Haskell ve Fonksiyonel Programlama

Tuan Susam

Icindekiler

Giris

- Fonksiyonel Programlama?

Temel Kavramlar

- Tipler & Tip Siniflari
- Listeler
- Fonksiyonlar
- Ifadeler

Uygulamali

- GHCi, Prelude ve Temel Fonksiyonlar
- Lambda Fonksiyonlari
- Katlanabilirlere (foldables) Giris

Fonksiyonel Programlama

- Fonksiyonlar ve bu fonksiyonlarin uygulamalari uzerine kurulu bir programlama yontemidir.
- Kullanilan matematiksel notasyon problemlerin acikca ve farkindalikla belirtilmesini saglar.
- Programlarin ozellikleri hakkinda denklemsel sebep verebilmeye olanak saglayan matematiksel bir temeli vardir.

Neden Fonksiyonel?

Declarative Programming

Fonksiyonel programlama, hesaplama kavramini matematiksel fonksiyonlarin hesaplanmasi olarak gorur. Diger dillerde sikca karsilastigimiz *changing-state* ve *mutable data*'dan kacinir. **Declarative programming**, ifadeler ve beyanlarla (**expressions** & **declarations**) yapilir, *statement* kavrami burada gecerli degildir. Hesaplamanin *control flow*'unu anlatmadan mantigini anlatarak yapilan bir programlamadir

Ornekleri

Common Lisp (1980s), Scheme (1970s), Clojure (2007), Wolfram Language (1988/2019), Racket (1995/2019), Erlang (1986/2019), OCaml (1996/2018), Haskell (1990/2010), F# (2005/2019)

Haskell

Adini Amerikali mantikci Haskell Brooks Curry'den aliyor

- Tembel Fonksiyonel Dil

Sektorde Haskell

- Facebook anti-spam programlari
- Swift Navigation
- Github
- Cryptool

Web Frameworks

- Yesod
- Snap

Tipler

Sayilar

- Int
- Integer
- Rational
- Float
- Double
- Complex (Data.Complex)

Bool

Boolean tipidir, *True* veya *False* degerlerinden birini alir

Char

Bir karakteri temsil eder *'a'* gibi, karakterler listesi ise *string*'dir

Tip Siniflari

Eq

Bir fonksiyonun tip degiskeni icin **Eq** sinifi belirteci varsa fonksiyonumuz taniminda $==$ veya $/=$ icermektedir.

5 'elem' [1..10]

Ord

Ord butun standart kiyaslama fonksiyonlarini icerir. Ornegin $>$, $<$, $>=$, $<=$. *Kiyaslama* fonksiyonu iki **Ord** uyesi (ayni tipten) alir ve bir *ordering* geri getirir.

data Ordering = LT | EQ | GT

Show

Show'un uyesi olan bir degeri alip bize "Gosterir".

Read

Show sinifinin tam tersidir

Tip Siniflari - 2

Enum

Sirali kiyasli tipler bu sinifa uyedir. Bu sinifa dahil tipler *()*, *Bool*, *Char*, *Ordering*, *Int*, *Integer*, *Float*, *Double*

['A'..'Z']

[3 .. 5]

succ 'b'

Bounded

uyulerinin ust ve alt sinirlari vardir.

Tip Siniflari - 3

Num, Integral, Floating

Numeric tipsinifidir. Uyeleri sayi gibi davranabilirler **Num** sinifi butun sayilari icine alirken, **Integral** sinifinda sadece **Int** ve **Integer**lar vardir. Benzer bir sekilde **Float** ve **Double** tipleri **Floating** tip sinifindadir.

fromIntegral

Tum bu tipler ve tip siniflari karmasasi icerisinde, problemlerimizin cozumu olan bu fonksiyon farkli tiplerle islem yapabilmemizi sagliyor.

fromIntegral (length "length bize Int getirecek") + pi

Biraz Sayilar

negate x = -x

abs x = if x < 0 then -x else x

```
signum x | x < 0 = -1
         | x == 0 = 0

         | x > 0  = 1
```

divMod

fonksiyonu bize sonuc olarak (bolum, kalan) seklinde 2'li-tup verir
x `div` y = fst (x `divMod` y)
x `mod` y = snd (x `divMod` y)

Biraz Daha Sayilar ve Fonksiyonlar :)

floor?

```
floor' :: Float -> Integer
floor' = read . takeWhile (/= '.') . show
```

Until

```
until :: (a -> Bool) -> (a -> a) a -> a
until p f x = if p x then x else until p f (f x)
until (>100) (*7) 1 == 343
```

Subtract

```
subtract x y = y-x
```

O Zaman List

Verileri cekmek veya tasimak icin kullanilan yapilardan biridir

[undefined,undefined] :: [a]
[sin,cos,tan] :: Floating a => [a -> a]
[[1,2,3],[4,5]] :: Num a => [[a]]
["Seminer Odasi",'M',231]

Liste Olusturmak

1:2:3:[] == [1,2,3]

Liste Elemani Cagirmak

[0,1,2,3,4,5]!!3 == 3

Listelerin Ozellikleri

[a] tipindeki bir liste asagidaki 3 formdan birindedir

- tanimsiz liste :: [a]
- bos liste [] :: [a]
- (x:xs) formunda olan bir listede x :: a'dir ve xs ::[a]

Dolayisiyla 3 Cesit liste vardir

- Sonlu listeler
- Parcali listeler
- Sonsuz listeler

Listelerle Calisan Fonksiyonlar

- ++
- head
- tail
- last
- reverse
- length
- sum
- take n
- iterate
- zip
- zipWith

Liste Tanimlamak

Kume Notasyonu

```
[x*x | x <- [1..10], isPrime x]
```

```
[(i,j) | i <- [1..5], j <- [1..5]]
```

Enum

```
[1..10]
```

String

```
"abcdef" == ['a','b','c','d','e','f']
```

Kelime Frekansi Problemlemi

Verilen bir metinde en cok tekrar eden n kelimeyi bize verecek fonksiyonu yazmak

Kaynaklar

Kitaplar

- Thinking Functional with Haskell
- Beginning Haskell

Web

www.learnyouahaskell.com/