## 程序流程控制

Swift提供了多种流程控制结构，包括可以多次执行任务的while循环，基于特定条件选择执行不同代码分支的if、guard和switch语句，还有控制流程跳转到其他代码位置的break和continue语句。

Swift 还提供了for-in循环，用来更简单地遍历数组（Array），字典（Dictionary），区间（Range），字符串（String）和其他序列类型。

Swift 的switch语句比 C 语言中更加强大。case 还可以匹配很多不同的模式，包括范围匹配，元组（tuple）和特定类型匹配。switch语句的 case 中匹配的值可以声明为临时常量或变量，在 case 作用域内使用，也可以配合where来描述更复杂的匹配条件。

### 3.1 for-in 循环

如果你有其他语言的基础，你一定会对大名鼎鼎的for循环非常熟悉。可是在Swift语言中，传统的三段式for循环已经被移除了，取而代之的是for-in循环。

如果我们用for-in循环打印0到4

//代码段 3.1 for-in 打印 0...4

//代码段 0..<5

代码中的(0...4)和(0..<5)被称作区间操作符，其中(0...4)是闭区间操作符，表达的是一个左闭右闭的区间，而(0..<5)是半开区间操作符，表达了一个左闭右开的区间。索引index第一次被赋值为区间中的第一个数字0，然后通过print语句打印出0这个数字。随后，index被赋值为区间中的下一个数字1，再次打印，依次类推，打印0到4这5个数字。最后由于区间(0..<5)不包含5，所以不会打印5。

上面的例子中，index是每次循环遍历开始时自动被赋值的常量。这种情况下，index不需要使用let关键字进行声明。

有时，我们并需要index这个索引，我们可以用\_代替index，从而忽略这个值。

//代码段3.2 打印 3遍 hello world

swift中的区间操作符一定要是一个合法的区间，也就是左边界要小于等于右边界。如果我们要实现逆序打印该怎么做？可以调用reverse()方法。

//代码3.3 逆序打印 4-0

### 3.2 while 循环

while循环会一直运行一段语句直到条件变成false。while循环非常适合使用在第一次迭代前，迭代次数未知的情况下。

Swift提供两种while循环的形式：

1. while循环，在每次执行循环体之前判断条件是否符合。
2. repeat-while循环，在每次执行循环体之后判断条件是否符合。

#### 1.while循环

while循环先判断条件再执行语句。如果条件为true，会重复运行一段语句，直到条件变为false。

//代码3.4 依旧是打印0-4

#### 2.repeat-while循环

repeat-while循环不像for-in和while循环在循环体开始执行之前判断条件语句，而是在循环体执行结束之后判断条件是否符合。

//代码3.5 repeat-while打印0~4

请注意，由于条件表达式出现在循环的尾部，所以循环体在条件被检测之前至少被执行一次。

如果条件为true，控制流会跳转回上面的repeat，然后重新执行循环体，直到条件变为false为止。

### 3.3 条件语句

根据特定的条件执行特定的代码通常是十分有用的。当错误发生时，你可能想运行额外的代码；或者，当值太大或太小时，向用户显示一条消息。要实现这些功能，你就需要使用条件语句。

Swift 提供两种类型的条件语句：if语句和switch语句。通常，当条件较为简单且可能的情况很少时，使用if语句。而switch语句更适用于条件较复杂、有更多排列组合的时候。

#### 1.if-else

If语句最简单的形式就是只包含一个条件，只有条件为true时，才执行相关代码。

// 有if

我们可以通过配合else从句，实现二选一执行。也就是当条件为false的时候，执行else语句。

//if else的例子

我们还可以通过多个if-else链接的方式，来实现更多的分支。

//if else 更多的例子

#### switch

switch语句会尝试把某个值与若干个模式（pattern）进行匹配。根据第一个匹配成功的模式，switch语句会执行对应的代码。当有可能的情况较多时，通常用switch语句替换if语句。

switch语句最简单的形式就是把某个值与一个或若干个相同类型的值作比较：

//switch的例子 简单的比较 a case a case A 其他

switch语句由多个 case 构成，每个由case关键字开始。为了匹配某些更特定的值，Swift 提供了几种方法来进行更复杂的模式匹配，这些模式将在本节的稍后部分提到。

与if语句类似，每一个 case 都是代码执行的一条分支。switch语句会决定哪一条分支应该被执行，这个流程被称作根据给定的值切换(switching)。

switch语句必须是完备的。这就是说，每一个可能的值都必须至少有一个 case 分支与之对应。在某些不可能涵盖所有值的情况下，你可以使用默认（default）分支来涵盖其它所有没有对应的值，这个默认分支必须在switch语句的最后面。

在swift中，当匹配的case分支中的代码被执行完毕后，会自动终止switch语句，不会继续执行下一个case分支。这也就是说，不需要在 case 分支中显式地使用break语句。

### 3.4 控制转移语句

控制转移语句可以改变你代码的执行顺序，通过它们可以实现代码的跳转。Swift中主要有3种控制转移语句: 1.continue 2.break 3.fallthrough

1. Continue

Continue语句的作用是告诉一个循环体立刻停止本次循环，重新开始下次循环。结束的仅仅是一次循环，而不是整个循环体。

//continue的代码

1. Break

Break语句的作用是中断当前循环，结束整个循环体。

//break的代码

1. fallthrough

fallthrough 语句可以让 case 之后的语句按顺序继续运行，且不论条件是否满足都会执行。

//一个含有fallthrough的例子

### 3.5 提前退出(guard)

guard语句和if语句有点类似，都是根据其关键字之后的表达式的布尔值决定下一步执行什么。但与if语句不同的是，guard语句只会有一个代码块，不像if语句可以if else多个代码块。

guard，顾名思义，就是守护。guard语句判断其后的表达式布尔值为false时，才会执行之后代码块里的代码，如果为true，则跳过整个guard语句，guard像守卫一样，在代码中设置检查点，当不满足条件时，程序执行就会被拦截下来。

//guard 示例代码

func check(age : Int){

    guard age <= 6 else {

        print("对不起，你已经不是宝宝了！")

        return

    }

    print("欢迎来到儿童乐园！")

}

check(age: 19)

//输出：对不起，你已经不是宝宝了！

check(age: 4)

//输出：欢迎来到儿童乐园！