Jonathan Davis

October 08, 2017

CSC 4010 – 800

7L in 7W – Io

Differences:

Io is a prototype language, meaning every object is a clone of another (everything stems from the base Object object), while C++ is essentially a systems programming language in which instances and classes are two distinct things.

Every type of object can be changed in Io, meaning the user is able to change the very Io language itself by changing objects such as those in the Operator Table or Io's method_missing rules. In C++, you cannot change any preexisting elements of the language (primitives, functions, etcs.) and cannot change user-created objects and structures without changing the source code and recompiling.

In C++, you must use a semicolon to end a statement or else you will get a compiler error. In Io, such syntax is not necessary at the end of a line.

Code written in C++ must be compiled before being able to run, while Io does not due to it being an interpreted language.

In C++, you must declare variables as a certain type (int, char, string, etc.) and that variable's type can never be changed. In IO, any object can be assigned any type of value at any time.

Similarities:

Both C++ and Io have for and while loops available, and both languages have a form of iterator at their disposal. (A foreach loop in Io, <iterator> in C++)

Both languages feature object inheritance, with an object's child inheriting its parent's attributes. However, in Io all clones (children) of an object (parent) can change values in the object's slots that will be reflected among all the clones. To avoid this, you must use the "self" keyword in Io when creating a slot.

Both C++ and Io feature arrays, however an array in C++ must be homogenous of the type it is declared to be of, while in Io an array (a list) can contain heterogenous items. Both languages allow the use of accessing a specific element in an array, "array[index]" in C++ and "listName at(index)" in Io.

Both C++ and Io have comparison operations that can utilize the values of true, false, and null (nil in Io). However, as was also the case in Ruby, a value of 0 in Io evaluates to true, while 0 would be false in C++.

Much like how C++ can pass values by reference instead of by value, Io passes messages themselves as well as the context of the message. However, this is the only way Io handles this; it cannot pass by value as C++ can.