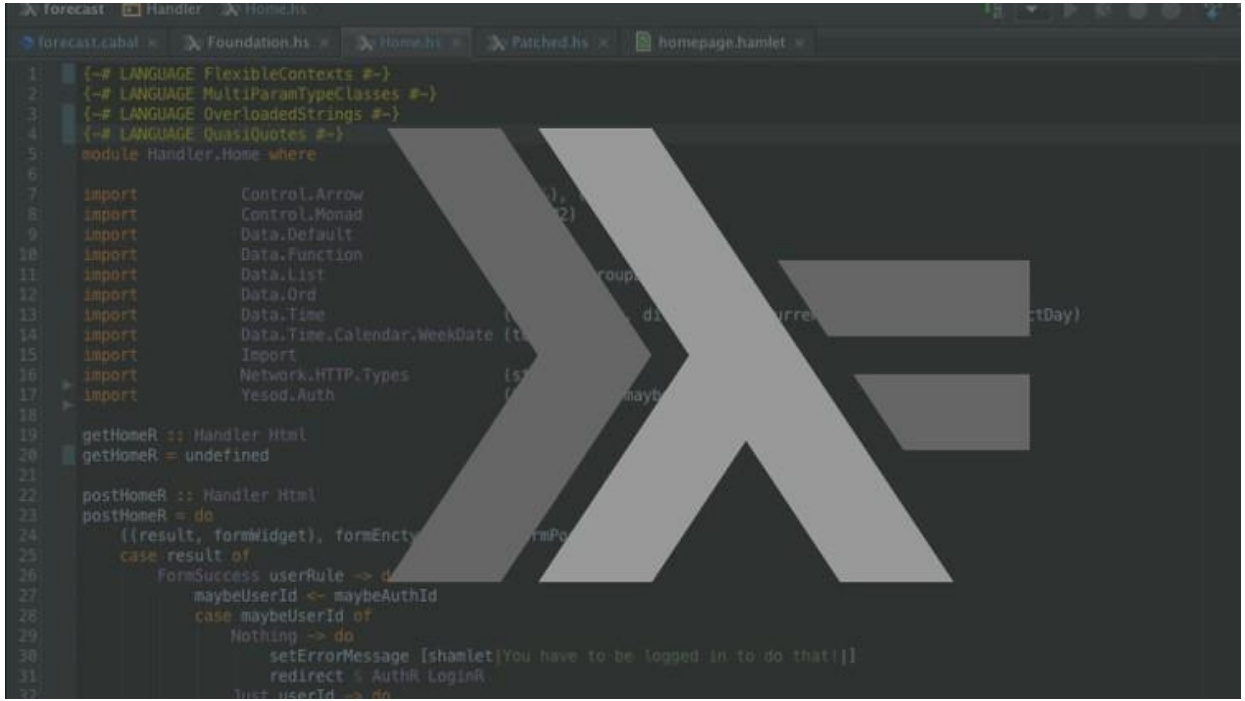
Basic Types


forecast.cabal
Foundation.hs
Home.hs
Patched.hs
homepage.hamlet
{-# LANGUAGE FlexibleContexts #-}
{-# LANGUAGE MultiParamTypeClasses #-}
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE QuasiQuotes #-}
module Handler.Home where
import Control.Arrow
import Control.Monad
import Data.Default
import Data.Function
import Data.List
import Data.Ord
import Data.Time
import Data.Time.Calendar.WeekDate
import Import
import Network.HTTP.Types
import Yesod.Auth
getHomeR :: Handler Html
getHomeR = undefined
postHomeR :: Handler Html
postHomeR = do
((result, formWidget), formEncty
case result of
FormSuccess userRule -> d
maybeUserId <- maybeAuthId
case maybeUserId of
Nothing -> do
setErrorMessage [shamlet|You have to be logged in to do that!|]
redirect $ AuthR LoginR

Basic Types

Booleans: `Bool`

Integers: `Int`, `Integer`

Reals: `Float`, `Double`

Characters: `Char`

Booleans

Type: `Bool`

Literals: `False` i `True`

Operations:

```
not  :: Bool -> Bool                -- negation
(||) :: Bool -> Bool -> Bool        -- disjunction
(&&) :: Bool -> Bool -> Bool        -- conjunction
```

Examples:

```
not True                    👉 False
not False                   👉 True

True || False               👉 True
True && False               👉 False

(False || True) and True    👉 True
not (not True)              👉 True
not not True                ❌ -- it means: (not not) True
```

Integers

Type:

- `Int`: Integers of 64 bits
- `Integer`: Integers (arbitrarily long)

Literals: `15`, `(-22)`, `85732635487345264468`

Operations: `+`, `-`, `*`, `div`, `mod`, `rem`, `^`.

Relational operators: `<`, `>`, `<=`, `>=`, `==`, `/=` (⚠ no `!=`)

Examples:

Integers

Examples:

```
3 + 4 * 5                    👉 23
(3 + 4) * 5                  👉 35
(3 + 4) * 5                  👉 35
2^10                         👉 1024
3 + 1 /= 4                   👉 False

div 11 2                     👉 5
mod 11 2                     👉 1
rem 11 2                     👉 1
mod (-11) 2                  👉 1
rem (-11) 2                  👉 -1
```

Reals

Type:

- `Float`: 32-bit floating-point reals
- `Double`: 64-bit floating-point reals

Literals: `3.14`, `1e-9`, `-3.0`

Operations: `+`, `-`, `*`, `/`, `**`.

Relational operators: `<`, `>`, `<=`, `>=`, `==`, `/=`

Integer to Real conversion: `fromIntegral`

Real to Integer conversion: `round`, `floor`, `ceiling`

Reals

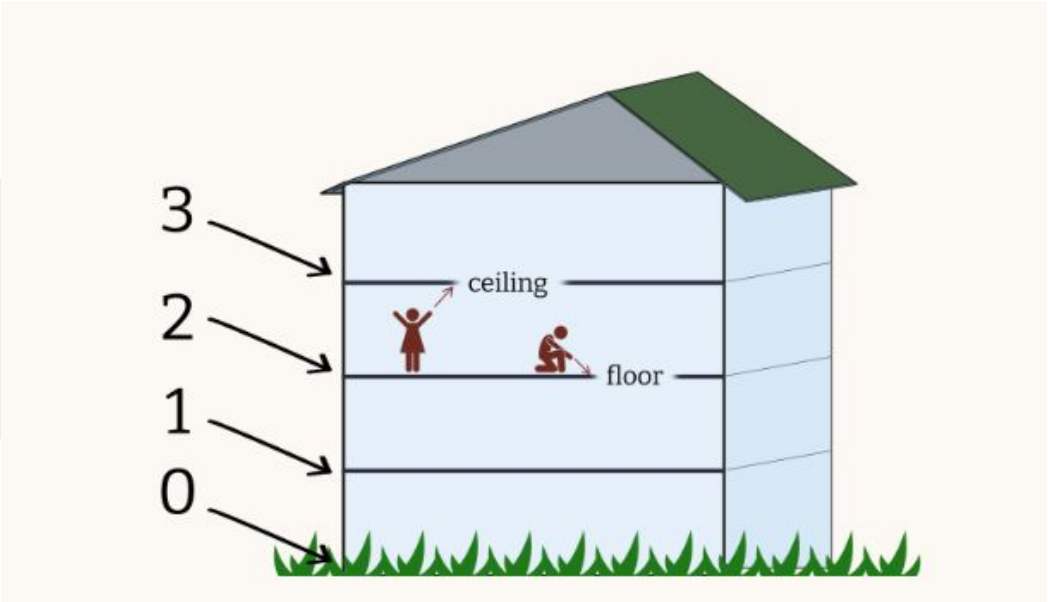
Examples:

```
λ> round 3.6
4

λ> round (-3.6)
-4

λ> map round [3.5, 4.5, 5.5, 6.5]
[4,4,6,6]
```

Reals

Examples:

```
λ> map ceiling [2.1, 2.2, 2.6, 2.9]
[3,3,3,3]

λ> map floor [2.1, 2.2, 2.6, 2.9]
[2,2,2,2]
```



Reals

Examples:

```
10.0 / 3.0      👉 3.3333333333333335
2.0 ** 3.0      👉 8.0
fromIntegral 4  👉 4.0
```

Characters

Type: `Char`

Literals: `'a'`, `'A'`, `'\n'`

Relational operators: `<`, `>`, `<=`, `>=`, `==`, `/=`

Conversion functions: (it is necessary `import Data.Char`)

- `ord :: Char -> Int`
- `chr :: Int -> Char`

Operator Precedence

9	`!!`		`.`
8			`^`, `^^`, `**`
7	`*` `/` `div` `mod` `rem` `quot`		
6	`+` `-`		
5			`:` `++`
4		`==` `/=` `<` `<=` `>` `>=` `elem` `notElem`	
3			`&&`
2			`||`
1	`>>` `>>=`		
0			`$` `$!` `seq`

Predefined Functions

is even / odd

```
even :: Integral a => a -> Bool
odd  :: Integral a => a -> Bool
```

minimum and maximum of two values:

```
min :: Ord a => a -> a -> a
max :: Ord a => a -> a -> a
```

Predefined Functions

greatest common divisor, least common multiple:

```
gcd :: Integral a => a -> a -> a
lcm :: Integral a => a -> a -> a
```

mathematicals:

```
abs  :: Num a      => a -> a
sqrt :: Floating a => a -> a
log  :: Floating a => a -> a
exp  :: Floating a => a -> a
cos  :: Floating a => a -> a
```

Summary

- Haskell provides predefined basic types for:
  - booleans (`Bool`),
  - integers (`Int` and `Integer`),
  - reals (`Float` and `Double`), and
  - characters (`Char`).
- Each type provides the logical, arithmetic, and relational operations common to many PLs.
- Attention: ⚠
  - It is necessary to indicate the negative sign with parentheses: `(-22)`.
  - `mod` and `rem` work differently with negative numbers.
  - The ≠ is written `/=` and not `!=`.