1. Import **Framework** (example **import cocoa** makes all Cocoa libraries, APIs, and runtimes that form the development layer for all of OS X, available in Swift.)

|  |  |  |
| --- | --- | --- |
| **Type** | **Typical Bit Width** | **Typical Range** |
| Int8 | 1byte | -127 to 127 |
| UInt8 | 1byte | 0 to 255 |
| Int32 | 4bytes | -2147483648 to 2147483647 |
| UInt32 | 4bytes | 0 to 4294967295 |
| Int64 | 8bytes | -9223372036854775808 to 9223372036854775807 |
| UInt64 | 8bytes | 0 to 18446744073709551615 |
| Float | 4bytes | 1.2E-38 to 3.4E+38 (~6 digits) |
| Double | 8bytes | 2.3E-308 to 1.7E+308 (~15 digits) |

1. Typealias
2. import Cocoa
3. typealias Feet = Int
4. var distance: Feet = 100
5. println(distance)

4) **Operator:** An operator is a symbol that tells the compiler to perform specific mathematical or logical manipulations. (+, -, =, ==)

5) **Function Type:**

func sum(a: Int, b: Int) -> Int {

return a + b

}

var addition: (Int, Int) -> Int = sum

println("Result: \(addition(40, 89))")

func another(addition: (Int, Int) -> Int, a: Int, b: Int) {

println("Result: \(addition(a, b))")

}

another(sum, 10, 20)

6) **Nested Func**

func calcDecrement(forDecrement total: Int) -> () -> Int {

var overallDecrement = 0

func decrementer() -> Int {

overallDecrement -= total

return overallDecrement

}

return decrementer

}

let decrem = calcDecrement(forDecrement: 30)

println(decrem())

**7) Closure**

{(parameters) -> return type in

statements

}

let divide = {(val1: Int, val2: Int) -> Int in

return val1 / val2

}

let result = divide(200, 20)

println (result)