

IFN647 Advanced Information Retrieval and Storage

Week 1 Practical: Introduction to C#, File Handling

This week's tutorial will cover the fundamental programming prerequisites required for this unit. If you have no previous programming experience please make sure you let your tutor know.

Activity 1: Introduction to Visual Studio

This activity will show you how to use Visual Studio. Visual Studio is an example of an integrated development environment which can assist programmers since they provide: a graphical user interface (GUI), a user-friendly text processing system, easy access to compilers, and access to help documentation.

You can access Visual Studio from the laboratory computers by following these steps:

1. Select **"Start Menu"->"All Programs"->" Visual Studio 2015**
2. You will require a Microsoft Account to sign into Visual Studio (use your QUT email address and password)
3. Be patient – starting Visual Studio on the lab machines takes a little while!

NB Ignore any capability warnings.

Activity 2: Writing your first C# Project

For this activity you are going to write your first C# application in Visual Studio. This requires you to create a project.

1. Select **"FILE->New->Project"** from the Menu
2. In the main section of the Dialog Box, Select **"Console Application"** (ensure that Visual C# is selected in the list of Templates on the Left). On the bottom of the Dialog Box, Enter **"HelloWorld"** as the **Name/Solution** name and choose an appropriate **Location** (for example on your Desktop). Make sure that **"Create Directory for Solution"** is ticked. This identifies what type of application we are creating and where it is to be stored on the hard drive. Here, we are declaring that we want to create an application that reads to and from the command line rather than a GUI (or something else).
3. Change the name of the Class from **"Program"** to **"HelloWorld"**. Also, right click name of the source code on the right hand side browser and change its name from the **"Program.cs"** to **"HelloWorld.cs"**
4. Inside of the Main function:
 - a. Create a string called **myString** with the value of **"Hello World"**;
 - b. Add commands to:
 - write out the value of **myString** to the console and (**Console.WriteLine**)
 - receive a line of input from the console (**Console.ReadLine**)

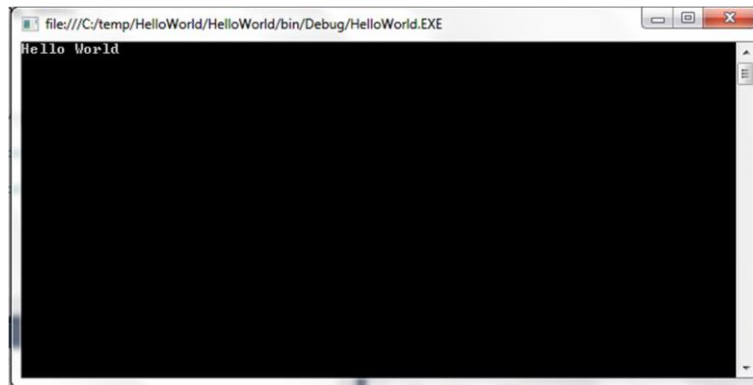
IFN647 Advanced Information Retrieval and Storage

Week 1 Practical: Introduction to C#, File Handling

To execute the program, you need to:

1. Build the program by selecting “**BUILD->Build Solution**” from the Menu.
2. Run the program by selecting “**DEBUG->Start Debugging**” from the Menu Bar or selecting “**Start**” from the Menu Bar or by hitting **F5**.

There may be error with your code which you need to fix before it can be compiled. However, if everything has been built correctly you should receive a similar output to the image on the next page.



Congratulations you have created and executed your first C# application!

Activity 3: Explore Your Solution

After you created your project, multiple files will have been created by Visual Studio. You do not need to know the details of all of these file, however, two types of important files are:

1. Source files. Which should be stored in the directory you created in Step 2 of Activity 2.
2. Executable. Which should be stored in the “**bin\Debug**” subdirectories of the directory you created in Step 2 of Activity 2.

For this project there should only be one source file (likely called “**HelloWorld.cs**” or “**Program.cs**”). You should open this files in a text reading program such as notepad to view its contents.

Likewise there should be one executable created “**HelloWorld.exe**”. Double click on it and see what happens.

IFN647 Advanced Information Retrieval and Storage

Week 1 Practical: Introduction to C#, File Handling

Activity 4: Getting Help

Online help for C# is available through MSDN for example:

- C# Reference: <https://msdn.microsoft.com/en-au/library/67ef8sbd.aspx>
- Using the Visual C# Development Environment: <https://msdn.microsoft.com/en-gb/enus/library/ms173063.aspx>

Since C# is part of the .NET Framework, it comes with an extensive library, referred to as the .NET Framework Class Library that can be accessed by developers. Information about how to use the .NET Framework Class Library is available from:

- .NET Framework Class Library: [https://msdn.microsoft.com/en-gb/enus/library/gg145045\(v=vs.110\).aspx](https://msdn.microsoft.com/en-gb/enus/library/gg145045(v=vs.110).aspx)

You can use the .NET Framework Class Library to find out how to use methods from classes in the library. For example, you can access the System.Console class (which you used in Activity 2) to read how to use the WriteLine and ReadLine methods.

- Console Class [https://msdn.microsoft.com/en-us/library/system.console\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.console(v=vs.110).aspx)

Visual Studio offers inline help. You'll notice that as you type in the name of a function, part of the function will autocomplete. This can be useful since it lists: the name of the function, the parameters required for function and how many times the function is overloaded. If you hover over the function name for a second the help command will pop up.

For example:

1. In the Main function of **HelloWorld**, start typing in **Console.ReadLine**, before you complete the function name it should autocomplete.

Use the help (either MSDN or inline help from Visual Studio) to:

1. Identify the purpose of the ReadLine Command.
2. Identify how many times the WriteLine Command is overloaded.

Similar inline help is provided for objects.

For example:

In the Main function of **HelloWorld** after the declaration of myString, start typing in the **myString** followed by a dot (e.g. "**myString.**"). This should bring up the list of functions for strings (since myString is a string).

IFN647 Advanced Information Retrieval and Storage

Week 1 Practical: Introduction to C#, File Handling

Activity 5: Reading a Text File

While console input/output is useful, most of the input/output operations we will perform in this unit will be file based. In this activity you will read in a text file stored on disk and then output its content to screen.

1. Create a new project as per Activity 2. Call this project "FileRead".
2. From Blackboard download the zip file **TextFiles.zip**.
3. Unzip **TextFiles.zip** into an appropriate location.
4. The "**TextFiles**" directory should have two Sub-directories: "**Activity 5**" and "**Activity 7**". The "**Activity 5**" subdirectory should contain a single text file called "**AlicePara.txt**". Take note of the filename and path of this text file.
5. Create a new project and select Console Application. Write an application that is able to parse the text in the "**AlicePara.txt**" file.
6. Using a **StreamReader** object read in all the text from "**AlicePara.txt**" and output it to the screen. You may need to check out the appropriate help document ([https://msdn.microsoft.com/en-us/library/system.io.streamreader\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.io.streamreader(v=vs.110).aspx)) to see which function to call.
7. Build and execute your application. Fixing any compiler errors. The output of should be similar to the image below.



Activity 6: Text Processing


We will also be performing a lot of text processing for this unit. Here, we want to add to the previous program by outputting the text in uppercase and then outputting the number of words in the file.

1. Convert the text from read from "**AlicePara.txt**" into uppercase using the appropriate string function. Visit [https://msdn.microsoft.com/en-us/library/system.string\(v=vs.110\).aspx](https://msdn.microsoft.com/en-us/library/system.string(v=vs.110).aspx) for help or use the in line help as in Activity 4 to help choose the right function.
2. Next split the string into individual words using the appropriate string function and store each word in an array. Using the appropriate array function to get a count of the number of words in the document.

IFN647 Advanced Information Retrieval and Storage

Week 1 Practical: Introduction to C#, File Handling

3. Print out the results from steps 1 and 2 to the screen. The output of should be similar to the image below.



```
file:///C:/temp/FileRead2/FileRead2/bin/Debug/FileRead2.EXE
ALICE WAS BEGINNING TO GET VERY TIRED OF SITTING BY HER SISTER ON THE
BANK, AND OF HAVING NOTHING TO DO: ONCE OR TWICE SHE HAD PEEPED INTO THE
BOOK HER SISTER WAS READING, BUT IT HAD NO PICTURES OR CONVERSATIONS IN
IT, AND WHAT IS THE USE OF A BOOK,' THOUGHT ALICE 'WITHOUT PICTURES OR
CONVERSATIONS?'
The number of words is 53
```

Activity 7: Directory Processing

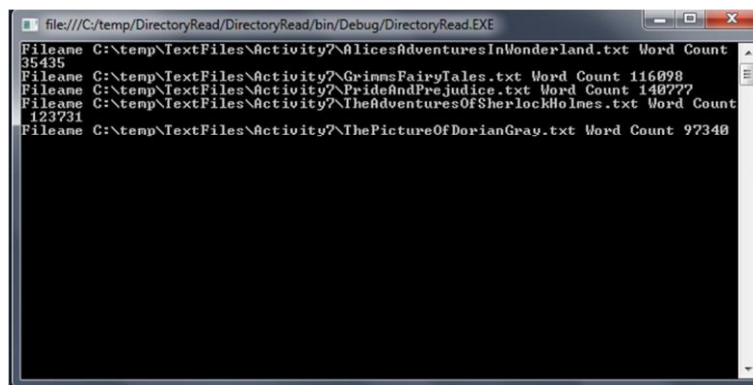
Often we would like to process multiple files in a directory. Here, we will use a method to traverse through a directory of text files, printing out the filenames and number of words in each of the files.

1. Download the **DirectoryRead.zip** file from Blackboard and unzip it to a suitable place.
2. Open the **DirectoryRead** solution in the unzipped directory. Either by Double clicking **DirectoryRead.sln** in the ~/Directory or selecting "**FILE->Open**" from the Menubar in Visual Studio.
3. Have a look at the **DirectoryRead** solution. Note that it contains three functions:
 - a. The Main function
 - b. WalkDirectoryTree – A function that traverses a directory tree
 - c. OutputFileDetails – A function that is currently empty, but which you will add to output some file details.
4. Note that the "**WalkDirectoryTree**" is called from the "**Main**" function. Have a look at the "**WalkDirectoryTree**" function and identify the two foreach code blocks. (*foreach (System.IO.FileInfo fi in files)* and *foreach (System.IO.DirectoryInfo dirInfo in subDirs)*). Currently they are almost empty. You will add one of the commands below to each of the, Decide which command should be added to which foreach code block and add them.

```
OutputFileDetails(name);
WalkDirectoryTree(name);
```
5. Call the "**WalkDirectoryTree**" function from you "**Main**" function – passing to it a pathname as a string. Note that the path that you want to pass it will be the "**Activity 7**" subdirectory you created in Activity 5.
6. Add code to the **OutputFileDetails** function so that for each file it prints out: the file and the number of words in the file. Again, you can use the **StreamReader** class to assist you.
7. Execute your application. The output of should be similar to the image below.

IFN647 Advanced Information Retrieval and Storage

Week 1 Practical: Introduction to C#, File Handling



```
file:///C:/temp/DirectoryRead/DirectoryRead/bin/Debug/DirectoryRead.EXE
Filename C:\temp\TextFiles\Activity7\AlicesAdventuresInWonderland.txt Word Count
35435
Filename C:\temp\TextFiles\Activity7\GrimmsFairyTales.txt Word Count 116098
Filename C:\temp\TextFiles\Activity7\PrideAndPrejudice.txt Word Count 140777
Filename C:\temp\TextFiles\Activity7\TheAdventuresOfSherlockHolmes.txt Word Count
123731
Filename C:\temp\TextFiles\Activity7\ThePictureOfDorianGray.txt Word Count 97340
```

8. Change the name of the traversed directory to its parent (i.e. the “TextFiles” directory). Execute the application. How is the output different to before?
9. Change the name of the traversed directory to one that does not exist. Execute the application. You will notice that an exception is generated and an error message is shown. Identify the lines of code that handle this exception.

Activity 8: Saving Files

This concludes the first week of practicals. Please externally save your files for future reference. The two best options are to either:

1. Save the files on a USB stick.
2. Zip the files and email them to yourself.

Getting a copy of Visual Studio for home use

In the S Block level 5 computer labs Visual Studio Community 2015 is installed. You can download a copy of Visual Studio Community 2015 from <https://www.visualstudio.com/products/visual-studio-community-vs>

The downloaded copy is on a 30 day trial, after which you will be required to have a Microsoft account to continue to use it as well you will need a Microsoft account to use Visual Studio on the workstations in the S Block Labs.

As an alternative you can download Visual Studio 2015 from the QUT DreamSpark site, <http://dreamspark.sef.qut.edu.au/>.

Log in to the DreamSpark site <http://dreamspark.sef.qut.edu.au/>

It is advisable you do this from the University labs, unless you have broadband! (Otherwise the download process may take a long time and your connection may fail)

IFN647 Advanced Information Retrieval and Storage

Week 1 Practical: Introduction to C#, File Handling

If using own laptop connect via a network cable to the QUT network either in S Block lab or Level 2 of the Library. Using a wireless connection to download is also likely to fail especially in the first weeks of semester when the wireless network is congested.

If you have not set up a password or do not have the email then the following may work:

Go to the Sign In page, click on the “Forgot username or password?” link to be sent an email containing a link to reset your password. If this does not work, then on the Sign In page click on the “Register” button and complete the necessary details.

Tutor for IFN647 – Contact Details

Anthony Gough: a1.gough@qut.edu.au