Haskell 1

Fonksiyonlar, Listeler, Liste Üreteçleri ve Veri Türleri

Şevket Umut ÇAKIR

PAÜ

24 Mart 2020

Giriş

- Haskell **saf bir fonksiyonel programlama dilidir**.
- Saf fonksiyonel programlama dillerinde programcı bilgisayarını ne yapacağını söylemek yerine işlerin ne olduğunu tanımlar.
- Örneğin bir sayının faktoryeli, 1 ile sayı arasındaki sayıların çarpımıdır.
- Örneğin bir sayı listesinin toplamı, ilk sayının diğer sayınların toplamına eklenmesi ile bulunabilir.
- Saf fonksiyonel programlama dillerinde **atama** ve **döngü** işlemleri bulunmaz.
- Haskell tembel bir dildir. Yani bir sonuç göstermeye mecbur olmadığı sürece fonksiyonları değerlendirmez ve hesaplamaları yapmaz.

Kurulum ve IDE Seçimi

- Kurulum için Haskell Platform'u indirip yüklemek gereklidir.
- Platform içinde Glasgow Haskell Derleyicisi, Cabal İnşa Sistemi, proje geliştirmek için Stack aracı, kod profilleme araçları ve sık kullanılan 35 paket gelmektedir.
- Basit uygulamalar yapmak için platformu kurmak yerine sadece derleyici ve yorumlayıcı kullanılabilir.
- Entegre Geliştirme Ortamları
  - Leksah: Haskell'e özgü IDE
  - Haskell için IntelliJ eklentisi
  - Eclipse IDE için EclipseFP Eklentisi
  - Vim, Emacs, Atom, Visual Studio Code

- **ghc**: Haskell derleyicisi. Çalıştırılabilir dosyalar üretir.

```
#kaynak.hs dosyasının derlenmesi
ghc -o kaynak kaynak.hs
```

- **ghci**: Haskell yourumlayıcısı. Kaynak kodu derlemeden yorumlar ve çıktı üretir.

```
$ ghci
Prelude> :l kaynak.hs
[1 of 1] Compiling Lib                ( kaynak.hs, interpreted )
Ok, one module loaded.
*kaynak>
```

İşleçler

Tablo: Çok Kullanılan İşleçler

İşleç	**İşlevi**	**İşleç**	**İşlevi**
+	Toplama	<	Küçüktür
-	Çıkarma, eksi	<=	Küçük veya eşit
*	Çarpma	==	Eşittir
/	Bölme	/=	Eşit değildir
^, ^^, **	Üs alma	>=	Büyük veya eşit
&&	Ve	>	Büyüktür
||	Veya	++	Liste birleştirme
\	Lambda	:	Başa ekleme (cons)
.	Fonksiyon birleştirme	!!	indeks
|	Guard ve case tanımlayıcı Liste üreteçlerindeki ayraç Veri tanımında alternatif	..	Listeler için aralık belirteci

İşleç Kullanımı

```
GHCi, version 8.6.3: http://www.haskell.org/ghc/  :? for
↪  help
Prelude> 3+4
7
Prelude> 3+4*5
23
Prelude> 3<4 || 5>9
True
Prelude> div 7 2
3
Prelude> 7/2
3.5
```

Fonksiyonlar

Fonksiyon Tanımlama

Fonksiyon tanımlamak çin ilk olarka fonksiyonun adı, sonra = sembolü ve fonksiyonun geri dönüş değeri yazılır.

```
--fonk1.hs dosyası içi
ikikati x = x + x
ikikatitoplam x y = x*2 + y*2
kucukikikati x = if x > 100 then x else 2*x
```

Fonksiyonlar

Fonksiyonları çağırmak için fonksiyon adı ve aralarında boşluk karakteri olacak şekilde parametreleri sırayla yazılır.

```
*Main> ikikati 7
14
*Main> ikikati 3.14
6.28
*Main> ikikatitoplam 3 5
16
*Main> kucukikikati 50
100
*Main> kucukikikati 120
120
*Main> ikikati 3 + ikikatitoplam 1 2
12
```

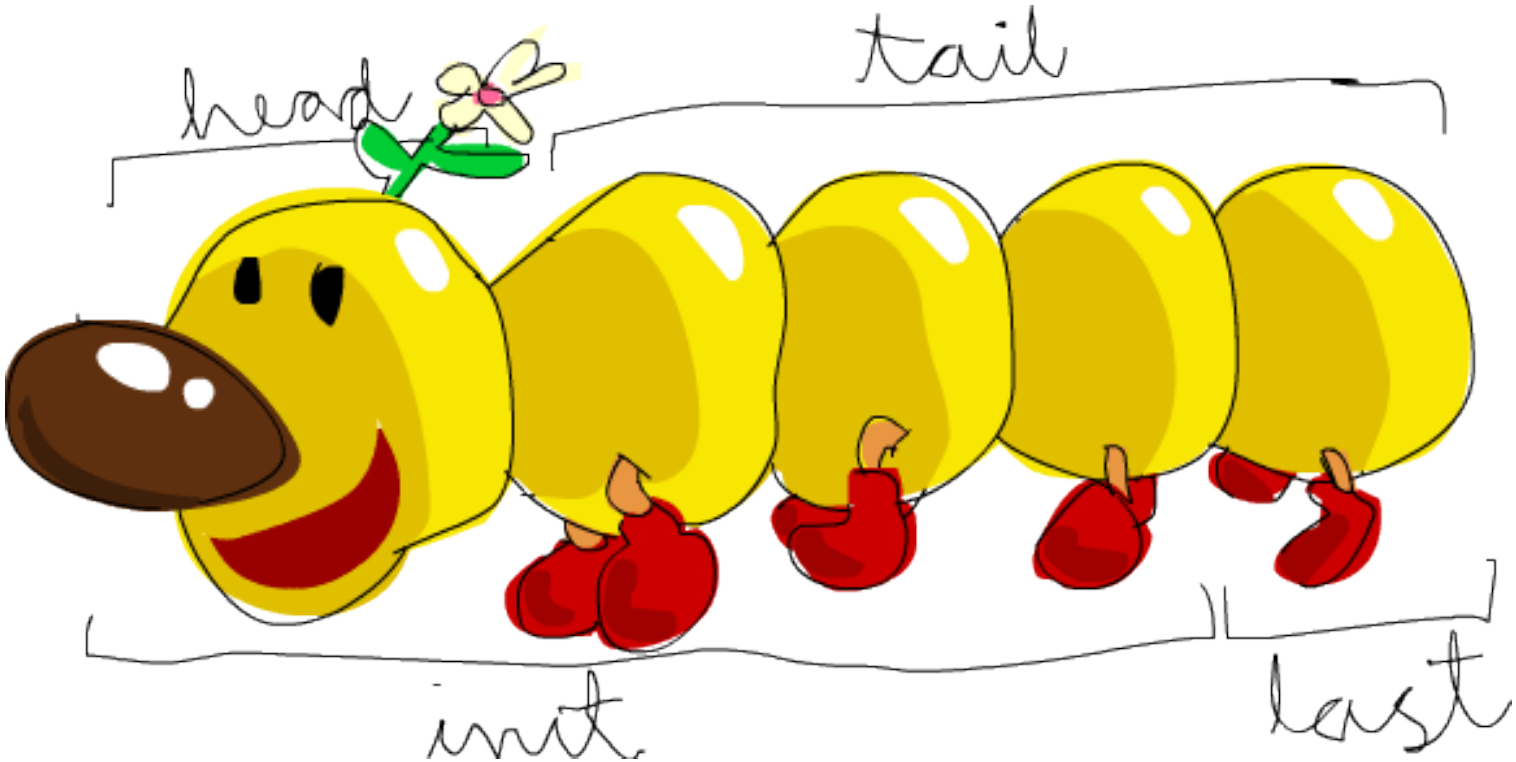
Listeler I

- Listeler içinde aynı türden elemanların bulunduğu bir veri yapısıdır.
- Listeleri oluşturmak için **[ ]** sembolleri arasında virgüllerle ayrılarak değerler yazılır.
- Listeler aşağıdaki işleçler ve fonksiyonlara sahiptir
  - **:, ++, !!** Sırasıyla listenin başına eleman ekleme, listeleri birleştirme ve belirtilen konumdaki elemanı verme
  - **head**: Listenin ilk elemanını
  - **tail**: İlk eleman hariç hepsi
  - **last**: Son eleman
  - **init**: Son eleman hariç hepsi
  - **length**: Listenin uzunluğu
  - **reverse**: Listeyi tersine çevirir

Listeler II

- **null**: Listenin boş olup olmadığını verir
- **take**: Listeden belirtilen sayıda eleman alır
- **drop**: Listeden belirtiden sayıda elemanı çıkarır
- **maximum**: Listenin en büyüğü
- **minimum**: Listenin en küçüğü
- **sum**: Listedeki elemanları toplamı
- **product**: Listedeki elemanların çarpımı
- **elem**: Bir eleman listede var mı

Listeler III



Şekil: head, tail, last, init fonksiyonlarının görseli[2]

Liste Kullanımı

```
Prelude> let sayilar=[4,8,15,16,23,42]
Prelude> 1:sayilar
[1,4,8,15,16,23,42]
Prelude> sayilar ++ [1,2,3]
[4,8,15,16,23,42,1,2,3]
Prelude> sayilar !! 3
16
Prelude> head sayilar
4
Prelude> length sayilar
6
Prelude> maximum sayilar
42
Prelude> product sayilar
7418880
Prelude> (sum sayilar) `div` (length sayilar)
18
Prelude> take 3 sayilar
[4,8,15]
Prelude> sayilar
[4,8,15,16,23,42]
Prelude> init "Merhaba"
"Merhab"
```

Liste Kullanımı

Listeleri oluştururken .. sembolü ile aralık tanımlanabilir.

```
Prelude> [1..20]
[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
Prelude> [1,3..20]
[1,3,5,7,9,11,13,15,17,19]
Prelude> take 10 [13,26..]
[13,26,39,52,65,78,91,104,117,130]
Prelude> take 10 (cycle [1,2,3])
[1,2,3,1,2,3,1,2,3,1]
Prelude> take 10 (repeat 5)
[5,5,5,5,5,5,5,5,5,5]
```

Liste Üreteçleri(List Comprehensions)

Liste Üreteçleri

Matematikte kullanılan $S = \{2 \cdot x | x \in \mathbb{N}, x \leq 10\}$ benzerindeki ifadeleri kullanarak listeler üretmeyi sağlar.

```
Prelude> [x*2 | x <- [1..10]]
[2,4,6,8,10,12,14,16,18,20]
Prelude> [x*2 | x <- [1..10], x*2 >= 12]
[12,14,16,18,20]
Prelude> [ x | x <- [50..100], x `mod` 7 == 3]
[52,59,66,73,80,87,94]
Prelude> [x*x | x<-[20..40], odd x, x `mod` 3==0]
[441,729,1089,1521]
Prelude> [ x*y | x <- [2,5,10], y <- [8,10,11], x*y > 50]
[55,80,100,110]
```

Hızlı Sıralama

```
quicksort [] = []
quicksort (x:xs) = quicksort [y | y <- xs, y <= x] ++ [x]
↪  ++ quicksort [y | y <- xs, y > x]
```

Çokuzlular(Tuples)

Çokuzlular

Çokuzlular içinde birden fazla farklı türden elemanın saklanabildiği bir veri yapısıdır.

- Elemanlar **( )** sembolleri arasına virgüller ile ayrılarak yazılır.
- En çok kullanılanı ikili olanıdır
- **fst** ikili çokuzlunun ilk elemanını, **snd** ikinci elemanını verir.
- Listelerde olduğu gibi değişken yapıya sahip değillerdir(immutable).
- **zip** fonksiyonu listeleri çokuzlular kullanarak birleştirmeyi sağlar.

Çokuzlu Kullanımı

```
Prelude> ("Haskell", 12, 3.14)
("Haskell",12,3.14)
Prelude> fst ("Merhaba",12345)
"Merhaba"
Prelude> snd ("Merhaba",12345)
12345
Prelude> zip [1 .. 5] ["one", "two", "three", "four",
↪  "five"]
[(1,"one"),(2,"two"),(3,"three"),(4,"four"),(5,"five")]
Prelude> zip [1..] ["apple", "orange", "cherry", "mango"]
[(1,"apple"),(2,"orange"),(3,"cherry"),(4,"mango")]
Prelude> [ (a,b,c) | c <- [1..10], b <- [1..c], a <-
↪  [1..b], a^2 + b^2 == c^2]
[(3,4,5),(6,8,10)] --tamsayı kenar uzunluklu dik üçgenler
```

fst ve **snd** Tanımı

fst ve **snd** fonksiyonlarının tanımları aşağıda görüldüğü gibi basit bir yapıya sahiptir.

```
fst (x, _) = x
snd (_, y) = y
```

Türler Üzerine

- Haskell dilinde tanımlı birçok veri türü bulunmaktadır.
- Herhangi bir verinin türünü öğrenmek için **ghci** ekranında önüne **:t** yazılabilir.
- Bazı türler polimorfik olmaktadır. Bunlar **a, b** gibi harfler ile gösterilir.

```
Prelude> :t (+)
(+) :: Num a => a -> a -> a
Prelude> :t head
head :: [a] -> a
Prelude> :t tail
tail :: [a] -> [a]
Prelude> :t fst
fst :: (a, b) -> a
```

enSonEleman

Fonksiyona parametre olarak verilen listenin en son elemanını veren fonksiyondur. Hazır fonksiyonları kullanabilirsiniz.

sondanIkinci

Fonksiyona parametre olarak verilen listenin sondan bir önceki elemanını veren fonksiyondur.

```
Prelude> sondanIkinci [1,2,3,4,5]
4
```

ortanca

Parametre olarak verilen listenin ortadaki elemanını verir. Liste sıralı olmak zorunda değildir ve sıralamaya ihtiyaç yoktur.

```
Prelude> ortanca [4,7,1,3,9,21,12]
3
Prelude> ortanca [4,7,1,3,9,21,12,4]
3
```

bilmoodle üzerindeki kodlar

```haskell
module Ogrenci where --Bu satırı değiştirmeyin/silmeyin

--enSonEleman fonksiyonunu yazın
enSonEleman::[a]->a
enSonEleman _ = error "enSonEleman henuz tanimlanmadi"

--sondanIkinci fonksiyonunu yazın(listenin sondan ikinci elemanını verir)
sondanIkinci::[a]->a
sondanIkinci _ = error "sondanIkinci henuz tanimlanmadi"

--ortanca fonksiyonunu yazın
ortanca::[a]->a
ortanca _ = error "ortanca henuz tanimlanmadi"

```

ilk, ikinci ve ucuncu

İki elemanlı çokuzlular için ilke elemanı veren `fst` ve ikinci elemanı veren `snd` fonksiyonları bulunmaktadır. Üç elemanlı çokuzlular için ilk, ikinci ve üçüncü elemanları veren fonksiyonları yazınız.

```
Prelude> ilk (1,2,3)
1
Prelude> ucuncu (123, "merhaba", 3.14)
3.14
```

bilmoodle üzerindeki kodlar

```haskell
module Ogrenci where --Bu satırı değiştirmeyin/silmeyin

--ilk fonksiyonunu yazın
ilk::(a,b,c)->a
ilk _ = error "ilk henuz tanimlanmadi"

--ikinci fonksiyonunu yazın
ikinci::(a,b,c)->b
ikinci _ = error "ilk henuz tanimlanmadi"

--ucuncu fonksiyonunu yazın
ucuncu::(a,b,c)->c
ucuncu _ = error "ilk henuz tanimlanmadi"

```

Kaynaklar I


Haskell web programming.
http://yannesposito.com/Scratch/fr/blog/Yesod-tutorial-for-newbies/.


Learn you a haskell for great good! a beginner's guide.
http://learnyouahaskell.com.


Learning haskell.
https://wiki.haskell.org/Learning_Haskell.


Try haskell.
http://tryhaskell.org.


Yet another haskell tutorial.
http://users.umiacs.umd.edu/~hal/docs/daume02yaht.pdf.


Zor yoldan haskell.
https://github.com/joom/zor-yoldan-haskell.