Haskell Unit 3: Floating-point Numbers and Characters

Antoni Diller

26 July 2011

Introduction

Haskell has two types for floating-point numbers:

`Float`	single-precision
`Double`	double-precision

Floating-point numbers can be represented in two ways. First, using a decimal point:

```
2.0
33.873
-8.3377
```

Second, by means of the so-called scientific notation:

`33.61e6`	(equivalent to $33.61 * 10^6$)
`3.7e-2`	(equivalent to $3.7 * 10^-2$)
`-3.7e2`	(equivalent to $-3.7 * 10^2$)

Haskell has the usual binary infix floating-point operators, namely

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`**`	exponentiation

It also has the unary prefix operator `-` (minus or negative) and the following unary prefix operators:

`cos`	cosine
`sin`	sine
`tan`	tangent
`log`	logarithms to base e
`acos`	inverse cosine
`asin`	inverse sine
`atan`	inverse tangent
`exp`	powers of e
`sqrt`	square root

Haskell has some useful functions for converting floating-point numbers into single-precision integers:

`ceiling 2.3`	is equivalent to	`3`
`floor 2.3`	is equivalent to	`2`
`round 2.3`	is equivalent to	`2`
`round 2.7`	is equivalent to	`3`

These are all of type `Float -> Int`. The function `fromInt` of type `Int -> Float` converts a limited-precision integer into a single-precision floating-point number.

Numerical type classes

So far four numerical types in Haskell have been introduced, namely `Int`, `Integer`, `Float` and `Double`. It is tedious to define a new function that squares its argument, say, for each numerical type:

```
sqInt :: Int -> Int
sqInt x = x * x

sqInteger :: Integer -> Integer
sqInteger x = x * x

sqFloat :: Float -> Float
sqFloat x = x * x

sqDouble :: Double -> Double
sqDouble x = x * x
```

Haskell has several type classes which allow one definition to do the work of more than one of the above monomorphic definitions:

```
sqIntegral :: Integral a => a -> a
sqIntegral x = x * x

sqFractional :: Fractional a => a -> a
sqFractional x = x * x

sqReal :: Real a => a -> a
sqReal x = x * x
```

The type class `Integral` contains the two types `Int` and `Integer`. The type class `Fractional` contains the two types `Float` and `Double`. The type class `Real` contains the four types `Int`, `Integer`, `Float` and `Double`. These, and some other important types, can be represented by the following inclusion diagram:

```
          Eq
         /  \
        /    \
       /      \
      /        \
     /          \
    /            \
  Ord            Num
    \            /
     \          /
      \        /
       \      /
        \    /
         \  /
         Real
         /  \
        /    \
       /      \
      /        \
     /          \
    /            \
 Integral     Fractional
```

Characters

The type `Char` contains characters. Elements of `Char` are written enclosed in single closing quotation marks, for example:

`'a'`	
`'B'`	
`'4'`	
`'\t'`	tab
`'\n'`	newline
`'\\'`	backslash
`'\''`	single closing quotation mark
`'\"'`	double quotation mark

There are several useful functions dealing with characters:

`toUpper`	`Char -> Char`	
`toLower`	`Char -> Char`	
`ord`	`Char -> Int`	into ASCII code
`chr`	`Int -> Char`	from ASCII code
`isAscii`	`Char -> Bool`	
`isUpper`	`Char -> Bool`	
`isLower`	`Char -> Bool`	
`isAlpha`	`Char -> Bool`	
`isDigit`	`Char -> Bool`	
`isAlphaNum`	`Char -> Bool`	