CENG202 PROGRAMLAMA DİLLERİ ÖDEVİ

24 ŞUBAT 2020

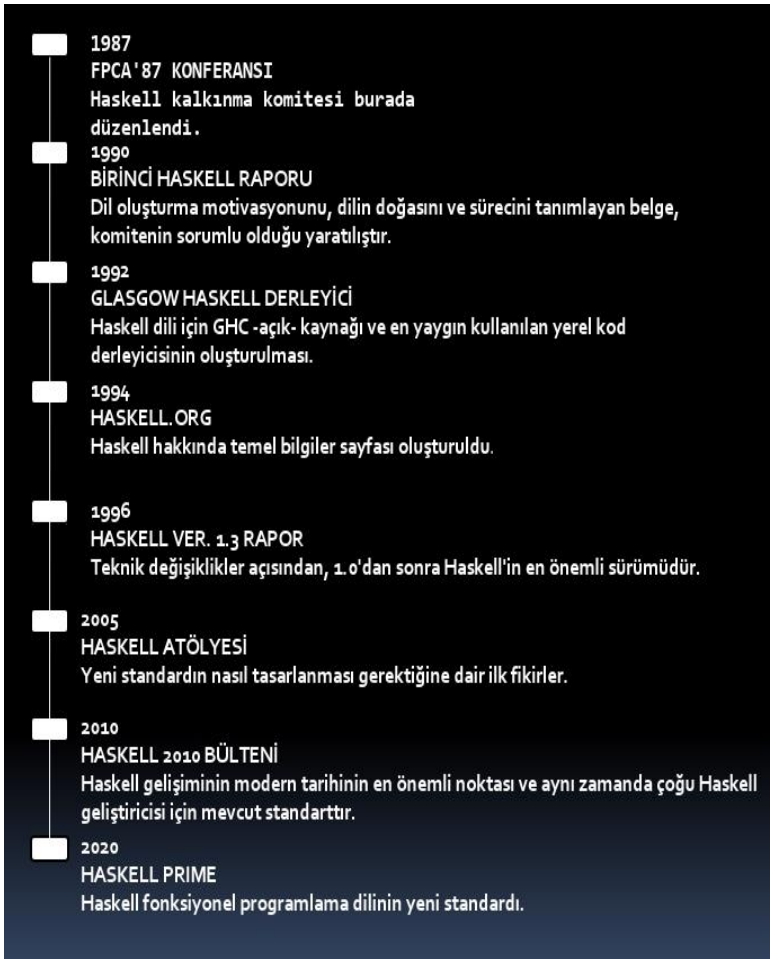
HASKELL PROGRAMLAMA DİLİ

HASKELL DİLİNİN TARİHÇESİ

Haskell, isim babası matematikçi Haskell Curry olan arı işlevsel programlama dilidir. Haskell'i birçok programlama dilinden ayıran özellikleri tembel değerlendirme, monadları ve tür sınıflarıdır. Haskell, Miranda dilinin semantikleri üzerine kuruludur.

Miranda'nın Araştırma Yazılımı Ltd. tarafından 1985'te yayınlanmasının ardından tembel fonksiyonel dillere ilgi arttı. 1987'de, bir düzineden fazla katı olmayan, tamamen işlevsel programlama dilleri vardı. Miranda en yaygın kullanılanıydı, ancak tescilli bir yazılımdı. Oregon, Portland'daki Fonksiyonel Programlama Dilleri ve Bilgisayar Mimarisi (FPCA '87) konferansında, bu diller için açık bir standart tanımlamak üzere bir komite oluşturulması konusunda fikir birliğine varıldı. Komitenin amacı, fonksiyonel dil tasarımında gelecekteki araştırmalar için bir temel oluşturacak şekilde mevcut işlevsel dilleri ortak bir dilde birleştirmekti.

Haskell 1.0-1.4 : Güvenli tip operatör aşırı yüklemesini mümkün kılan tip sınıfları ilk önce Philip Wadler ve Stephen Blott tarafından Standart ML için önerilmiş, ancak ilk olarak 1987 ve sürüm 1.0 arasında Haskell'de uygulanmıştır.
Haskell'in ilk versiyonu ("Haskell 1.0") 1990 yılında tanımlanmıştır. Komitenin çabaları bir dizi dil tanımı ile sonuçlanmıştır (1.0, 1.1, 1.2, 1.3, 1.4).



Haskell 98 : 1997'nin sonlarında, Haskell 98'de zirveye ulaşan seri, dilin kararlı, minimal, taşınabilir bir versiyonunu ve beraberindeki öğretim için standart bir kütüphaneyi ve gelecekteki genişletmeler için bir temel olarak belirtmeyi amaçladı. Şubat 1999'da Haskell 98 dil standardı ilk olarak Haskell 98 Raporu olarak yayınlandı.Ocak 2003'te, gözden geçirilmiş bir sürüm Haskell 98 Dil ve Kütüphaneler: Düzeltilmiş Rapor olarak yayınlandı. Mevcut de facto standardını temsil eden Glasgow Haskell Derleyici (GHC) uygulaması ile dil hızla gelişmeye devam ediyor.

Haskell 2010 : 2006'nın başlarında, Haskell Prime'ın gayri resmi olarak Haskell Prime adlı halinin tanımlanması süreci başladı. Bunun, dil tanımını onarıp yenilemek ve yılda bir defaya

kadar yeni bir güncelleme üretmek için devam eden artımlı bir süreç olması amaçlanmıştır. Haskell 2010 olarak adlandırılan ilk güncelleme Kasım 2009'da açıklandı ve Temmuz 2010'da yayınlandı. Haskell 2010, daha önce derleyiciye özgü bayraklar aracılığıyla etkinleştirilmiş birkaç iyi kullanılmış ve tartışmasız özelliği içeren, dilde artımlı bir güncellemedir.

DİLİN TASARIM AMAÇLARI(DESİGN GOALS)

Dilin tasarım amacı okuyucuya mümkün olan en kısa sürede faydalı kod yazma konusunda yetkin hizmet sağlamasıdır. Haskell dili hızı yavaş ve ayrıntılı yapılmaya çalışılmıştır.

Haskell dilini geliştiren komitenin temel amacı aşağıdaki özellikleri karşılayan bir dil tasarlamaktı:

- Büyük sistemler inşa etmek de dahil olmak üzere öğretim, araştırma ve uygulamalar için uygun olmalıdır.
- Tamamen resmi bir sözdizimi ve anlambilimin yayınlanmasıyla açıklanmalıdır.
- Serbestçe kullanılabilir olmalıdır. Herkesin dili uygulamasına ve dilediği kişiye dağıtmasına izin verilmelidir.
- Geniş bir fikir birliğine sahip olan fikirlere dayanmalıdır.
- Fonksiyonel programlama dillerinde gereksiz çeşitliliği azaltmalıdır.

PROGRAMLAMA DİLİNİN HEDEF KİTLESİ

Matematik veya fonksiyonel programlama konusunda yüksek bilgiye sahip olmayan programcılar, akademisyenler, endüstri alanında çalışanların kullanması hedeflenmiştir.

PROGRMLAMA DİLİNİN KULLANIM ALANLARI

1. Yapay zekada kullanılır.(Fonksiyonel dil olmasından kaynaklı)
2. Bilgi temsiliyeti (knowledge representation)
3. Makine öğrenmesi (machine learning)
4. Doğal dil işleme (natural language processing)
5. Konuşma ve görme modellemesi
6. Havacılık ve savunma, finans, web girişimleri, donanım tasarım firmaları ve bir çim biçme makinesi üreticisine kadar çeşitli ticari kullanım alanlarına sahiptir.

Fonksiyonel Paradigma
Programlar değerlendirilerek yürütülen tek bir deyimden (expressiondan) oluşur.
Değiştirilebilen veriden kaçınır.
Başka fonksiyonları argüman olarak alan üst düzey fonksiyonlar kullanılır.
Tamamen fonksiyonel dillerde yan etki kullanımı yasaktır.
Otomatik bir hafıza yönetimi vardır.
İşlemci kullanımı ve hafıza yönetimi açısından daha az verimlidir.

DESTEKLEDİĞİ PARADİGMALAR (İmperative, Declarative, Functional, Object Oriented etc.)

Haskell, fonksiyonel programlama paradigmasını (functional programming) destekler.

ARİTMETİK İŞLEM NOTASYONU(İnfix/Prefix/Postfix)

Haskell dilinde diğer dillerde bulunan birçok işleç bulunmaktadır. Aritmetik işleçler ortada(infix gösterim) yer alırlar. İnfix ifadeleri prefix biçimine dönüştürmek için işleç parantez içine alınabilir. Tersi olarak prefix biçimindeki bir işleci infix biçiminde kullanmak için tek tırnak(') sembolü kullanılabilir.

BELLEK YÖNETİMİ(Programcı Denetimli/Çöp Toplayıcı)

Dahili(otomatik) bir bellek yönetimi vardır. İşlemci kullanımı ve hafıza yönetimi açısından daha az verimlidir.

DEĞİŞKEN KAPSAMLARI(Lexical Scope/Dynamic Scope)

Haskell dilinde *sözcüksel olarak kapsam belirlenir (lexical scope).* Bir *blok* yeni bir kapsam tanımlar. Değişkenler bu kapsamda bildirilebilir ve dışarıdan görülemez. Ancak, kapsam dışındaki değişkenler (kapalı kapsamlarda) geçersiz kılınmadığı sürece görülebilir. Bu kapsam kuralları işlevler ve prosedürlerin adları için de geçerlidir.

TİP SİSTEMİ(Güçlü(Strong)/Zayıf(Weak))

&TİP KONTROLÜ(Static Type Checking/Dynamic Type Checking)

Haskell Hindley-Milner tip çıkarımı temelinde güçlü ve statiktir. Bu, hatalardan kaçınmanıza *yüksek derecede* yardımcı olur. Haskell'de hataların çoğu henüz derleme aşamasında yakalanır. Bunun asıl sebebi de tip çıkarımının derleme sırasında yapılmasıdır. Örneğin tip çıkarımı nerede yanlış parametreyi yanlış yerde kullandığınızı yakalar.

Statik tip sistemi hızlı çalıştırma için genelde önemlidir. Ama çoğu statik tip sistemli diller kavramları genellemede kötüdür. Haskell'in kurtarıcı yeteneği, tipleri kendi kendine *çıkarım* yaparak bulabilmesidir.

KOD ÖRNEKLERİ
PROBLEM ÇÖZÜMLERİ

Hello World! Application.hs

```haskell
module Main where

main :: IO ()
main = putStrLn "Hello world!"

```

"Hello world" UYGULAMASI KODU

```
C:\Program Files\Haskell Platform\8.6.5\bin\ghci.exe
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( C:\\Users\Lenovo\Desktop\Haskell
pplication.hs, interpreted )
Ok, one module loaded.
*Main> main ::IO()
Hello world!
*Main>
```

ÇIKTISI

fibonacci.hs

```haskell
fib = 1 : 1 : [a+b | (a,b) <- zip fib (tail fib)]

```

FİBONACCİ PROBLEMİ

```
C:\Program Files\Haskell Platform\8.6.5\bin\ghci.exe
*Main> take 15 fib
[1,1,2,3,5,8,13,21,34,55,89,144,233,377,610]
*Main> take 50 fib
[1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,121393,19
6418,317811,514229,832040,1346269,2178309,3524578,5702887,9227465,14930352,24157817,39088169,63245986,10233
4155,165580141,267914296,433494437,701408733,1134903170,1836311903,2971215073,4807526976,7778742049,1258626
9025]
*Main> take 75 fib
[1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,2584,4181,6765,10946,17711,28657,46368,75025,121393,19
6418,317811,514229,832040,1346269,2178309,3524578,5702887,9227465,14930352,24157817,39088169,63245986,10233
4155,165580141,267914296,433494437,701408733,1134903170,1836311903,2971215073,4807526976,7778742049,1258626
9025,20365011074,32951280099,53316291173,86267571272,139583862445,225851433717,365435296162,591286729879,95
6722026041,1548008755920,2504730781961,4052739537881,6557470319842,10610209857723,17167680177565,2777789003
5288,44945570212853,72723460248141,117669030460994,190392490709135,308061521170129,498454011879264,80651553
3049393,1304969544928657,2111485077978050]
*Main>
```

ÇIKTISI

mükemmelSayı.hs

```
1  mükemmelSayı :: Integral a => a -> Bool
2  mükemmelSayı n = n == sum [i | i <- [1..n-1], n `mod` i == 0]
3
```

MÜKEMMEL SAYI HESAPLAMA

```
C:\Program Files\Haskell Platform\8.6.5\bin\ghci.exe
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( C:\\Users\Lenovo\Desktop\Haskell
eted )
Ok, one module loaded.
*Main> mükemmelSayı 28
True
*Main> mükemmelSayı 50
False
*Main>
```

ÇIKTISI

asalSayı.hs

```
1   asalSayı n q = if (n `mod` q) == 0 then
2                  False
3               else
4                  if (q > (floor (sqrt (fromIntegral n)))) then
5                      True
6                  else
7                      asalSayı n (q+1)
8   main = do{  putStrLn "Bir Sayı giriniz: ";
9               x <- readLn;
10              if (asalSayı x 2) then
11                 putStrLn "Asal"
12              else
13                 putStrLn "Asal DEĞİL";
14          }
15
```

ASAL SAYI HESAPLAMA UYGULAMASI

```
C:\Program Files\Haskell Platform\8.6.5\bin\ghci.exe
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( C:\\Users\Lenovo\Desktop\Haskell denem kodları umarım\Göderdiklerim\asalSayı.hs, i
nterpreted )
Ok, one module loaded.
*Main> asalSayı 45 2
False
*Main> asalSayı 13 2
True
*Main>
```

ÇIKTISI

collatzSanısı.hs

```haskell
collatzSanısı :: Int -> [Int]
collatzSanısı 1 = [1]
collatzSanısı n = n:sequence
    where sequence
            | even n    = collatzSanısı (n `div` 2)
            | otherwise = collatzSanısı (n*3 + 1)

```

COLLATZ SANISI HESAPLAMA UYGULAMASI

```
C:\Program Files\Haskell Platform\8.6.5\bin\ghci.exe
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( C:\\Users\Lenovo\Desktop\Haskell denem kodları umarım\bok\collatzSanısı.hs, interp
reted )
Ok, one module loaded.
*Main> collatzSanısı 15
[15,46,23,70,35,106,53,160,80,40,20,10,5,16,8,4,2,1]
*Main> collatzSanısı 45
[45,136,68,34,17,52,26,13,40,20,10,5,16,8,4,2,1]
*Main>
```

ÇIKTISI

insertionSort.hs

```haskell
insert :: Int -> [Int] -> [Int]
insert x [] = [x]
insert x (y:ys) = if x < y
                  then x:y:ys
          else y : insert x ys
insertionSort :: [Int] -> [Int]
insertionSort [x] = [x]
insertionSort (x:xs) = insert x (insertionSort xs)

```

İNSERTİON SORT UYGULAMASI

```
C:\Program Files\Haskell Platform\8.6.5\bin\ghci.exe
GHCi, version 8.6.5: http://www.haskell.org/ghc/  :? for help
[1 of 1] Compiling Main             ( C:\\Users\Lenovo\Desktop\Haskell
hs, interpreted )
Ok, one module loaded.
*Main> a = [89,98,46,45,28,1,5,7,9,45]
*Main> insertionSort a
[1,5,7,9,28,45,45,46,89,98]
*Main>
```

ÇIKTISI

mergeSort.hs

```haskell
mergeSort :: (Ord a) => [a] -> [a]
mergeSort [] = []
mergeSort [a] = [a]
mergeSort a =
  merge (mergeSort firstFew) (mergeSort lastFew)
    where firstFew = take ((length a) `div` 2) a
          lastFew = drop ((length a) `div` 2) a
-- Expects a and b to already be sorted
merge :: (Ord a) => [a] -> [a] -> [a]
merge a [] = a
merge [] b = b
merge (a:as) (b:bs)
  | a < b     = a:(merge as (b:bs))
  | otherwise = b:(merge (a:as) bs)

```

MERGE SORT UYGULAMASI

ÇIKTISI

KAYNAKÇA

https://en.wikipedia.org/wiki/Haskell_(programming_language)

https://serokell.io/blog/haskell-history

https://courses.cs.washington.edu/courses/cse341/03wi/imperative/scoping.html

https://wiki.haskell.org/5_ad%C4%B1mda_Haskell

https://e-bergi.com/y/programlama-paradigmalari/

SQL, Lisp ve Haskell benim gördüğüm insanların yazmaktan çok düşündüğü tek programlama dilleridir.

Philip Greenspun