Basic Operators

`=` 没有返回值，以避免和 `==` 混用

算数操作符会检测并避免 overflow

swift 提供了范围操作符 `a..<b`（左闭右开区间）和 `a...b`（闭区间）

terminology

- 一元操作符，分为前缀和后缀
- 二元操作符，中缀
- 三元操作符，只有一个 `a ? b : c`

赋值操作符

`=` 可以初始化或更新一个变量的值

如果右边是一个 tuple 的话，可以分解到多个变量或常量

```
let (x, y) = (1, 2)
// x is equal to 1, and y is equal to 2
```

`=` 没有返回值，用作 if 的条件时会产生错误

算术操作符

`+ - * /`

不允许溢出，如果需要溢出功能可以使用溢出操作符，如 `a &+ b`

`+` 同样支持字符串的连接

`%` 取余

一元的前缀操作符 `-` 可以取负，而 `+` 不改变变量的值

可以像 C 一样将算数操作符和赋值结合在一起使用 `+=`

比较操作符

swift 提供了包括等于不等于大于小于等 6 种操作符

同样有 `===` 和 `!==` 用于测试两个对象的引用是否指向同一实例

可以比较两个类型和元素数量相同的 tuple，从左往右依次比较。能够比较的前提是操作符可以作用于 tuple 的每个分量

三元操作符的语义与 C 的相同

Nil-Coalescing Operator

`a ?? b` 用于展开 optional，语义上等同于 `a != nil ? a!: b`

要求 a 必须是 optional，而 b 的类型与 a 匹配

有短路效应，如果 a 非 nil 则 b 不会被计算

Range 操作符

分为两种

- close range：`a...b`，定义了一个 `[a,b]` 的区间
- half-open range：`a..<b`，定义了一个 `[a,b)` 的区间

可以用在 for-in 循环

对于数组下标，也可以用单边的 range，如

```
for name in names[2...] {
    print(name)
}

for name in names[...2] {
    print(name)
}

for name in names[..<2] {
    print(name)
}
```

单边的 range 同样可以用在循环中

逻辑操作符

支持逻辑与或非

可以组合使用，使用时最好以括号来明确优先级