Judy Wang, Tino Nguruve, Linh Le
Deepak Kumar
Principles of Programming Languages
26 October 2020

Swift Structure, Types and Operators, Variables and Declaration, and Control Flow

Structure of a Swift Program

Similar to many other programming languages, one of the first things that needs to be done is importing the packages you will need for your program. Some common packages to import are Foundation and UIKit, which already has Foundation within it's package. People can also import Swift, but it is not necessary. The Swift package allows people to print words, for example a person can explicitly say Swift.println("") or simply say println(). From there you are able to use the word "func" to begin functions, which is similar to go lang. Swift does not need semicolons, but a programmer could if they wanted to and would not get errors. In addition, brackets are very important in Swift and play a similar role in other languages like Java in order to determine scopes.

Types and Operators in Swift

Swift has built in types such integers(**int**), strings, boolean (**bools**), and floating point types. For integers, Swift is similar to go lang where you can establish how many bytes you want your integers to have ranging from Int8-Int64. Similar to C, there are also unsigned integers where it excludes negative numbers. This means you have more memory to make longer numbers. This can be represented as UInt8-UInt64. For floating point types, floats have allotted 32 bytes, while doubles are allotted 64 bytes. The operators that can be used for floating point types and integers are addition (+), subtraction (-), multiplication (*), division (/) and modulo (%). For boolean values or **bool**, they are represented as true or false, which are utilized in condition statements. There are many ways that conditions can be used, for example: Equal to (a

== b) , Not equal to (a != b), Greater than (a > b), Less than (a < b), Greater than or equal to (a >= b), Less than or equal to (a <= b) where a and b can be represented as a boolean type. Strings can be represented as **string**. String literals are encapsulated by quotations. Similar to booleans, strings can also use comparator operations to compare strings. If you wanted to see the length of a string, the syntax would be variableName.count.

Variables and Declarations

As with any other languages, constants and variables in Swift associate a name with a value of a particular type. Once a constant or variable of a certain type is declared, users cannot declare it again with the same name, or change it to store values of a different type. Nor can you change a constant into a variable or a variable into a constant. The names cannot begin with a number, nor can it contain whitespace characters, mathematical symbols, arrows, private-use Unicode scalar values, line- and box-drawing characters. Allowing names to include Unicode characters means emoji characters can also be used for variable names in Swift, but this is not recommended. The convention of Swift naming is camelCase. Variables must either be declared to be only a type with the syntax **var <name>: <type>** or initialized a specific value **var <name>: <type> = <value>** before they're used, while constants should be initialized with keyword **let**. Swift's type inference allows the variables to be initialized with the syntax **var <name> = <value>** without the type annotation, and instead lets the compiler infer the type of data to store based on the initial value assigned. Multiple variables of the same type can also be declared with **var <name1>, <name2>, ...: <type>,** or initialize individual values with **var <name1> = <value1>, <name2> = <value2>,...**. Additionally, Swift is a type-safe language, thus encouraging the user to be explicit about the types of values the variables and constants will

hold. It will give error messages if a type mismatch is found during compiling between the original type declared for a variable and the value the user tries to pass to it.

Control Structures in Swift

As compared to C-like languages Swift's control flow structures are considered more robust. Cases can match many different patterns, including interval matches, tuples, and casts to a specific type. Swift's control flow statements include while loops, if, guard and switch statements to execute the code based on certain conditions. It is important to point out that Swift's conditional statements are similar to Go's conditional statements. It also has statements such as break and continue to transfer the flow of execution to another point in the code. To iterate over arrays, dictionaries, ranges, strings, and other sequences, Swift uses for-in loops which makes everything easier. In order to iterate through ranges, it uses the syntax 1...5 within the for loop. This means that it will iterate through 1-5 inclusive. Matched values in a switch case can be bound to temporary constants or variables for use within the case's body, and complex matching conditions can be expressed with a where clause for each case.

Work Cited

https://www.oreilly.com/library/view/ios-8-programming/9781491909645/ch01.html

https://www.programiz.com/swift-programming/operators

https://medium.com/coding-blocks/data-types-in-swift-where-it-all-starts-c1311fa93368#:~:text=Swift%20offers%20a%20collection%20of,found%20in%20most%20programming%20languages.