

Haskell Introduction (1A)

Copyright (c) 2016 Young W. Lim.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Based on

Haskell Tutorial by Medak & Navratil

<ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf>

Function

increment :: Int -> Int

function name

arg type

return type

Int increment (Int)

increment $x = x + 1$

Int arg

Int return

increment 2

3 :: Int

Function Examples

`and1 :: Bool -> Bool -> Bool`

function name

arg1 type

arg2 type

return type

Bool and1 (Bool, Bool)

`and1 a b = if a == b then a
 else False`

Bool arg1, arg2

Bool return

Function Examples

`and1 :: Bool -> Bool -> Bool`
`and1 True True = True` ← Rule (I)
`and1 x y = False` ← Rule (II)

`and1 :: Bool -> Bool -> Bool`
`and1 True True = True` ← Rule (I)
`and1 _ _ = False` ← Rule (II)

wild character

Where Statement – Local Definition

roots :: (Float, Float, Float) -> (Float, Float)

Input tuple

Output tuple

roots :: (a, b, c) = (x1, x2) where

x1 = e + sqrt d / (2*a)

x2 = e - sqrt d / (2*a)

d = b*b - 4*a*c

e = - b / (2*a)

p1, p2 :: (Float, Float, Float)

Variable declaration

p1 = (1.0, 2.0, 1.0)

Variable assignment

p2 = (1.0, 1.0, 1.0)

Variable assignment

Function Call Results

? roots p1 (-1.0,-1.0) :: (Float, Float)
(94 reductions, 159 cells)

? roots p2
(Program error: {primSqrtFloat (-3.0)}
(61 reductions,183 cells)

If-then-else Statement

roots :: (Float, Float, Float) -> (Float, Float)

Input tuple

Output tuple

roots :: (a, b, c) = (x1, x2) do

```
if d < 0
  then error "sorry"
  else (x1, x2)
```

where

```
x1 = e + sqrt d / (2*a)
x2 = e - sqrt d / (2*a)
d = b*b - 4*a*c
e = - b / (2*a)
```

Indentation

References

[1] <ftp://ftp.geoinfo.tuwien.ac.at/navratil/HaskellTutorial.pdf>