Шпаргалка по haskell

Замечания

– Списки можно сравнивать, списки **большей** длины **больше**:

```
[1,2,3] > [100,200]
```

– Оператор .. в генераторах:

```
[a..b] = [a,a+1,...b], b>=a
[a..b] = [], b<a
```

– *let* в генетаторах:

```
[x*y | x<-[1,2,3], let y=2] == [2,4,6]
```

– Числа с плавающей точкой лучше **не использовать в интервалах!** Могут вызвать непредсказуемое поведение.

– Порядок применения функций в генераторах:

```
[x `f` y | x<-[a1,b1,...], y<-[a2,b2,...]] ==
== [a1 `f` a2, a1 `f` b2, ..., b1 `f` a2]
```

– Можно объявлять инфиксную функцию:

```
b `a` c = b + c
5 `a` 10 == 15
```

– Вычитание в сечениях:

```
map (-4) [4,4,4] == BOTTOM
map (subtract 4) [4,4,4] == [0,0,0]
```

– Каррирование в лямбда-выражениях:

```
\x y -> x+y == \x -> \y -> x+y
```

– Операторы ($) и (.) **правоассоциативны**

– Именованные образцы:

```
(f name@(x:xs) = ...) == (f (x:xs) = ... where name=(x:xs))
```

Импорт модулей

– Все функции в глобальное пространство имен:

```
import A.B
```

– Только функции a, b:

```
import A.B (a, b)
```

– Кроме a, b:

```
import A.B hiding (a, b)
```

– Вызов через $A.B.func$:

```
import qualified A.B
```

– Вызов через $C.func$:

```
import qualified A.B as C
```

Классы типов

```
name :: (Class x) => T1 [->T2->...->TN], Ti=x
```

Eq – равны (=) либо не равны (/=)
Ord – отношения порядка (>, <, >=, <=)
Show – применима функция show:

```
show :: (Class x) => x -> String
```

Read – применима функция read :: x:

```
read :: (Class x) => String -> x
```

Enum – значения можно пронумеровать
Bounded – есть верхняя/нижняя граница
Num – экземпляры ведут себя как числа
Floating – как числа с плавающей точкой
Integral – как целые числа