# 数据类型与运算

本章主要介绍Swift4.0的常量、变量、基本数据类型、以及运算符等一些基础知识。

## 2.1变量与常量

Swift语言中的基本数据类型，按其取值可以分为常量与变量，在程序执行过程中，其值一旦设定不能够发生变化的量称为常量，其值能够发生变化的量称为变量。

### 2.1.1声明常量和变量

常量和变量必须在使用前声明，使用的声明关键字也不同，用let来声明常量，用var来声明变量。

let pi:Double=3.1415926 //不可修改

var schoolName:String = “HDU” //可以修改

可以用println()函数来输出这个常量和变量

println(pi) //输出 “3.1415926”

Println(schoolName) //输出 “HDU”

对于常量的声明时，需要定义其数据类型，但是读者可以尝试一下不定义数据类型也能够编译成功。这是由于编译器xcode支持对Swift语言的类型自动推断，当声明变量时如果直接给变量赋初值，则编译器会根据赋值的类型来确定变量的类型，之后变量的类型将不可更改。

### 2.1.2变量与常量的命名规范

Swift语言中，常量和变量的命名规则十分宽泛，可以包括Unicode字符和数字，需要注意的是，不可使用预留关键字来作为常量或者变量的名称，例如let、var这类的关键字不可作为量值名来声明。另外，常量和变量的命名不可以数字开头，空格、数字符号、制表符、箭头等符号也不可用在命名中。

和其他面向对象的语言一样，Swift也可以用中文进行变量的命名，示例如下：

var 杭电 = “杭电”

可以使用穿插数字进行命名，注意数字不能作为开头：

//含有数字的命名

var name1 = “雨燕”

可以使用下划线进行命名：

var \_swift\_ = “swift”

虽然Swift支持的命名方式十分广泛，但在实际开发中，良好的代码风格可以大大提高编码效率与代码的可读性，Swift语言官方文档采用驼峰命名的方式，所谓驼峰命名是指以单词进行名称的拼接，名称的首字母一般为小写，之后每个单词的首字母大写，其他字母均小写，示例如下：

//驼峰命名

var userName = “雨燕”

注： Swift中的命名也有一些约定俗成的规则，例如常量与变量属性首字母会小写，类名、枚举名、结构体名首字母会大写。

## 2.2关于注释

对于程序中的代码是给编译器看的，而注释是写给开发者自己看的注释性文本，在代码进行编译时，注释语句是不会被编译进工程中，使用注释能够让别人读懂你的代码。而Swift语言中采用和C语言类似的注释方式，使用//符号来注释单行内容，同时也可以用/\*开头，以\*/结尾的方式进行多行注释，示例如下：

//单行注释

/\*

多行注释

多行注释

多行注释

\*/

Swift语言的注释还有一个十分有趣的特性，即可以进行注释的嵌套，示例如下：

//单行注释//注释中的注释

/\*

多行注释

/\*

注释中的注释

\*/

注释

注释

\*/

## 2.3数据类型(布尔、Int、浮点、字符、字符串)

本节主要向读者介绍布尔型、整型、浮点型、字符型以及字符串类型在swift中的应用。

### 2.3.1 布尔型数据

布尔类型很多时候也叫做逻辑类型，对于熟悉Java编程语言的读者可能会了解，在Java语言中，Bool类型其实并非严格意义上的逻辑布尔类型，Java中可以使用0与非零来表达逻辑假与逻辑真。而在Swift语言中则不同，Swift语言的Bool类型十分严格，只有true和false两种值，分别表示真和假。同样，在Swift语言的条件语句以及需要进行逻辑判断语句中，所使用的条件表达式的值也必须为Bool类型。下面有两个例子：

例1：

var bool1 =true

if bool1{

println(“oh,right,boo1 is true!”)

}else{

println(“oh,no,boo1 not true!”)

}

//输出 “oh,right,bool1 is true”

例2：

var bool2 = 1

if bool2{

//这个例子不会通过编译，会报错

}

### 2.3.2 整数型数据

Swift语言中的整型数据分为有符号整型数据与无符号整型数据。所谓有符号与无符号，通俗的理解即为分正负号与不分正负号。

Swift提供了8，16，32和64位的有符号和无符号整数类型。这些整数类型和C语言的命名方式很像，比如8位无符号整数类型是UInt8,32位有符号整数类型是Int32。就像Swift的其他类型一样，整数类型采用大写命名法。

你可以访问不同整数类型的min和max属性来获取对应类型的最大值和最小值：

let minValue = UInt8.min //minValue为0，是无符号8位整数类型的最小值

let minValue = Int8.min //minValue为-128,是有符号8位类型的最小值

let maxValue = UInt8.max //maxValue为255，是无符号8位整数类型的最大值

let minValue = Int8.max //minValue为127,是有符号8位类型的最大值

同理可得UInt16,UInt32,UInt64的最大值与最小值，请读者自行验证，在此不再赘述。

### 2.3.3 浮点型数据

浮点型数据用来表示一些小数，浮点型数据分为单精度浮点型与双精度浮点型。分别用Float与Double表示。

Swift语言中也支持使用科学计数法来表示数字，在十进制中使用e来表示10的n次方，在十六进制中使用p来表示2的n次方，示例代码如下：

var num = 1.31e4 //1.13\*(10^4) = 13100

var num2 =0x1p2 //1\*(2^2) = 4

### 2.3.4 字符类型

字符类型用来表示单个字符，如数字字符、英文字符、符号字符和中文字符等都可以使用字符类型来表示，

类似于C语言的Char,Swift语言中使用Character来描述字符类型，Character类型占9个字节的内存空间，String类型占24个字节的内存空间。在Swift中可以使用MemoryLayout枚举来获取某个类型所占用的内存空间，其单位为字节，示例如下：

MemoryLayout<String>.size //24个字节 获取String类型占用的内存空间

创建字符的两种方法，示例如下：

var c :Character = “a” //直接创建出一个字符

var c2=Character(“a”) //使用构造方法创建字符类型变量

### 2.4字符串

字符串类型顾名思义为一串字符的组合，在开发中被广泛的应用。

创建字符串

var string1 = “hello world”

print(string1)

var string2 =String(“hello world!”)

print(string2)

以上程序执行输出结果：

hello world!

hello world!

空字符串

var string1=””

if string1.isEmpty{

print(“string1是空字符串”)

}else{

print(“string1不是空字符串”)

}

以上程序执行输出结果为：

string1是空字符串

## 2.5运算符

运算符是一个符号，用于告诉编译器执行一个数学或逻辑运算。

2.5.1赋值运算符

赋值运算符应该是在编程中出现频率最高的运算符之一，在对于一些量进行赋值时，都要使用到赋值运算符”=”。需要注意，”=”在Swift语言中是赋值运算符，并不是相等运算符，对于一些编程初学者，很容易将相等运算符”==”与赋值运算符”=”混淆使用。示例如下：

//元组赋值

var tuple1=(1,2,3)

//Bool赋值

var bol=true

2.5.2区间运算符

在C语言中，来描述一个变量的取值范围一般会用(i>0 && i<10)。但是在Swift中有新的写法，并且常用于for-in循环结构中，开发者常常会使用区间运算符来定义循环次数，示例如下：

//a...b为闭区间写法

for index in 0...3{

print(index)

}

输出结果： 0 1 2 3

//a..<b为左闭右开区间

for index in 0..<3{

print(index)

}

输出结果：0 1 2

2.5.3算术运算符

算术运算符用于进行一些基本的数学运算，例如常见的加、减、乘、除、取余等。但是在Swift4中已经取消了++、--。

故对于自增、自减，在swift中时使用赋值运算符与算术运算符相结合的复合赋值运算符。示例如下：

var tmp=1

//加赋值复合运算

tmp+=3 //tmp = tmp + 3

//减赋值复合运算

tmp-=3 //tmp = tmp-3

2.5.4条件运算符

条件运算符(三目运算符)是一种三元运算符，其可以简便实现代码中的条件选择逻辑。例如如下代码为一个简单的条件选择语句示例：

var a =1

var b =2

if a>b{

print(“a>b”)

}else{

print(“b>=a”)

}

上面代码对变量a和b进行了比较，并将比较结果进行打印，如果使用条件运算符，上面的逻辑可以简写成如下形式：

print(a>b ? “a>b”:”b>=a”)

即三目运算符的基本语法格式：条件？成立执行：不成立执行