MAU11601:Introduction To Programming - Tutorial 2

Liam Burke
School Of Mathematics, Trinity College Dublin

October 6, 2022

Introduction - The Factorial Function

In this tutorial you will implement three versions of a the same function to compute the factorial of a positive integer n. The factorial function is defined as

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot (n-2) \cdot (n-1) \cdot n = \prod_{i=1}^{n} i.$$

From this definition we see that

$$n! = n \cdot (n-1)!,$$

and we also define $0! = 1$. Some examples are

$$3! = 3 \cdot 2 \cdot 1 = 6$$

$$6! = 6 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1 = 720$$

Q1

Create a MATLAB run scrip *run.m* which defines an integer n and calls three versions of a factorial function called *my_factorial1*, *my_factorial2* and *my_factorial3*. Create these three different functions in three separate files, each with the file name saved as the same name given to the function. Each function should have one input, the integer n, and one output, the factorial of n. The three functions should be implemented in the following ways:

- *my_factorial1* should use a *for loop*.
- *my_factorial2* should use a *while loop*.
- *my_factorial3* should use *recursion* (see below).

Use the built in MATLAB *error* function with appropriate error message to throw an error when a negative integer is inputted into each function. After all function calls have completed in the *run.m* script, check that all three answers outputted by the function are correct. Check this using the built in MATLAB *factorial* function. Use the same conditional statement for all three functions (as opposed to checking the answer for all three functions separately). If all three implementations are correct, use the built in MATLAB *fprintf* function to print "*All three versions are correct!*". If not, the code should print "*Not all functions are working correctly!*".

Test your code using the example $10! = 3,628,800$.

Use MATLAB comments to explain what your code is doing. Give a brief summary of the purpose of each function in each function file.

Recursion

A recursive function is a function that iteratively calls itself and performs an iterative task in a similar manner as *for* and *while loops*. In many cases however, it is often preferred to use recursion such as when there is many different branches and conditions the iteration can take which make it more complicated than a simple iteration. An example of this involves searching through a file system. Every recursive function has two components: a base case and a recursive step. The base case stops the function from calling itself forever and represents a known condition for which the process has completed its task. The recursive step is the set of all cases where a recursive call, or a function call to itself, is made.

The body of code in your *my_factorial3* function should also contain a call to *my_factorial3*.