# KIT506 Programming Practical 1: Getting Started

[ [↶ Close all sections](https://mylo.utas.edu.au/content/enforced/214790-AW_EAI_17S2_15468_0_0_0_1_1/kit506_pracs/KIT506_Prac_01.html?d2lSessionVal=f4tSZ092i00c4rlmCOfiXiw7h&ou=214790&d2l_body_type=3) ] Only visible sections will be included when printing this document.

**Aims:**

* to write a first C# program;
* to become familiar with compiler (error) messages; and
* to gain experience debugging some programs.

# ∨ 1 Foreword

This series of six introductory practical classes are not sufficient to teach you how to program. You should also work through the [Introduction to Programming (in C#)](https://mylo.utas.edu.au/content/enforced/214790-AW_EAI_17S2_15468_0_0_0_1_1/modules%E2%80%8E%EF%BB%BF/00%20Introductory%20material%20for%20KIT506/beginners_guide_to_c_sharp.html?ou=214790) notes in Module 0 on MyLO, and the first 12 parts of Microsoft’s [C# Fundamentals for Absolute Beginners](https://mva.microsoft.com/en-US/training-courses/c-fundamentals-for-absolute-beginners-16169) video tutorials.

# ∨ 2 Open Visual Studio and create a new C# Console Application

From the **Start screen**, select **Visual Studio 2013** in the **Development Tools** group.

Visual Studio is a fully-featured integrated development environment, meaning it provides editing tools for writing program code, the ability to compile and execute (and debug) programs, and a host of other features, such as tools to design graphical user interfaces (which you’ll use in the second half of semester).

Follow the following steps to create a new console (text-driven) application:

1. Choose New > Project… from the File menu.
2. Select Visual C# in the tree of templates and choose Console Application.
3. Name your project HelloWorld and check the location at the bottom of the New Project window.   
   We suggest you create a specific folder on your network drive for all of the projects you create in this series of practical calsses.   
   Click OK.
4. When the Application Wizard starts click Next.
5. A template program will be generated for you. It will include a number of **using** statements, which indicate which collections of pre-existing code your program requires to operate. You do not need to worry about these now if you’re new to programming.

Type (or paste) the following line into the body of the Main() method, between the open brace { and closing brace }:

Console.WriteLine("Hello world!");

Your program (ignoring the **using** statements) will now look like:

**namespace** HelloWorld

{

**class** Program

{

**static** void Main(string[] args)

{

Console.WriteLine("Hello world!");

}

}

}

The namespace declaration is how C# organises pieces of code into related groups. You will encounter this as part of the object-oriented modelling part of the unit as the more general name package.

Currently your program is named Program, so try renaming it by placing the cursor inside the text Program and pressing F2 on the keyboard. Call it HelloWorld (no spaces).

**NOTE:** *Code editors (and these practical notes) use colour to indicate the meaning of different parts of code. This is called*syntax highlighting*. You do not have to—indeed, you cannot—type green or blue text. The editor will do that for you. As you learn what the colours represent you will be able to spot some syntax errors because the colours will look ‘wrong’.*

# ∨ 3 Compile and execute your program

Before your program’s instructions can be executed, the program needs to be compiled. This is the process of transforming your human-readable source code you have created into a machine-readable language. If there are any errors in your code, many compilers are configured to pick them up and report the location of the problem back to you.

You can compile and run your program in a single step by selecting Start or Start Without Debugging from the Debugmenu, or by pressing F5 (for Start) or Ctrl+F5 (for Start without debugging). If you choose to start (with debugging, the default) then when your program finishes it will immediately close the console window, so most of the time you will use Start without debugging. **Do that now:** Press Ctrl + F5 (the Ctrl key and F5 key at the same time) to run your program.

# ∨ 4 Customise the message

Go back to editing your HelloWorld program. Find a short poem or piece of text you like that spans multiple lines. Write statements using Console.WriteLine() to print each line in order.

Change the order of the WriteLine() statements, recompile and run your program. Is the output what you expected?

# ∨ 5 Introduce some syntax errors

Initially, compiler error messages can seem unfamiliar and overwhelming. It takes a little practice to know what to look for. Start by looking for line numbers that correspond to your source code. Common errors are missing semi-colons at the end of a line of code and misspellings. You may find the use of auto-complete helps the speed and accuracy of your code writing. As you begin writing a program statement you can request the possible completions by pressing the Shift key and the space bar simultaneously (they will also appear automatically most of the time).

Let’s practice reading error messages by introducing some (not all of these are errors though!). Try each of the following one at a time: introduce the error, try compiling, look at the error message(s) produced, then undo the change. Some errors will appear in the Error List at the bottom of the window before you even try to compile.

When Visual Studio pops up a message saying There were build errors. Would you like to continue and run the last successful build?, select **No**, and deal with the error.

Suggested errors:

* delete the “}” at the end of the file;黄色 加下标小红
* misspell “Main” as “Man”; 黄色
* delete a semicolon or two; 下标小红
* delete the closing double quote (") on one of the WriteLine() statements; and
* misspell “args” as “double” (which is a reserved word). What happens if you change it to “arg” or “fred”? 长线红 arg反应不大 黄色，小红下标

Note that the error Visual Studio reports may be identified at the wrong location (the error may be earlier but cause the compiler to fail to understand something later), or that the error causes a different error to be indicated. You need to think about what the compiler is telling you and some interpretation may be required.

# ∨ 6 Test yourself: read some code

Here is a sample program that has some syntax errors. What are they? Once you’ve had a guess, paste it into a new C# console application (Reading1, which you can delete later) by replacing all of the code apart from the **using** statements at the top to see what the compiler says.

class Reading1

static void Main(string[] args)

Console.WriteLine("What's missing?");

// 左边有长黄线

Here is another sample program that has some syntax errors. What are they?

class Reading2

{

static void Main(string[] args)

{

Console.WriteLine("Something is")

Console.WriteLine("not quite")

Console.WriteLine("right")

}

}

红后标+左边黄

Finally, what’s the output from running this program? (Make your prediction before trying it out.)

**class** Reading3

{

**static** void Main(string[] args)

{

Console.WriteLine("Strings are versatile.");

Console.WriteLine("You can append to their end: " + 10);

Console.Write("WriteLine adds a newline, while ");

Console.WriteLine("Write does not.");

}

}

完好的

# ∨ 7 What’s this? Solutions?

Each of these practicals includes sample solutions. To enable them you need to click this button: