Programlama Dilleri Laboratuvar Föyü
Haskell Programlama Dili 1
Fonksiyonlar, Listeler, Liste Üreteçleri ve Veri Türleri

Şevket Umut ÇAKIR

1 Giriş

Haskell **saf bir fonksiyonel programlama dilidir**. Emirli(imperative) programlama dillerinde programcı bilgisayara ardışık bir dizi görev verir ve bilgisayar sırayla bu görevleri yerine getirir. Örneğin programcı **a** değişkenine 5 değerinin atanması ve bu değişkenle ilgili bir takım işlemlerin gerçekleştirilmesi görevlerini verebilir. Emirli dillerde bazı işlemleri tekrarlı olarak yapmak için bulunan kontrol akış yapıları bulunmaktadır. Saf fonksiyonel programlama dillerinde programcı bilgisayarını ne yapacağını söylemek yerine işlerin ne olduğunu tanımlar. Örneğin bir sayının faktoryeli, 1 ile sayı arasındaki sayıların çarpımıdır; bir sayı listesindeki sayıların toplamı, ilk sayı ile diğer sayıların toplamının toplamıdır. Bu yapılar fonksiyonlar biçiminde ifade edilir. Saf fonksiyonel programlama dillerinde bir değişkene bir değer atayıp, sonrasında bu değeri değiştiremezsiniz. Eğer **a** 5'dir dediyseniz **a** 5'dir, değişmez.

Haskell **tembel** bir dildir. Bunun anlamı, tersi belirtilmediği sürece, bir sonuç göstermeye mecbur olmadığı sürece fonksiyonları değerlendirmez ve hesaplamaları yapmaz. Bu referans saydamlığı ile uyumludur ve programlar veriler üzerinde dönüşümler olarak düşünülebilir. Haskell statik tip sistemine sahip bir dildir ve tipi belli olmayan değerler için tip çıkarımı(type inference) kullanmaktadır.

1.1 Kurulum ve IDE Seçimi

Bilgisayara Haskell kurmak için Haskell Platform'u indirip yüklemek gerekmektedir. 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.

Haskell için özelleştirilmiş bir entegre geliştirme ortamı(IDE) Leksah bulunmaktadır. Bunun yanı sıra kod renklendirme ve tamamlama gibi özelliklerin

bulunduğu kod düzenleyicileri de bulunmaktadır. Herhangi bir metin düzenleyicisi de Haskell programlarını yazmak için yeterli olacaktır.

- Leksah: Haskell'e özgü IDE
- Haskell için IntelliJ eklentisi
- Eclipse IDE için EclipseFP Eklentisi
- Vim, Emacs, Atom, Visual Studio Code

Haskell programları derlenerek veya yorumlanarak çalıştırılabilir. Bunlar için sırasıyla **ghc** ve **ghci** araçları kullanılabilir. Deneylerde uygulamalar daha çok etkileşimli yorumlayıcı üzerinden gerçekleştirilecektir. Bunun için komut satırına **ghci** yazarak yorumlayıcı ekranına girilebilir. Fonksiyonlarımızı *fonksiyonlarim.hs* dosyasına kaydetmiş isek yorumlayıcı içinde `:l fonksiyonlarim` yazarak yüklenebilir. Dosyayı tekrar yüklemek için `:r` komutunu vermek yeterlidir.

2 Konu Anlatımı ve Deney Hazırlığı

2.1 İşleçler(Operatörler)

Haskell dilinde diğer dillerde bulunan bir çok işleç bulunmaktadır. Aritmetik işleçler ortada(infix gösterim) yer alırlar. Infix ifadeleri prefix biçimine dönüştürmek için işleç parantez içine alınabilir. Tersi olarak prefix biçimindeki bir işleci infix biçiminde kullanmak için ‘ sembolü kullanılabilir. Tablo 1'de çok kullanılan işleçler gözükmektedir.

Tablo 1: 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

Aşağıda bazı işleçlerin örnek kullanımları verilmiştir.

```
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
```

2.2 Fonksiyonlar

Fonksiyonlar fonksiyonel programlamanın temelini oluşturur. Fonksiyonları tanımlamak genellikle bir dosya içine kaydetmekle gerçekleşir. Fonksiyon adı ve parametreler boşlukla ayrılarak yazıldıktan sonra = sembolü ve fonksiyonun gövdesi verilir. Aşağıda örnek fonksiyon tanımları bulunmaktadır.

```
--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
```

Aşağıda tanımlanan bu fonksiyonların kullanımı gözükmektedir.

```
*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
```

2.3 Listeler

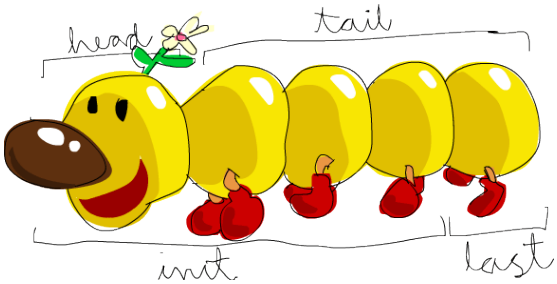
Listeler içinde aynı türden elemanların bulunduğu bir veri yapısıdır. Listeler köşeli parantezler içinde virgüllerle ayrılarak yazılan elemanlardan oluşur. Aşağıda listelerde çalışan fonksiyonlar ve işlevleri verilmiştir. Haskell saf bir fonksiyonel programlama dili olduğu için bu işlemler geriye yeni bir liste döndürür ve orjinal liste değişmez.

- **:**, **++**, **!!** 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
- **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

```
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"
```

Listeleri oluştururken aralık kullanılabilir. Bunun için **..** işleci kullanılır.

```
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]
```



Şekil 1: head, tail, last, init fonksiyonlarının görseli[2]

```
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]
```

2.4 Liste Üreteçleri(List Comprehensions)

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

```
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]
```

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

2.5 Çokuzlular(Tuples)

Çokuzlular içinde farklı türlerden birden fazla elemanın barındırılabileceği bir veri türüdür. Parantezler içinde virgüllerle birbirinden ayrılarak elemanlar çokuzlu içine yazılır. `fst` ve `snd` iki elemanlı çokuzluların ilk ve ikinci elemanlarını almak için kullanılır. `zip` fonksiyonu ise listeleri çokuzlular kullanarak birleştirmeyi sağlar. Aşağıda örnek kullanımları vardır.

```
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 uzuluklu dik üçgenler
```

`fst` ve `snd` fonksiyonlarının aşağıda olduğu gibi gayet basit tanımlamaları bulunmaktadır.

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

2.6 Türler Üzerine

Haskell dilinde tanımlı olan bir çok veri türü bulunmaktadır. Herhangi bir öğenin türünü öğrenmek için **ghci** ekranında ifadenin önüne **:t** sembolü yerleştirmek yeterlidir. Bazı türler sayısal olabileceği gibi bazıları da çok biçimli(polymorphic) olabilmektedir. fst fonksiyonun giriş ve çıkış değerleri buna örnektir.

```
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
```

3 Deneyin Uygulanması

Bu deneyde yazılması istenen kodların bilgisi aşağıda verilmiştir.

3.1 enSonEleman Fonksiyonu

Fonksiyona parametre olarak verilen listenin en son elemanını veren fonksiyondur.

3.2 sondanIkinci Fonksiyonu

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

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

3.3 ortanca Fonksiyonu

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
```

3.4 ilk, ikinci ve ucuncu Fonksiyonları

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
```

Kaynaklar

[1] *Haskell Web Programming*. URL: `http://yannesposito.com/Scratch/fr/blog/Yesod-tutorial-for-newbies/` (son erişim: 2.4.2019).

[2] *Learn You a Haskell for Great Good! A Beginner's Guide*. URL: `http://learnyouahaskell.com` (son erişim: 2.4.2019).

[3] *Learning Haskell*. URL: `https://wiki.haskell.org/Learning_Haskell` (son erişim: 2.4.2019).

[4] *Try Haskell*. URL: `http://tryhaskell.org` (son erişim: 2.4.2019).

[5] *Yet Another Haskell Tutorial*. URL: `http://users.umiacs.umd.edu/~hal/docs/daume02yaht.pdf` (son erişim: 2.4.2019).

[6] *Zor Yoldan Haskell*. URL: `https://github.com/joom/zor-yoldan-haskell` (son erişim: 2.4.2019).