# Real World Haskell中文版

## 第一章：入门

Prelude> :set prompt "ghci>"

ghci>

ghci> :module + Data.Ratio

ghci> True && 1

<interactive>:1:8:

No instance for (Num Bool)

arising from the literal `1' at <interactive>:1:8

Possible fix: add an instance declaration for (Num Bool)

In the second argument of `(&&)', namely `1'

In the expression: True && 1

In the definition of `it': it = True && 1

* “No instance for (Num Bool)” 告诉我们**ghci**尝试解析数字1为Bool类型但是失败。
* “arising from the literal `1’” 表示是由于使用了数字1而引发了问题。
* “In the definition of `it’” 引用了一个**ghci**的快捷方式。我们会在后面提到。
* Prelude> :set +t
* Prelude> 'c' *-- 输入表达式*
* 'c' *-- 输出值*
* it **::** Char *-- 输出值的类型*
* Prelude> "foo"
* "foo"
* it **::** [Char]

Prelude Data.Ratio> 3 + 2

5

Prelude Data.Ratio> :**type** it

it **::** Integer

Prelude Data.Ratio> :**type** 3 + 2

3 + 2 **::** Num a **=>** a

## 第二章：类型和函数

Haskell中每个函数和表达式都有各自的类型T。一个值的类型标识了它和该类型的其他值所共有的一簇属性（property）。

Haskell 中的类型有三个有趣的方面：首先，它们是强（strong）类型的；其次，它们是静态（static）的；第三，它们可以通过自动推导（automatically inferred）得出。

:i Integral Integer

函数应用的优先级比操作符要高

Haskell 有一个特殊的类型 () ，这种类型只有一个值 () ，它的作用相当于包含零个元素的元组，类似于 C 语言中的 void

Haskell 的函数应用是左关联的。

Prelude> lines "the quick**\n**brown fox**\n**jumps"

["the quick","brown fox","jumps"]

Side effect 副作用指的是，函数的行为受系统的全局状态所影响。

Tip：x:xs，这个xs是指复数个x

Haskell 是一门以表达式为主导（expression-oriented）的语言。在命令式语言中，代码由陈述（statement）而不是表达式组成，因此在省略 if 语句的 else 分支的情况下，程序仍是有意义的。但是，当代码由表达式组成时，一个缺少 else分支的 if 语句，在条件部分为 False 时，是没有办法给出一个结果的，当然这个 else 分支也不会有任何类型，因此，省略 else 分支对于 Haskell 是无意义的，编译器也不会允许这么做。

在使用严格求值的语言里，函数的参数总是在应用函数之前被求值。以 isOdd 为例子：子表达式 (1 + 2) 会首先被求值，得出结果 3 。接着，将 3 绑定到变量 n ，应用到函数 isOdd 。最后， mod 3 2 返回 1 ，而 1 == 1 返回 True 。

Haskell 使用了另外一种求值方式 —— 非严格求值。在这种情况下，求值 isOdd (1 + 2) 并不会即刻使得子表达式 1 + 2 被求值为 3 ，相反，编译器做出了一个“承诺”，说，“当真正有需要的时候，我有办法计算出 isOdd (1 + 2) 的值”。 非严格求值通常也被称为惰性求值。

用于追踪未求值表达式的记录被称为块（thunk）。这就是事情发生的经过：编译器通过创建块来延迟表达式的求值，直到这个表达式的值真正被需要为止。如果某个表达式的值不被需要，那么从始至终，这个表达式都不会被求值。

可以通过代换（substitution）和重写（rewriting）去了解 Haskell 求值表达式的方式。

惰性求值可以延迟计算直到真正需要一个值为止，并且在求值时，也只执行可以确立出（establish）值的那部分表达式。[译注：比如之前提到的， (||) 的左操作对象的值为 True 时，就无需对右操作对象估值的情况。]

函数的返回值可能是一个块（一个被延迟计算的表达式）。