# JIT121 Programming Principles Tutorial 1

## Learning Objectives

1. Introduction to Visual Studio IDE
2. Small C# programs
3. Pre and postfix increment operators
4. Introduction to VS Debugging tool

## Assumptions

* + - * You are familiar with Lecture 1
      * That you are familiar with Visual Studio (VS) and have written a small Console program. To demonstrate the use of Visual Studio, we will code a Console application. Look at the document, **Starting VS first time.pdf.**

## Activities Overview

1. Key Concepts - variables
2. Small C# program
3. Increment Notation
4. Correcting logical error
5. Debugging with VS

### Activity 1: Key concepts

Extract the file **KeyConcepts\_VARIABLES.zip**, and run the exe file contained withinand complete the three exercises (one per tab on the GUI).Talk to me if you do not understand the correct answers.

# Activity 2: Small C# program

In this activity, you will develop an algorithm for a simple problem. Once you have developed an algorithm, you can code the algorithm to produce a C# program.

**The Problem:**

We want to write a program to convert temperatures from Fahrenheit to Celsius.

Your program should accept a temperature from the user in Fahrenheit and then print the corresponding temperature in Celsius.

The formula for converting Fahrenheit (f) to Celsius (c) is: c = 5/9 (f – 32)

**Solving the Problem:**

Before you start writing code, you should first develop an algorithm to solve this problem. You need to consider what inputs and outputs are required for your program. You also need to consider what information your program will need to store (i.e. variables).

When developing an algorithm to solve a problem, you need to break the problem into a small number of smaller problems. Then each of those steps can be broken down (or refined) into smaller steps and so on until you have simple enough steps that you can easily translate them into a programming language.

If it’s possible to misinterpret an algorithm in any way, then the algorithm is deficient.

Our goal is to express the algorithm in simple steps that correspond to operations in C# so that we can be sure that our algorithm is correct before we write any code.

Once you have considered your algorithm, write it in pseudocode. Pseudocode means writing in English which is not ambiguous. There are no rules for writing pseudocode except:

* Statements will be clear, accurate and concise
* It will not be a programming language
* It will be understandable to a non-programmer

Even if you do not yet know how some operation is done in C#, just write in pseudocode what you want to happen clearly and precisely. Then coding process is merely translating the pseudocode into C#.

# Activity 3: Increment Notation

**Operator Abbreviations Practice!**

Without using VS, determine the values of **n** and **m**, after each of the following has been executed:

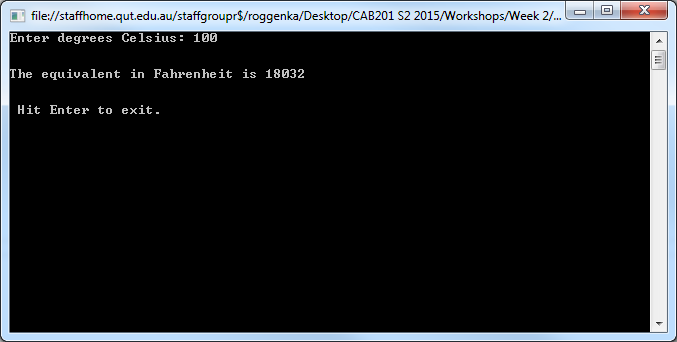
|  |  |
| --- | --- |
| Problem 1:  **int** n = 2;  n += 4;  Problem 2:  **int** n = 5;  **int** m = 3 \* ++n;  Problem 3:  **int** n = 2;  **int** m = n--; | Problem 4:  **int** n = 6;  n \*= 2;  Problem 5:  **int** n = 2;  **int** m = 4;  m -= n++;  Problem 6:  **int** n = 2;  **int** m = 1;  m += 2 \* n++; |

Check your answers using Visual Studio.

### Activity 4: Correcting a Logical Error

After writing the Fahrenheit to Celsius program from Tutorial 1 a student decided to write a program which would convert Celsius temperature to Fahrenheit.

The screenshot below shows what happens when the student entered 100 (degrees C). The student knew the result should be 212 degrees Fahrenheit but instead the output was 18032.



Extract the file **SimpleTempConversion\_Tutorial1 Exercise\_WITH BUG.zip** and open up the code file **program.cs**. Run the project. What is the error that this student has unknowingly introduced into their code? Correct the error and test that your correction actually gives the right result.

**Activity 5: Introduction to the Visual Studio Debugger**

The Visual Studio **Debugger** gives you a powerful way to see the run-time behaviour of your programs, and – in later workshops – help you to fix run-time problems in programs.

This section shows you how to use the Debugger to control and examine the execution of a program. At this stage, the main goal is to make sure that you understand how basic programming constructs – such as statement-sequences, if statements, and while statements – are executed by your computer. To do this, you will be working with some existing programs.

Once you’ve completed this section, you will know enough so that you can view the execution of your own program code, and check that its execution path is what you intended when you wrote that code.

See the document **Debugging Introduction.pdf**.