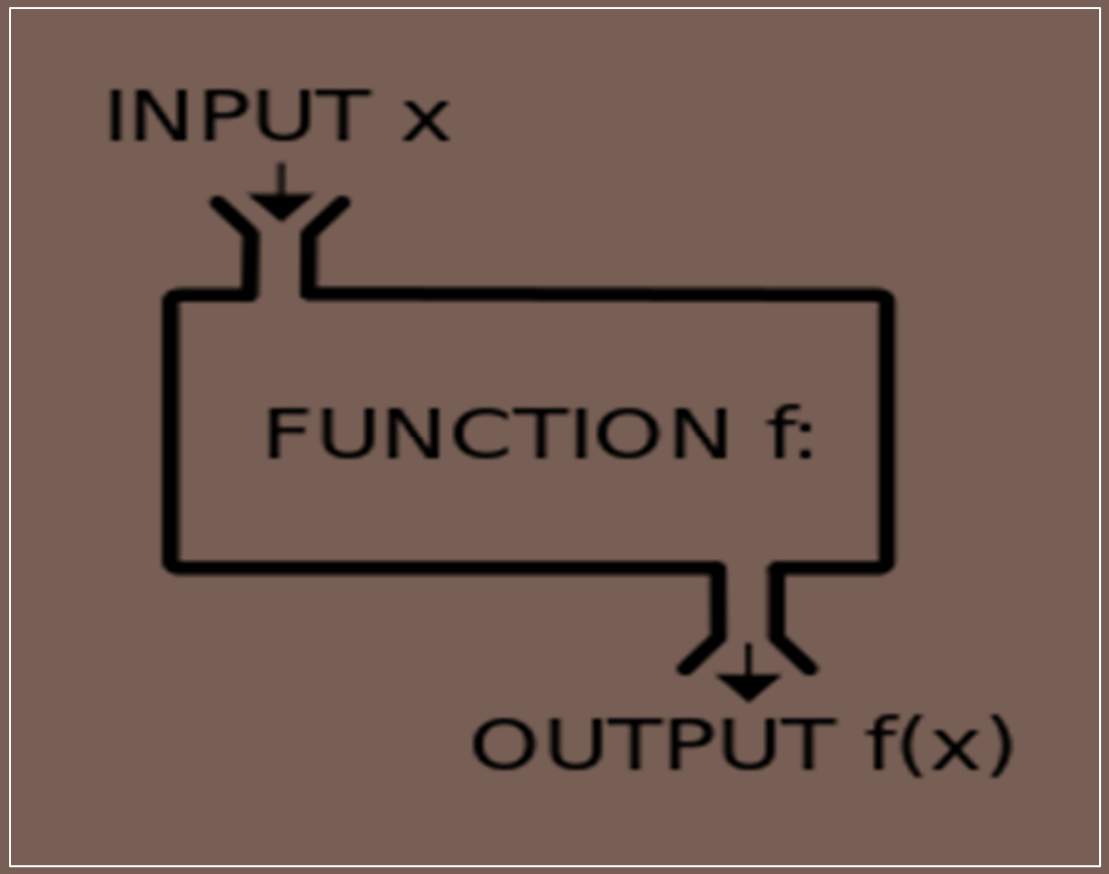


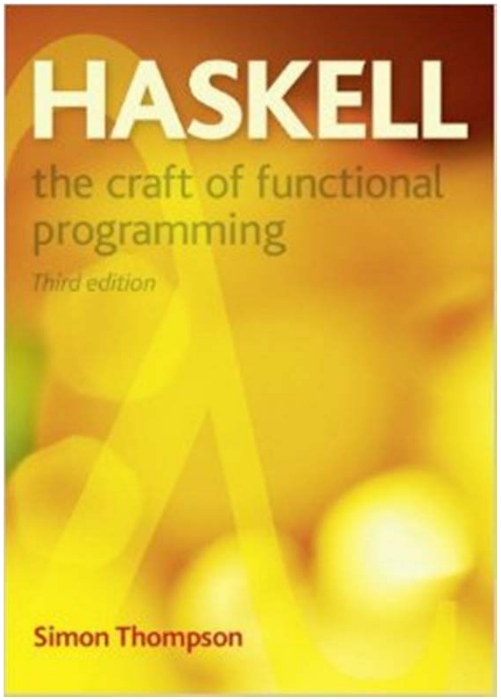
FUNCTIONAL PROGRAMMING
1 HASKELL BASICS

4/13/2014

Dr. Ahmed Sallam

Reference

- Main: Introduction to Haskell By Brent Yorgey
- Further



Functional Programming

- Function is the atom of the language, and can be used exactly as any other sort of value.
- Programming style is centered around evaluating expressions rather than executing instructions.

Haskell

- Named after logician Haskell Curry.
- It was created in the late 1980 by a committee of academics where everyone had his favorite functional language.
- It combines some of the best ideas from existing languages, and a few new ideas.

Haskell Characteristics

- Functional
- Pure
- Lazy
- Statically typed

Haskell Characteristics

- Pure
  - Immutable declarations

```
Prelude> let x = 5
Prelude> let f y = x + 1
Prelude> f 0
6
Prelude> let x = 7  -- not an assignment, a new declaration
Prelude> f 0
6
```

  - Referential transparency

```
y = f x
g = h y y
g = h (f x) (f x)
```

Haskell Characteristics

- Pure
  - No side effects: (e.g. updating global variables)
  - Parallelism: evaluating expressions in parallel.
- Lazy
  - Expressions are not evaluated until their results are needed.
- Statically Typed
  - Each expression has a fixed type. (expression with errors will not compiled)

Basic concepts

- Types
  - Haskell's rich of types.
  - Clarify thinking in the program structure
  - Serve as documentation
  - Turns run-time errors into compile-time errors.
- Abstraction
  - "Don't repeat yourself"
  - Polymorphism
  - Higher order functions

Basic concepts

- Wholemeanl programming
  - first solve a more general problem, then extract the interesting bits and pieces by transforming the general program into more specialised ones.

C language	Haskell

lst = [2,3,5,7,11]
 	
int total = 0;
for (int i = 0; i < lst.length; i++) {
  total = total + 3 * lst[i];
}
 	
print total;
	let lst = [2,3,5,7,11]
	
	let total = sum (map (3*) lst)
	
	map :: (Integer -> Integer) -> [Integer] -> [Integer]
	
	sum :: [Integer] -> Integer

Declarations and Variables

- Get in the mode

Basic Types

- Int : +/- 2^n (Calculate n as following)

```
(minBound :: Int, maxBound :: Int)
```

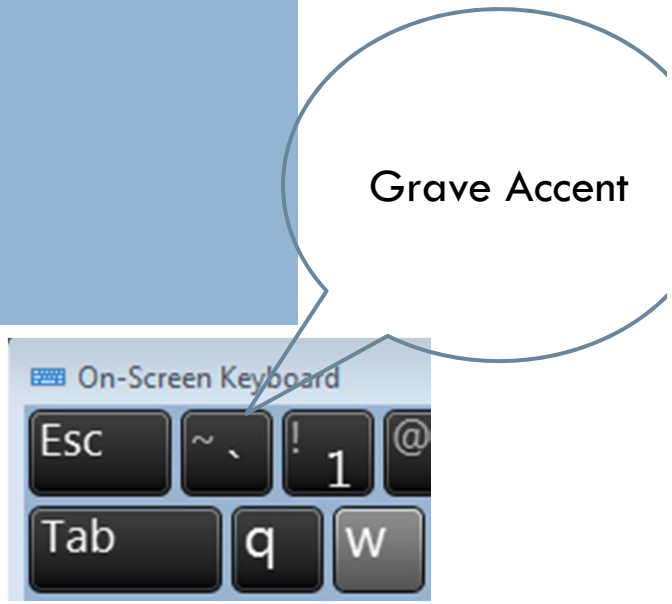
- Integer: limited by your machine memory.
- Float, Double: real numbers.
- Bool: Booleans.
- Char: Unicode Characters.
- String

GHCI Tips

- :load or :l is used to load .hs file.
- :reload or :r is used to reload .hs file
- :type or :t is used to ask for expression type
- :? Is used to show help (show available commands)

Arithmetic


```
ex01 = 3 + 2
ex02 = 19 - 27
ex03 = 2.35 * 8.6
ex04 = 8.7 / 3.1
ex05 = mod 19 3
ex06 = 19 `mod` 3
ex07 = 7 ^ 222
ex08 = (-3) * (-7)
```



- Note: mod operation

- Note: negative numbers

Arithmetic


```
i = 30 :: Int
n = 10 :: Integer

-- i+n  gives error
```

- Casting must be explicit
  - fromIntegral : ineger to all
  - Round, floor, ceiling
  - Use (div) instead of (/) with integers

Logic


```
ex11 = True && False
ex12 = not (False || True)

ex13 = ('a' == 'a')
ex14 = (16 /= 3)
ex15 = (5 > 3) && ('p' <= 'q')
ex16 = "Haskell" > "C++"
```

- Comparisons: (< , >, <=, >=, =, /=)
- Combine : (||, && , not ())

Logic


```
if b then t
        else f
```

- Else is mandatory

Function basic definition


```
-- Compute the sum if the integers from 1 to n.
mySum:: Integer -> Integer
mySum 0 = 0
mySum n = n + mySum (n - 1)
```

```
-- Pairs
sumPair :: (Int, Int) -> Int
sumPair (x,y) = x + y
```

```
-- Check even/odd numbers.
evenOdd :: Integer -> String
evenOdd n
  | n `mod` 2 == 0 = "Even"
  | otherwise      = "Odd"
```

Using function

- Functions has the highest priority.

```
sum :: Int -> Int -> Int -> Int
sum x y z = x + y + z
```

  - sum 5 n+1 3 parse as (sum 5 n) + (1 3)
  - If you want to pass n+1 as argument , write
    - sum 5 (n+1) 3

Lists

- a = 1 : []
- b = 3 : (1 : [])
- c = [2,3,4] == 2 : 3 : 4 : []

Function on lists


```
listlen :: [Int] -> Int

listlen [] = 0
listlen (h:t) = 1 + listlen t - - note how we spearate head from list
```

- We can replace h with _

Useful links

- Haskell Portal
  http://www.haskell.org/haskellwiki/Haskell

- Haskell Platform
  http://www.haskell.org/platform/