2: Arithmetic, Program Tracing & Decisions (week 3)

**KIT506 Programming Practical 2: Arithmetic, Program Tracing & Decisions**

[ [↶ Close all sections](https://mylo.utas.edu.au/content/enforced/214790-AW_EAI_17S2_15468_0_0_0_1_1/kit506_pracs/KIT506_Prac_02.html?d2lSessionVal=f4tSZ092i00c4rlmCOfiXiw7h&ou=214790&d2l_body_type=3) ] Only visible sections will be included when printing this document.

**Aims:**

* to gain experience with the arithmetic operators; and
* to implement a simple algorithm that utilises variables of C#’s *value* types and strings

**∨ 1 Reading and Tracing Exercises**

**1.1 Reading arithmetic**

Suppose that the following statements were placed in the Main method of a new C# program. What would happen when the program is executed? Answer the questions below.

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | int a, b, c;  a = 10;  b = 8;  c = a + b;  a = c / b; |

What value does each variable have after each of these lines has been executed? (Create a table like the one below, or take a copy of the code and write down the values next to each line.)

| **Line** | **a** | **b** | **c** |
| --- | --- | --- | --- |
| 3 | 10 |  |  |
| 4 | 10 | 8 |  |
| 5 | 10 | 8 | 18 |
| 6 | 2 | 8 | 18 |

Enable Solutions

**1.2 Constants, assignment and arithmetic**

Suppose the following statements were placed in the Main method of a different C# program. What would happen as the program is executed?

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | const int PRIME = 31;  const int DIVISOR = 3;  int a, b;  a = PRIME;  b = 2;  a = (a + b) / DIVISOR; |

What value does each variable after each of the indicated lines has been executed?

| **Line** | **PRIME** | **DIVISOR** | **a** | **b** |
| --- | --- | --- | --- | --- |
| 1 | 31 |  |  |  |
| 2 | 31 | 3 |  |  |
| 5 | 31 | 3 | 31 |  |
| 6 | 31 | 3 | 31 | 2 |
| 7 | 11 | 3 | 31 | 2 |

After lines 1 and 2, can the values of PRIME and DIVISOR ever change?

**Tip:** *When reading someone else’s code for the first time, consider creating a tracing table to help you follow what’s going on.*

**1.3 What’s the output?**

Suppose the following lines of code were placed in the Main method of a C# program (and were the only contents of the Main method). What would the output of the program be?

int height = 176; *//height in cm*

double mass = 67.4; *//mass in kg*

double bmi;

bmi = mass/Math.Pow(height/100.0, 2);

Console.Write("Calculating BMI using height of " + height + " cm "); //176

Console.WriteLine("and weight of " + mass + " kg."); //67.4

Console.WriteLine("BMI: " + bmi); //116.689…

**∨ 2 Newspaper Article Writer**

The Console.ReadLine() method allows your console application to read a line of text entered by the user. Create a new program with the following statements in the Main() method:

string name;

Console.Write("Enter your name: ");

name = Console.ReadLine();

Console.WriteLine("Name is " + name);

That’s quite boring, so modify your program as follows:

1. Declare two more variables, int age and string country.
2. Add more user prompts and Console.ReadLine()s to read the user’s age and country into those variables. When reading the int you will need to use Int32.Parse(Console.ReadLine()), which will convert the text the user entered (that *looks* like a number but is just text) into the internal binary representation of an integer value.
3. Modify the message at the end to be similar to the following, where each *italicised* value comes from the variable with that name:

*name* (*age*), of *country*, was this year’s surprise Eurovision Song Contest winner. This is *country*’s first win in the contest.

**Tip:** In addition to Int32.Parse(Console.ReadLine()) for reading integer values from the console, you can read real-valued numbers (type double) with Double.Parse(Console.ReadLine()) and bool values (**true** or **false**) with Boolean.Parse(Console.ReadLine()).

**∨ 3 Making some simple decisions**

Create a new program that prompts the user for a positive integer *n* and stores it in a variable. Then apply the following rules:

* if *n* is even your program should divide it by 2 and store the result back in *n* (that is, overwrite the value of *n*);
* otherwise it should multiple *n* by 3 and add 1, and store the result back in *n*

And then display the new value of *n*. Hint: you can determine if a number is even by checking if the remainder from dividing it by 2 is equal to zero (*n* % 2 == 0 in C#).