL**

---

### **Chapter 7 Creating and managing classes and objects**

Understanding classification

The purpose of encapsulation

Defining and using a class

Controlling accessibility

- Working with constructors
- Overloading constructors
- Deconstructing an object
- Understanding static methods and data
  - Creating a shared field
  - Creating a static field by using the *const* keyword
  - Understanding static classes
  - Static *using* statements
- Anonymous classes
- Summary
- Quick reference

## **Chapter 8 Understanding values and references**

- Copying value type variables and classes
- Understanding null values and nullable types
  - The null-conditional operator
  - Using nullable types
  - Understanding the properties of nullable types
- Using *ref* and *out* parameters
  - Creating *ref* parameters
  - Creating *out* parameters
- How computer memory is organized
  - Using the stack and the heap
- The *System.Object* class
- Boxing
- Unboxing
- Casting data safely
  - The *is* operator
  - The *as* operator
  - The *switch* statement revisited
- Summary
- Quick reference

## **Chapter 9 Creating value types with enumerations and structures**

- Working with enumerations
  - Declaring an enumeration
  - Using an enumeration
  - Choosing enumeration literal values

Choosing an enumeration's underlying type

Working with structures

Declaring a structure

Understanding differences between structures and classes

Declaring structure variables

Understanding structure initialization

Copying structure variables

Summary

Quick reference

## **Chapter 10 Using arrays**

Declaring and creating an array

Declaring array variables

Creating an array instance

Populating and using an array

Creating an implicitly typed array

Accessing an individual array element

Iterating through an array

Passing arrays as parameters and return values for a method

Copying arrays

Using multidimensional arrays

Creating jagged arrays

Accessing arrays that contain value types

Summary

Quick reference

## **Chapter 11 Understanding parameter arrays**

Overloading—a recap

Using array arguments

Declaring a *params* array

Using *params object[ ]*

Using a *params* array

Comparing parameter arrays and optional parameters

Summary

Quick reference

## **Chapter 12 Working with inheritance**

What is inheritance?

Using inheritance

- The *System.Object* class revisited

- Calling base-class constructors

- Assigning classes

- Declaring new methods

- Declaring virtual methods

- Declaring *override* methods

- Understanding *protected* access

Creating extension methods

Summary

Quick reference

## **Chapter 13 Creating interfaces and defining abstract classes**

Understanding interfaces

- Defining an interface

- Implementing an interface

- Referencing a class through its interface

- Working with multiple interfaces

- Explicitly implementing an interface

- Interface restrictions

- Defining and using interfaces

Abstract classes

- Abstract methods

Sealed classes

- Sealed methods

- Implementing and using an abstract class

Summary

Quick reference

## **Chapter 14 Using garbage collection and resource management**

The life and times of an object

- Writing destructors

- Why use the garbage collector?

- How does the garbage collector work?

- Recommendations

Resource management

- Disposal methods

- Exception-safe disposal

The *using* statement and the *IDisposable* interface

Calling the *Dispose* method from a destructor

Implementing exception-safe disposal

Summary

Quick reference

## **PART III DEFINING EXTENSIBLE TYPES WITH C#**

---

### **Chapter 15 Implementing properties to access fields**

Implementing encapsulation by using methods

What are properties?

Using properties

Read-only properties

Write-only properties

Property accessibility

Understanding the property restrictions

Declaring interface properties

Replacing methods with properties

Generating automatic properties

Initializing objects by using properties

Summary

Quick reference

### **Chapter 16 Handling binary data and using indexers**

What is an indexer?

Storing binary values

Displaying binary values

Manipulating binary values

Solving the same problems using indexers

Understanding indexer accessors

Comparing indexers and arrays

Indexers in interfaces

Using indexers in a Windows application

Summary

Quick reference

### **Chapter 17 Introducing generics**

The problem: Misusing with the *object* type



The generics solution

- Generics vs. generalized classes

- Generics and constraints

Creating a generic class

- The theory of binary trees

- Building a binary tree class by using generics

Creating a generic method

- Defining a generic method to build a binary tree

Variance and generic interfaces

- Covariant interfaces

- Contravariant interfaces

Summary

Quick reference

## Chapter 18 Using collections

What are collection classes?

- The *List*<*T*> collection class

- The *LinkedList*<*T*> collection class

- The *Queue*<*T*> collection class

- The *Stack*<*T*> collection class

- The *Dictionary*<*TKey*, *TValue*> collection class

- The *SortedList*<*TKey*, *TValue*> collection class

- The *HashSet*<*T*> collection class

Using collection initializers

The *Find* methods, predicates, and lambda expressions

- The forms of lambda expressions

Comparing arrays and collections

- Using collection classes to play cards

Summary

Quick reference

## Chapter 19 Enumerating collections

Enumerating the elements in a collection

- Manually implementing an enumerator

- Implementing the *IEnumerable* interface

Implementing an enumerator by using an iterator

- A simple iterator

- Defining an enumerator for the *Tree*<*TItem*> class by using an iterator

Summary

Quick reference

## **Chapter 20 Decoupling application logic and handling events**

Understanding delegates

- Examples of delegates in the .NET Framework class library

- The automated factory scenario

- Implementing the factory control system without using delegates

- Implementing the factory by using a delegate

- Declaring and using delegates

Lambda expressions and delegates

- Creating a method adapter

Enabling notifications by using events

- Declaring an event

- Subscribing to an event

- Unsubscribing from an event

- Raising an event

Understanding user interface events

- Using events

Summary

Quick reference

## **Chapter 21 Querying in-memory data by using query expressions**

What is LINQ?

Using LINQ in a C# application

- Selecting data

- Filtering data

- Ordering, grouping, and aggregating data

- Joining data

- Using query operators

- Querying data in *Tree<TItem>* objects

- LINQ and deferred evaluation

Summary

Quick reference

## **Chapter 22 Operator overloading**

Understanding operators

- Operator constraints

- Overloaded operators
- Creating symmetric operators
- Understanding compound assignment evaluation
- Declaring increment and decrement operators
- Comparing operators in structures and classes
- Defining operator pairs
- Implementing operators
- Understanding conversion operators
  - Providing built-in conversions
  - Implementing user-defined conversion operators
  - Creating symmetric operators, revisited
  - Writing conversion operators
- Summary
- Quick reference

---

## PART IV BUILDING UNIVERSAL WINDOWS PLATFORM APPLICATIONS WITH C#

---

### Chapter 23 Improving throughput by using tasks

- Why perform multitasking by using parallel processing?
  - The rise of the multicore processor
- Implementing multitasking by using the Microsoft .NET Framework
  - Tasks, threads, and the *ThreadPool*
  - Creating, running, and controlling tasks
  - Using the *Task* class to implement parallelism
  - Abstracting tasks by using the *Parallel* class
  - When not to use the *Parallel* class
- Canceling tasks and handling exceptions
  - The mechanics of cooperative cancellation
  - Using continuations with canceled and faulted tasks
- Summary
- Quick reference

### Chapter 24 Improving response time by performing asynchronous operations

- Implementing asynchronous methods
  - Defining asynchronous methods: The problem
  - Defining asynchronous methods: The solution
  - Defining asynchronous methods that return values
  - Asynchronous method gotchas

- Asynchronous methods and the Windows Runtime APIs
  - Tasks, memory allocation, and efficiency
- Using PLINQ to parallelize declarative data access
  - Using PLINQ to improve performance while iterating through a collection
  - Canceling a PLINQ query
- Synchronizing concurrent access to data
  - Locking data
  - Synchronization primitives for coordinating tasks
  - Canceling synchronization
  - The concurrent collection classes
  - Using a concurrent collection and a lock to implement thread-safe data access
- Summary
- Quick reference

## **Chapter 25 Implementing the user interface for a Universal Windows Platform app**

- Features of a Universal Windows Platform app
- Using the Blank App template to build a Universal Windows Platform app
  - Implementing a scalable user interface
  - Applying styles to a UI
- Summary
- Quick reference

## **Chapter 26 Displaying and searching for data in a Universal Windows Platform app**

- Implementing the Model–View–ViewModel pattern
  - Displaying data by using data binding
  - Modifying data by using data binding
  - Using data binding with a *ComboBox* control
  - Creating a ViewModel
  - Adding commands to a ViewModel
- Searching for data using Cortana
  - Providing a vocal response to voice commands
- Summary
- Quick reference

## **Chapter 27 Accessing a remote database from a Universal Windows Platform app**

- Retrieving data from a database
  - Creating an entity model
  - Creating and using a REST web service

Inserting, updating, and deleting data through a REST web service

Reporting errors and updating the UI

Summary

Quick reference

*Index*