CALLYWITH COLLEGE · SUMMER PREPARATION WORK

Getting Started with C#

Preparation for A Level Computer Science and the Level 3 Cambridge Advanced National in Computing: Application Development

About this prep work

Over the next two years you will be writing real programs in C# (pronounced "C-sharp"), a professional programming language created by Microsoft. It is used to build desktop apps, websites, mobile apps and games (it is the language behind the Unity game engine), and much more. This booklet is designed to get you comfortable with the basics of C# over the summer, so that when your first lessons begin you already recognise the language and can spend your energy building things rather than wrestling with unfamiliar syntax.

You do not need any previous experience with C# to do this. If you took GCSE Computer Science you will recognise many of the ideas (variables, loops, conditions and so on), but the way C# writes them will be new. If C# is completely new to you, do not worry: everything is explained step by step on the two websites you will be using.

Two websites, used together

1. W3Schools C# Tutorial — clear, example-led explanations of each topic, with a built-in "Try it Yourself" editor. You will use this to read about and understand each concept.

2. LearnCS.org (Learn C#) — a free, interactive tutorial that lets you write and run C# code directly in your browser. You will use this to practise what you have just read.

You will switch between the two: read a topic on W3Schools to understand it, then head straight to LearnCS.org to write some code and make it stick. Reading about programming is never enough on its own; the ideas only really click once you type code yourself and watch it run.

How this worksheet works

The work is split into 14 short parts, each covering one topic area. Work through them in order, because each part builds on the ones before. Every part follows the same simple pattern:

READ the listed pages on W3Schools to learn the topic.

PRACTISE by completing the matching exercise on LearnCS.org. Type the code yourself and run it; do not just read the solution.

CHECK YOUR UNDERSTANDING by answering a few short questions in your own words, so you know it has gone in.

Tick the boxes as you go, and use the progress tracker below to see how far you have come. A few parts have no interactive exercise; those include a short HAVE A GO task instead.

> **Suggested pace:** aim for one or two parts a week across the summer. It is far better to do a little regularly than to cram it all at once. In total this takes most students somewhere around 12 to 20 hours.

Progress tracker

Tick off each part as you finish it.

Part	Topic	Done	Date
1	What C# is, and your first program	☐	
2	Variables and data types	☐	
3	Getting input from the user	☐	
4	Converting between types (casting)	☐	
5	Operators and maths	☐	
6	Working with text (strings)	☐	
7	Booleans and making decisions	☐	
8	Loops (repeating things)	☐	
9	Arrays	☐	
10	Lists and dictionaries	☐	
11	Methods	☐	
12	Object-oriented programming: classes and objects	☐	
13	More object-oriented programming	☐	
14	Files and error handling	☐	

Before you begin

Nothing to install

Both websites run C# inside your web browser, so you can do everything on a college computer, a home laptop, or even a tablet. There is nothing to download or set up.

How to run your code

- On W3Schools, look for the green "Try it Yourself" button under each example. It opens an editor where you can change the code and run it to see what happens.
- On LearnCS.org, each chapter has a code window on the right. Type your code, press Run, and the output appears below. There is a Solution button if you get stuck, but try to use it only after you have had a proper go yourself.

Optional: save your progress

Both sites let you create a free account to keep track of what you have completed. This is entirely optional and not needed for the prep work.

Optional: for the keen

If you would like to try C# the way we will use it in class, you can install the free Visual Studio Community edition, or Visual Studio Code together with the .NET SDK, on a home computer. This is not required for the prep work (the websites are enough), but it is a good head start. We will help everyone get set up in the first couple of weeks of term, so do not worry if you cannot install anything at home.

What a C# program looks like

Every program you write will start off looking something like this. By the end of this booklet, every line of it will make sense to you.

```
using System;

class Program
{
    static void Main()
    {
        Console.WriteLine("Hello, Callywith!");
    }
}
```

Tips for getting the most out of it

- Type the code yourself rather than copying and pasting. Making (and then fixing) small mistakes is one of the best ways to learn.
- Experiment. Change the examples, break them on purpose, and see what happens. You cannot damage anything.
- Do not rush. It is completely fine to reread a page or redo an exercise until it makes sense.
- Errors are normal. Every programmer sees them constantly. Read the error message, because it is usually telling you exactly what is wrong.
- Keep this booklet and your notes, and bring them to your first lessons.

The prep work

Work through Parts 1 to 14 in order. Remember the pattern for each part: READ on W3Schools, PRACTISE on LearnCS.org, then CHECK YOUR UNDERSTANDING.

Part 1 · What C# is, and your first program

Start here. You will find out what C# is, run your very first program, and learn how a C# program is put together and how to print text to the screen.

READ — read these pages on W3Schools, in order

- ☐ C# Intro
- ☐ C# Get Started
- ☐ C# Syntax
- ☐ C# Output (print text)
- ☐ C# Comments

PRACTISE — write and run the code yourself on LearnCS.org

- ☐ Hello, World!

CHECK YOUR UNDERSTANDING — answer in your own words

1. In your own words, what is C# used for? Name two kinds of software you can build with it.
2. What does the line Console.WriteLine("Hello"); do? How is WriteLine different from Write?
3. What are comments, and why do programmers add them? How do you write a single-line comment in C#?

Part 2 · Variables and data types

Variables are named boxes that store information while a program runs. Here you will learn how to create them and the different types of data they can hold.

READ — read these pages on W3Schools, in order

- ☐ C# Variables
- ☐ C# Constants
- ☐ C# Display Variables
- ☐ C# Multiple Variables
- ☐ C# Identifiers (naming rules)
- ☐ C# Data Types

PRACTISE — write and run the code yourself on LearnCS.org

- ☐ Variables and Types

CHECK YOUR UNDERSTANDING — answer in your own words

1. What is a variable? Show how you would declare one that stores a whole number and one that stores some text.
2. What is the difference between int, double, string and bool? Give an example value for each.
3. What is a constant, and when would you use one instead of an ordinary variable?

Part 3 · Getting input from the user

So far your programs have only shown text. Now you will let the user type something in and use their answer.

READ — read these pages on W3Schools, in order

☐ C# User Input

PRACTISE — write and run the code yourself on LearnCS.org

☐ User Input

CHECK YOUR UNDERSTANDING — answer in your own words

1. Which C# command reads a line of text typed in by the user?
2. When you read input, what data type do you get back, even if the user typed a number? Why does that matter for the next topic?

Part 4 · Converting between types (casting)

User input always arrives as text. To do maths with it, you first have to convert it into a number. That is what casting and type conversion are for.

READ — read these pages on W3Schools, in order

☐ C# Type Casting

PRACTISE — write and run the code yourself on LearnCS.org

☐ Type Conversion

CHECK YOUR UNDERSTANDING — answer in your own words

1. What is the difference between implicit and explicit conversion? One sentence for each is fine.
2. If a user types the number 7 into your program, why can you not immediately add 3 to it? What must you do first?
3. Name one method you could use to turn the text "7" into the number 7.

Part 5 · Operators and maths

Operators are the symbols you use to do calculations and comparisons: adding, subtracting, comparing values and combining conditions. There is no interactive exercise for this topic, so a short challenge is provided below instead.

READ — read these pages on W3Schools, in order

- ☐ C# Operators
- ☐ C# Assignment Operators
- ☐ C# Comparison Operators
- ☐ C# Logical Operators
- ☐ C# Math

HAVE A GO

Using the green "Try it Yourself" button on any W3Schools example (or the editor on LearnCS.org), write a short program that: creates two number variables; prints their sum, difference, product and remainder (use the % operator); and then prints whether the first number is greater than the second (a true or false result). Note anything that surprised you in the box below.

CHECK YOUR UNDERSTANDING — answer in your own words

1. What does the % (modulus) operator give you? Give an example.
2. What is the difference between = and == in C#? (Mixing these up is one of the most common beginner mistakes.)
3. What do the && and || operators mean, and when would you use each?

Part 6 · Working with text (strings)

Text values are called strings. Here you will learn how to join them together, slot values neatly inside them, and pull out individual characters.

READ — read these pages on W3Schools, in order

- ☐ C# Strings
- ☐ C# String Concatenation
- ☐ C# String Interpolation
- ☐ C# Access Strings
- ☐ C# Special Characters

PRACTISE — write and run the code yourself on LearnCS.org

- ☐ Strings

CHECK YOUR UNDERSTANDING — answer in your own words

1. What is string concatenation? Show how you would join "Cally" and "with" into one string.
2. What is string interpolation, and why is it often tidier than concatenation? (Hint: the $ symbol.)
3. How would you find out how many characters are in a string?

Part 7 · Booleans and making decisions

Programs become useful when they can make decisions. Here you meet the true/false (boolean) type, and the if / else if / else and switch statements that let your code choose what to do.

READ — read these pages on W3Schools, in order

- ☐ C# Booleans
- ☐ C# If...Else
- ☐ C# Else
- ☐ C# Else If
- ☐ C# Short Hand If...Else
- ☐ C# Switch

PRACTISE — write and run the code yourself on LearnCS.org

- ☐ Conditionals

CHECK YOUR UNDERSTANDING — answer in your own words

1. What type of value does a condition (such as age > 17) evaluate to?
2. Explain in your own words the difference between if, else if and else.
3. When might you choose a switch statement instead of a long chain of else if statements?

Part 8 · Loops (repeating things)

Loops let you repeat a block of code without writing it out many times. You will learn three kinds (while, for and foreach) and how to stop or skip a loop with break and continue. There are three exercises to complete for this part.

READ — read these pages on W3Schools, in order

- ☐ C# While Loop
- ☐ C# For Loop
- ☐ C# Foreach Loop
- ☐ C# Break / Continue

PRACTISE — write and run the code yourself on LearnCS.org (complete each one)

- ☐ For loops
- ☐ Foreach loops
- ☐ While loops

CHECK YOUR UNDERSTANDING — answer in your own words

1. What is the difference between a while loop and a for loop? When is each most useful?
2. What does a foreach loop do, and what kind of thing do you usually use it with?
3. What do break and continue do inside a loop?

Part 9 · Arrays

An array stores many values of the same type under a single name, like a numbered list of scores. Here you will create arrays, loop through them, sort them, and even make grids (2D arrays).

READ — read these pages on W3Schools, in order

- ☐ C# Arrays
- ☐ C# Loop Through an Array
- ☐ C# Sort Arrays
- ☐ C# Multidimensional Arrays

PRACTISE — write and run the code yourself on LearnCS.org

- ☐ Arrays

CHECK YOUR UNDERSTANDING — answer in your own words

1. What is an array? Give an example of a situation where one would be useful.
2. Array positions are numbered starting from which number? What would you write to get the first item of an array called scores?
3. How could you go through every item in an array and print it out? Name a type of loop that suits this well.

Part 10 · Lists and dictionaries

Arrays have a fixed size. Lists can grow and shrink as your program runs, and dictionaries let you look values up by a key (like a real dictionary: look up a word, get its definition). These two topics are not in the W3Schools basic menu, so use the explanation at the top of each LearnCS.org exercise below to learn the topic before you write the code.

READ

There is no separate W3Schools reading for this part. Read the explanation at the top of each LearnCS.org exercise below before completing it.

PRACTISE — write and run the code yourself on LearnCS.org (complete each one)

- ☐ Lists
- ☐ Dictionaries

CHECK YOUR UNDERSTANDING — answer in your own words

1. What is the main advantage of a List over an array?
2. What is a dictionary, and what do we mean by its keys and its values? Give a real-world example of data that would fit a dictionary well.

Part 11 · Methods

A method is a named, reusable block of code that does a job; you "call" it whenever you need it. Methods keep programs organised and stop you repeating yourself. You will learn to create them, pass information in (parameters), and get answers back (return values).

READ — read these pages on W3Schools, in order

- ☐ C# Methods
- ☐ C# Method Parameters
- ☐ C# Default Parameter

- ☐ C# Return Values
- ☐ C# Named Arguments
- ☐ C# Method Overloading

PRACTISE — write and run the code yourself on LearnCS.org

- ☐ Methods

CHECK YOUR UNDERSTANDING — answer in your own words

1. What is a method, and why are methods useful? (Think about repeating code.)
2. What is the difference between a parameter and a return value?
3. You have already used one method many times in this course without writing it yourself. Which one? (Hint: it prints things to the screen.)

Part 12 · Object-oriented programming: classes and objects

C# is an object-oriented language. A class is a blueprint that describes what something is (its data) and what it can do (its methods); an object is a specific thing made from that blueprint. This is one of the most important ideas in the whole course, so take your time. There are three exercises for this part.

READ — read these pages on W3Schools, in order

- ☐ C# OOP (intro)
- ☐ C# Classes and Objects
- ☐ C# Multiple Objects
- ☐ C# Class Members
- ☐ C# Constructors
- ☐ C# Access Modifiers
- ☐ C# Properties

PRACTISE — write and run the code yourself on LearnCS.org (complete each one)

- ☐ Basic Classes
- ☐ Class Variables
- ☐ Class Properties

CHECK YOUR UNDERSTANDING — answer in your own words

1. In your own words, what is the difference between a class and an object? Try to think of a real-world analogy.
2. What is a constructor, and what is it for?
3. What do the keywords public and private control?

Part 13 · More object-oriented programming

These topics build on the last part and complete the picture of object-oriented programming in C#. You will revisit all of them in class, so aim to get the general idea now rather than mastering every detail. There is no interactive exercise, so a short reflection is provided.

READ — read these pages on W3Schools, in order

- ☐ C# Inheritance
- ☐ C# Polymorphism
- ☐ C# Abstraction
- ☐ C# Interface
- ☐ C# Multiple Interfaces
- ☐ C# Enums

HAVE A GO

These are bigger ideas that take time to sink in. After reading, write a one-sentence, plain-English summary of each in the box below: inheritance, polymorphism, abstraction, interfaces and enums. Do not worry about getting them perfect; a rough understanding now will make the lessons much easier.

CHECK YOUR UNDERSTANDING — answer in your own words

1. Inheritance lets one class be based on another. Give a real-world example of a general thing and a more specific version of it (for example, Animal and Dog).
2. In one sentence, what problem do you think classes and objects are trying to solve in large programs?

Part 14 · Files and error handling

The final two topics. Reading and writing files lets your programs remember information after they close, and exception handling lets your program cope gracefully when something goes wrong instead of crashing.

READ — read these pages on W3Schools, in order

- ☐ C# Files
- ☐ C# Exceptions

HAVE A GO

There is no interactive exercise here. In the box below, note down: (a) one situation where a program would need to save data to a file, and (b) one situation where something could go wrong while a program is running that it should handle gracefully (for example, the user typing letters where a number was expected).

CHECK YOUR UNDERSTANDING — answer in your own words

1. Why would a program need to read from or write to a file? Give one example.
2. What is an exception? What does a try...catch block let your program do when one happens?

You have reached the end. Well done!

If you have worked through all 14 parts, you have now seen every core feature of the C# language: output, variables, input, operators, strings, decisions, loops, arrays, lists and dictionaries, methods, and object-oriented programming with classes. That is a genuinely strong foundation to start the course with.

Going further (optional)

- Put it all together: try writing a small console program of your own. A quiz, a times-tables generator, a simple number-guessing game, or a program that works out something useful all make great projects. Even 20 to 30 lines is a real achievement.
- Test yourself on W3Schools: the tutorial has a set of C# Exercises and a C# Quiz that are a good way to check what has stuck.
- Revisit anything that felt shaky. There is no prize for rushing; understanding is what counts.

What to expect in your first lessons

- We will be programming in C#, most likely using Visual Studio. We will get everyone set up in the first week, so it does not matter if you could not install anything at home.
- We will assume you have met the basics in this booklet (variables, input and output, conditions, loops, arrays, methods and classes), but we will recap the key ideas as we go, and no one will be left behind.
- Bring your questions. If something in here did not make sense, that is genuinely useful for us to know, so note it down and ask.

See you in September.