C# COMPLETE CHEAT SHEET

Every command from Week 1 | 22nd Survey Division | TAFE Cert IV Programming

Print this. Laminate it. Keep it next to your keyboard.

OUTPUT & INPUT

Code	What It Does	Notes
`Console.WriteLine("Hello");`	Print text + new line	Most common output method
`Console.Write("Hello");`	Print text, stay on same line	No Enter at end
`Console.ReadLine();`	Wait for user to type + press Enter	ALWAYS returns a string, even if they type a number
`Console.WriteLine($"Hi {name}");`	Print with variable embedded	$ before the quote activates { } interpolation
`Console.WriteLine("A: " + x);`	Print by joining strings with +	Concatenation — works but $ is cleaner
`Console.WriteLine($"{i + 1}. {name}");`	Print with maths inside { }	You can do maths inside the braces

VARIABLES & TYPES

Code	What It Does	Notes
`string name = "Wu";`	Create a text variable	Double quotes always
`int age = 29;`	Create a whole number	No decimals allowed
`double price = 19.99;`	Create a decimal number	For money, measurements
`bool alive = true;`	Create true/false	Lowercase true/false in C#
`char letter = 'A';`	Create single character	Single quotes, one char only
`const int YEAR = 2026;`	Create a constant (locked)	Cannot change after creation
`var x = 5;`	Let C# figure out the type	Only works if you assign a value immediately
`string name;`	Declare without a value	Must assign before using or compiler error
`int age = 0;`	Declare with default value	Safe — always has a value

ARRAYS

Code	What It Does	Notes
`string[] movies = {"A","B","C"};`	Create an array (list)	[] = array of. Fixed size once created.
`movies[0]`	Access first item	Arrays start at 0, NOT 1

`movies[movies.Length - 1]`	Access last item	Last item is always Length minus 1
`movies.Length`	How many items in array	No () — it's a property not a method
`int[] nums = {1, 2, 3};`	Array of numbers	All items must be same type
`string[] lines = File.ReadAllLines("f");`	File lines into an array	Each line becomes one array item
`movies[5]`	Access item 6	If only 4 items exist = IndexOutOfRangeException CRASH

ARITHMETIC OPERATORS

Code	What It Does	Notes
`5 + 3`	Add → 8	Also joins strings: "Hi" + " " + "Wu" = "Hi Wu"
`5 - 3`	Subtract → 2	
`5 * 3`	Multiply → 15	Asterisk, not x
`10 / 3`	Divide → 3 (not 3.33!)	int / int = int (chops decimal). Use 10.0/3.0 for 3.33
`10 % 3`	Remainder → 1	Modulo. 10 divided by 3 = 3 remainder 1
`i++`	Add 1 to i	Same as i = i + 1
`i--`	Subtract 1 from i	Same as i = i - 1
`i += 5`	Add 5 to i	Same as i = i + 5
`i -= 3`	Subtract 3 from i	Same as i = i - 3
`i *= 2`	Multiply i by 2	Same as i = i * 2
`i /= 4`	Divide i by 4	Same as i = i / 4

COMPARISON OPERATORS (return true or false)

Code	What It Does	Notes
`x == 5`	Is x equal to 5?	TWO equals signs = asking. ONE = setting.
`x != 5`	Is x NOT equal to 5?	
`x < 5`	Is x less than 5?	
`x > 5`	Is x greater than 5?	
`x <= 5`	Is x less than or equal to 5?	
`x >= 5`	Is x greater than or equal to 5?	

LOGICAL OPERATORS

Code	What It Does	Notes
`a && b`	AND — both must be true	Python: and
`a || b`	OR — either can be true	Python: or
`!a`	NOT — flips true to false	Python: not

CONDITIONS (IF / ELSE)

Code	What It Does	Notes
`if (condition) { }`	Run code only if true	Parentheses around condition REQUIRED
`else if (condition) { }`	Check next condition	Only checked if all above were false
`else { }`	Run if nothing above was true	No condition — it's the fallback
`if (x > 5 && x < 10) { }`	Both conditions must be true	AND logic
`if (x < 5 || x > 10) { }`	Either condition can be true	OR logic
`if (!isAdmin) { }`	If isAdmin is false	NOT flips the boolean

LOOPS

Code	What It Does	Notes
`for (int i = 0; i < 10; i++) { }`	Loop with counter	i starts at 0, runs while < 10, adds 1 each time
`foreach (string s in list) { }`	Loop through each item	No counter — just gives you each item
`while (condition) { }`	Loop while true	Careful: infinite loop if condition never becomes false
`break;`	Exit the loop entirely	Emergency stop — loop is OVER
`continue;`	Skip this iteration, next item	Jump to next loop cycle
`for (int i = 0; i < arr.Length; i++)`	Loop through array by index	Use when you need the position number (i)
`foreach (string line in lines)`	Loop through array by value	Use when you just want each item

STRING OPERATIONS

Code	What It Does	Notes
`$"Hello {name}"`	Interpolation — embed variables	$ BEFORE the opening quote, variables in { }
`@"C:\Users\file"`	Literal string — backslashes as-is	@ BEFORE the opening quote
`$@"C:\{user}\file"`	Both interpolation AND literal	For file paths with variables
`line.Split(',')`	Chop string at every comma	Returns an array of pieces
`name.ToLower()`	Make all lowercase	"Wu" → "wu"
`name.ToUpper()`	Make all uppercase	"Wu" → "WU"
`name.Contains("wu")`	Does it contain this text?	Returns true or false. Case-sensitive!
`name.ToLower().Contains("wu")`	Case-insensitive search	Chain: lowercase first, then search

`"Wu".PadRight(10)`	Pad to 10 chars with spaces	"Wu " — for column alignment
`"Wu".PadLeft(10)`	Right-align to 10 chars	" Wu"
`name.Length`	Number of characters	Property, no ()
`name.Trim()`	Remove spaces from both ends	" Wu " → "Wu"
`name.Replace("a", "b")`	Replace all a with b	Returns new string (doesn't modify original)
`string.IsNullOrWhiteSpace(s)`	Is it null, empty, or just spaces?	Static method — use for blank line checks
`name.ToString()`	Convert to string	Every type has this

TYPE CONVERSION

Code	What It Does	Notes
`int.Parse("42")`	String to int	Crashes with FormatException if not a number
`double.Parse("19.99")`	String to double	Crashes if not a valid decimal
`age.ToString()`	Number to string	29 → "29"
`(int)19.99`	Double to int (chop decimals)	19.99 → 19 (does NOT round — just cuts)
`(int)Math.Round(19.99)`	Double to int (proper rounding)	19.99 → 20 (rounds first, then converts)
`Convert.ToInt32("42")`	String to int (alternative)	Handles null better than int.Parse

MATH CLASS

Code	What It Does	Notes
`Math.Max(5, 10)`	Returns the larger → 10	Not Maths. American spelling.
`Math.Min(5, 10)`	Returns the smaller → 5	
`Math.Abs(-5)`	Removes negative → 5	Absolute value
`Math.Round(3.7)`	Rounds → 4	Banker's rounding (2.5 rounds to 2, 3.5 rounds to 4)
`Math.Sqrt(25)`	Square root → 5	Returns a double
`Math.Pow(2, 3)`	Power → 8	2 to the power of 3

FILE READING & WRITING

Code	What It Does	Notes
`File.ReadAllLines("data.csv")`	Read file into string array	Each line = one array item
`File.ReadAllText("data.txt")`	Read entire file as one string	Good for small files
`File.WriteAllLines("out.txt", arr)`	Write array to file	Each array item = one line
`File.WriteAllText("out.txt", str)`	Write string to file	Overwrites if file exists
`new StreamWriter("out.csv")`	Create file writer (line by line)	Wrap in using() to auto-close
`writer.WriteLine($"{a},{b}")`	Write one line to file	Inside a using(StreamWriter) block
`File.Exists("data.csv")`	Does the file exist?	Returns true/false — check before reading

ERROR HANDLING (TRY-CATCH)

Code	What It Does	Notes
`try { }`	Attempt risky code	If it fails, jumps to catch
`catch (FileNotFoundException ex) { }`	Catch specific error	Only runs if file not found
`catch (FormatException ex) { }`	Catch format error	Only runs if type conversion fails
`catch (IndexOutOfRangeException ex) { }`	Catch array bounds error	Only runs if array index invalid
`catch (Exception ex) { }`	Catch ANY error	General fallback — put LAST
`ex.Message`	Get the error description	Human-readable error text

KEYWORDS & SPECIAL WORDS

Code	What It Does	Notes
`null`	Nothing. Variable points nowhere.	Not 0, not "", not false. NOTHING.
`new`	Create a fresh instance of something	new StreamWriter(), new List<string>()
`using System.IO;`	Import a toolkit (top of file)	Gives access to File, StreamWriter etc
`using (resource) { }`	Auto-close when done (in code)	For files, connections — auto-cleanup
`const`	Lock a variable forever	const int X = 5; — can never change
`var`	Let C# guess the type	Only works with immediate assignment
`true / false`	Boolean values	Lowercase in C# (Python uses True/False)
`return`	Exit a method and give back a value	You'll use this more in Week 2+

ESCAPE CHARACTERS (inside strings)

Code	What It Does	Notes
`\n`	New line (Enter)	"Line1\nLine2" prints on two lines
`\t`	Tab (indent)	"A\tB" = "A B"
`\"`	Literal quote mark	"She said \"hi\"" prints: She said "hi"
`\\`	Literal backslash	Without @, you need \\ for one \

STRUCTURE RULES

Rule	Example	Why
Every { must have a matching }	if () { }	Compiler error if mismatched
Statements end with ;	Console.WriteLine("Hi");	Semicolon = full stop
Structure lines do NOT end with ;	if (age > 18)	The { block follows instead
Conditions need ( )	if (x == 5)	Required in C#, optional in Python
Arrays start at 0	arr[0] = first item	NOT arr[1]
C# is case-sensitive	Console ≠ console	Spelling + capitalisation matter
= sets, == asks	x = 5 vs x == 5	Assignment vs comparison
Variables die at their }	{ int x = 5; } // x dead	Scope: declare outside if needed after
for() has 3 parts separated by ;	for (start; condition; step)	Those ; are separators, not endings
Indent after every {	if { \n code	For YOUR brain, not the compiler
Fix the FIRST error	Ignore cascade errors	One error often causes 10 more

VISUAL STUDIO SHORTCUTS

Shortcut	What It Does	When
F5	Run program (with debugging)	Testing your code
Ctrl+F5	Run without debugging	Just want to see output
Ctrl+A, Ctrl+K, Ctrl+D	Auto-fix all indentation	After EVERY edit
Ctrl+K, Ctrl+C	Comment out selected lines	Disable code temporarily
Ctrl+K, Ctrl+U	Uncomment selected lines	Re-enable code
Ctrl+Z	Undo	When you break something
Ctrl+S	Save	Constantly
Ctrl+. (dot)	Quick fix / suggestions	Red squiggly line? Try this
Click a { or }	Highlights matching brace	Finding missing braces
F12	Go to definition	See what a method does

KEYWORD (setup) { stuff; } — That's every C# structure. Same shape. Every time.

Vidimus Omnia.