Jonathan Davis

September 19, 2017

CSC 4010 – 800

7L in 7W – Ruby

The most obvious difference between Ruby and C++ is that the Ruby language can be (and is almost always) interpreted, while C++ needs to be compiled.

Both Ruby and C++ have classes and objects. However, Ruby is a pure object-oriented language while C++ is procedural with the ability to also be object-oriented.

C++ allows classes to inherit from multiple parents, while Ruby allows classes to inherit from only one parent (called the superclass).

Both languages use = as an assignment statement and == to test for equality. Both languages also use logical operators in mostly the same way, with an exception: In Ruby, the interpreter executes code only until the value of a test is clear (when using && or ||); to execute an entire expression, you must use a single & or |.

In Ruby, everything but nil and false evaluate to true; this means 0 evaluates to true, unlike in C++ where 0 would evaluate to false.

Both Ruby and C++ utilize arrays with the notation “variable_name[index]”, however Ruby has a lot of syntactic sugar for arrays such as the ability to sort them via “variable_name.sort” or access the last element in an array without knowing its size by using “variable_name[-1]”. Also, unlike in C++, arrays in Ruby do not have to be homogeneous.

Ruby does not do type checking until attempting to execute code (dynamic typing), unlike C++ which typically uses static typing and would catch mismatches at compilation time before any code can be executed.

You can implement functions in both Ruby and C++, however there is no such thing as a void function in Ruby, as every function in Ruby returns something (whether it be an explicit return or the value of the last expression in the function).

Modules in Ruby more or less serve the same purpose that namespaces do in C++.