Math Functions and Programming Language Functions: A Comparison

All modern and most pre-modern programming languages support functions in some form. The programming language concept of function is based on the mathematical concept of function. But there are significant differences.

Let's consider a typical function declaration in CMIS 102-style pseudocode

Declare WierdProcess(String s, Integer x, Float y) Returns String {….}

or in Java,

String WierdProcess(String s, int x, float y) {….}

Each of the data types String, Integer, and Float is a very large finite set. What we have here, in math notation, is something that looks like a function:

WierdProcess: String x int x float → String

for which the domain is the Cartesian product String x int x float, and the codomain is String.

But there are several ways in which a computer program function can fail to be a true function in the mathematical sense. These include

- The function may fail for some inputs, causing the program to crash, or an exception to be thrown.
- The output of the function may depend on data other than that provided as input parameters. This is typically the case for input functions, and for "methods" of a class whose return value depends on the values of class members in an object.
- The function may do other things in addition to returning a value, such as writing to memory or performing an output operation. These are called *side effects*.
- The function may not return any value at all (a "void" function) – it exists for its side effects.

In summary, a programming language function qualifies as a function in the mathematical sense only if

- It returns a value for every possible combination of input parameter values
- The value returned depends only on the input parameter values
- There are no side effects.