



# Contents

<b>About</b>	1
<b>Chapter 1: Getting started with C# Language</b>	2
Section 1.1: Creating a new console application (Visual Studio)	2
Section 1.2: Creating a new project in Visual Studio (console application) and Running it in Debug mode	4
Section 1.3: Creating a new program using .NET Core	7
Section 1.4: Creating a new program using Mono	9
Section 1.5: Creating a new query using LingPad	9
Section 1.6: Creating a new project using Xamarin Studio	12
<b>Chapter 2: Literals</b>	18
Section 2.1: uint literals	18
Section 2.2: int literals	18
Section 2.3: sbyte literals	18
Section 2.4: decimal literals	18
Section 2.5: double literals	18
Section 2.6: float literals	18
Section 2.7: long literals	18
Section 2.8: ulong literal	18
Section 2.9: string literals	19
Section 2.10: char literals	19
Section 2.11: byte literals	19
Section 2.12: short literal	19
Section 2.13: ushort literal	19
Section 2.14: bool literals	19
<b>Chapter 3: Operators</b>	20
Section 3.1: Overloadable Operators	20
Section 3.2: Overloading equality operators	21
Section 3.3: Relational Operators	22
Section 3.4: Implicit Cast and Explicit Cast Operators	24
Section 3.5: Short-circuiting Operators	25
Section 3.6: ?: Ternary Operator	26
Section 3.7: ?. (Null Conditional Operator)	27
Section 3.8: "Exclusive or" Operator	27
Section 3.9: default Operator	28
Section 3.10: Assignment operator '='	28
Section 3.11: sizeof	28
Section 3.12: ?? Null-Coalescing Operator	29
Section 3.13: Bit-Shifting Operators	29
Section 3.14: => Lambda operator	29
Section 3.15: Class Member Operators: Null Conditional Member Access	31
Section 3.16: Class Member Operators: Null Conditional Indexing	31
Section 3.17: Postfix and Prefix increment and decrement	31
Section 3.18: typeof	32
Section 3.19: Binary operators with assignment	32
Section 3.20: nameof Operator	32
Section 3.21: Class Member Operators: Member Access	33
Section 3.22: Class Member Operators: Function Invocation	33

Section 3.23: Class Member Operators: Aggregate Object Indexing	33
<b>Chapter 4: Conditional Statements</b>	34
Section 4.1: If-Else Statement	34
Section 4.2: If statement conditions are standard boolean expressions and values	34
Section 4.3: If-Else If-Else Statement	35
<b>Chapter 5: Equality Operator</b>	36
Section 5.1: Equality kinds in c# and equality operator	36
<b>Chapter 6: Equals and GetHashCode</b>	37
Section 6.1: Writing a good GetHashCode override	37
Section 6.2: Default Equals behavior	37
Section 6.3: Override Equals and GetHashCode on custom types	38
Section 6.4: Equals and GetHashCode in IEqualityComparator	39
<b>Chapter 7: Null-Coalescing Operator</b>	41
Section 7.1: Basic usage	41
Section 7.2: Null fall-through and chaining	41
Section 7.3: Null coalescing operator with method calls	42
Section 7.4: Use existing or create new	43
Section 7.5: Lazy properties initialization with null coalescing operator	43
<b>Chapter 8: Null-conditional Operators</b>	44
Section 8.1: Null-Conditional Operator	44
Section 8.2: The Null-Conditional Index	44
Section 8.3: Avoiding NullReferenceExceptions	45
Section 8.4: Null-conditional Operator can be used with Extension Method	45
<b>Chapter 9: nameof Operator</b>	47
Section 9.1: Basic usage: Printing a variable name	47
Section 9.2: Raising PropertyChanged event	47
Section 9.3: Argument Checking and Guard Clauses	48
Section 9.4: Strongly typed MVC action links	48
Section 9.5: Handling PropertyChanged events	49
Section 9.6: Applied to a generic type parameter	49
Section 9.7: Printing a parameter name	50
Section 9.8: Applied to qualified identifiers	50
<b>Chapter 10: Verbatim Strings</b>	51
Section 10.1: Interpolated Verbatim Strings	51
Section 10.2: Escaping Double Quotes	51
Section 10.3: Verbatim strings instruct the compiler to not use character escapes	51
Section 10.4: Multiline Strings	52
<b>Chapter 11: Common String Operations</b>	53
Section 11.1: Formatting a string	53
Section 11.2: Correctly reversing a string	53
Section 11.3: Padding a string to a fixed length	54
Section 11.4: Getting x characters from the right side of a string	55
Section 11.5: Checking for empty String using String.IsNullOrEmpty() and String.IsNullOrWhiteSpace()	56
Section 11.6: Trimming Unwanted Characters Off the Start and/or End of Strings	57
Section 11.7: Convert Decimal Number to Binary,Octal and Hexadecimal Format	57
Section 11.8: Construct a string from Array	57
Section 11.9: Formatting using ToString	58
Section 11.10: Splitting a String by another string	59

<a href="#">Section 11.11: Splitting a String by specific character</a>	59
<a href="#">Section 11.12: Getting Substrings of a given string</a>	59
<a href="#">Section 11.13: Determine whether a string begins with a given sequence</a>	59
<a href="#">Section 11.14: Getting a char at specific index and enumerating the string</a>	59
<a href="#">Section 11.15: Joining an array of strings into a new one</a>	60
<a href="#">Section 11.16: Replacing a string within a string</a>	60
<a href="#">Section 11.17: Changing the case of characters within a String</a>	60
<a href="#">Section 11.18: Concatenate an array of strings into a single string</a>	61
<a href="#">Section 11.19: String Concatenation</a>	61
<b><a href="#">Chapter 12: String.Format</a></b>	62
<a href="#">Section 12.1: Since C# 6.0</a>	62
<a href="#">Section 12.2: Places where String.Format is 'embedded' in the framework</a>	62
<a href="#">Section 12.3: Create a custom format provider</a>	62
<a href="#">Section 12.4: Date Formatting</a>	63
<a href="#">Section 12.5: Currency Formatting</a>	64
<a href="#">Section 12.6: Using custom number format</a>	65
<a href="#">Section 12.7: Align left/ right, pad with spaces</a>	65
<a href="#">Section 12.8: Numeric formats</a>	66
<a href="#">Section 12.9: ToString()</a>	66
<a href="#">Section 12.10: Escaping curly brackets inside a String.Format() expression</a>	67
<a href="#">Section 12.11: Relationship with ToString()</a>	67
<b><a href="#">Chapter 13: String Concatenate</a></b>	68
<a href="#">Section 13.1: + Operator</a>	68
<a href="#">Section 13.2: Concatenate strings using System.Text.StringBuilder</a>	68
<a href="#">Section 13.3: Concat string array elements using String.Join</a>	68
<a href="#">Section 13.4: Concatenation of two strings using \$</a>	69
<b><a href="#">Chapter 14: String Manipulation</a></b>	70
<a href="#">Section 14.1: Replacing a string within a string</a>	70
<a href="#">Section 14.2: Finding a string within a string</a>	70
<a href="#">Section 14.3: Removing (Trimming) white-space from a string</a>	70
<a href="#">Section 14.4: Splitting a string using a delimiter</a>	71
<a href="#">Section 14.5: Concatenate an array of strings into a single string</a>	71
<a href="#">Section 14.6: String Concatenation</a>	71
<a href="#">Section 14.7: Changing the case of characters within a String</a>	71
<b><a href="#">Chapter 15: String Interpolation</a></b>	73
<a href="#">Section 15.1: Format dates in strings</a>	73
<a href="#">Section 15.2: Padding the output</a>	73
<a href="#">Section 15.3: Expressions</a>	74
<a href="#">Section 15.4: Formatting numbers in strings</a>	74
<a href="#">Section 15.5: Simple Usage</a>	75
<b><a href="#">Chapter 16: String Escape Sequences</a></b>	76
<a href="#">Section 16.1: Escaping special symbols in string literals</a>	76
<a href="#">Section 16.2: Unicode character escape sequences</a>	76
<a href="#">Section 16.3: Escaping special symbols in character literals</a>	76
<a href="#">Section 16.4: Using escape sequences in identifiers</a>	76
<a href="#">Section 16.5: Unrecognized escape sequences produce compile-time errors</a>	77
<b><a href="#">Chapter 17: StringBuilder</a></b>	78
<a href="#">Section 17.1: What a StringBuilder is and when to use one</a>	78
<a href="#">Section 17.2: Use StringBuilder to create string from a large number of records</a>	79

<b>Chapter 18: Regex Parsing</b>	80
<a href="#">Section 18.1: Single match</a>	80
<a href="#">Section 18.2: Multiple matches</a>	80
<b>Chapter 19: DateTime Methods</b>	81
<a href="#">Section 19.1: DateTime Formatting</a>	81
<a href="#">Section 19.2: DateTime.AddDays(Double)</a>	82
<a href="#">Section 19.3: DateTime.AddHours(Double)</a>	82
<a href="#">Section 19.4: DateTime.Parse(String)</a>	82
<a href="#">Section 19.5: DateTime.TryParse(String, DateTime)</a>	82
<a href="#">Section 19.6: DateTime.AddMilliseconds(Double)</a>	83
<a href="#">Section 19.7: DateTime.Compare(DateTime t1, DateTime t2)</a>	83
<a href="#">Section 19.8: DateTime.DaysInMonth(Int32, Int32)</a>	83
<a href="#">Section 19.9: DateTime.AddYears(Int32)</a>	84
<a href="#">Section 19.10: Pure functions warning when dealing with DateTime</a>	84
<a href="#">Section 19.11: DateTime.TryParseExact(String, String, IFormatProvider, DateTimeStyles, DateTime)</a>	84
<a href="#">Section 19.12: DateTime.Add(TimeSpan)</a>	86
<a href="#">Section 19.13: Parse and TryParse with culture info</a>	86
<a href="#">Section 19.14: DateTime as initializer in for-loop</a>	87
<a href="#">Section 19.15: DateTime.ParseExact(String, String, IFormatProvider)</a>	87
<a href="#">Section 19.16: DateTime.ToString, ToShortDateString, ToLongDateString and ToString formatted</a>	88
<a href="#">Section 19.17: Current Date</a>	88
<b>Chapter 20: Arrays</b>	89
<a href="#">Section 20.1: Declaring an array</a>	89
<a href="#">Section 20.2: Initializing an array filled with a repeated non-default value</a>	89
<a href="#">Section 20.3: Copying arrays</a>	90
<a href="#">Section 20.4: Comparing arrays for equality</a>	90
<a href="#">Section 20.5: Multi-dimensional arrays</a>	91
<a href="#">Section 20.6: Getting and setting array values</a>	91
<a href="#">Section 20.7: Iterate over an array</a>	91
<a href="#">Section 20.8: Creating an array of sequential numbers</a>	92
<a href="#">Section 20.9: Jagged arrays</a>	92
<a href="#">Section 20.10: Array covariance</a>	94
<a href="#">Section 20.11: Arrays as IEnumerable&lt;&gt; instances</a>	94
<a href="#">Section 20.12: Checking if one array contains another array</a>	94
<b>Chapter 21: O(n) Algorithm for circular rotation of an array</b>	96
<a href="#">Section 21.1: Example of a generic method that rotates an array by a given shift</a>	96
<b>Chapter 22: Enum</b>	98
<a href="#">Section 22.1: Enum basics</a>	98
<a href="#">Section 22.2: Enum as flags</a>	99
<a href="#">Section 22.3: Using &lt;&lt; notation for flags</a>	101
<a href="#">Section 22.4: Test flags-style enum values with bitwise logic</a>	101
<a href="#">Section 22.5: Add and remove values from flagged enum</a>	102
<a href="#">Section 22.6: Enum to string and back</a>	102
<a href="#">Section 22.7: Enums can have unexpected values</a>	103
<a href="#">Section 22.8: Default value for enum == ZERO</a>	103
<a href="#">Section 22.9: Adding additional description information to an enum value</a>	104
<a href="#">Section 22.10: Get all the members values of an enum</a>	105
<a href="#">Section 22.11: Bitwise Manipulation using enums</a>	105
<b>Chapter 23: Tuples</b>	106

[Click here to download full PDF material](#)