하스켈과 타입

홍민희
http://hongminhee.org/

Data type

- 널리 쓰이는 번역어: **자료형**
- 추측할 수 있듯, 자료(data)를 분류(type)하는 것
- 그럼 분류한 자료들은 어떻게 생긴 것일까?

Value

- 널리 쓰이는 번역어: **값**
- 어딘가에 적혀서 나중에 읽힐 수 있다
- 이쪽에서 저쪽으로 전해줄 수 있다

Value

- 널리 쓰이는 번역어: **값**
- 메모리나 디스크에 저장하고 나중에 열어볼 수 있다
- 네트워크를 통해 전송될 수 있다

Value

- 널리 쓰이는 번역어: **값**
- 이름지어질 수도 있고, 이름이 어떤 값을 뜻하는지도 알 수 있다
- 함수 인자로 들어갈 수도 있고, 함수로부터 나올 수도 있다

값

```
> let factorial n = product [1..n]

> let x = 10

> factorial x

3628800
```

이름지어질 수 있다

```
> let factorial n = product [1..n]

> let x = 10

> factorial x

3628800
```

이름의 뜻을 알 수 있다

```
> let factorial n = product [1..n]

> let x = 10

> factorial x

3628800
```

함수 인자로 들어갈 수 있다

```
> let factorial n = product [1..n]

> let x = 10

> factorial x

3628800
```

함수로부터 나올 수 있다

```
> let factorial n = product [1..n]

> let x = 10

> factorial x

3628800
```

값의 성질

- 값들은 저마다 독특한 성질이 있다
- 1 다음은? 2. 그럼 2 다음은? 3.
- `['a', 'b', 'c']`의 길이는? 3.
- `[]` 다음은? 목록한테 '다음'이란 뭘까?
- 1의 길이는? 정수한테 '길이'란 뭘까?

값의 성질

```
> succ 1

2

> succ 2

3

> length ['a', 'b', 'c']

3
```

값의 성질

```
> succ ['a', 'b', 'c']


<interactive>:16:1:
    No instance for (Enum [Char]) arising from a
use of ‘succ’

    In the expression: succ ['a', 'b', 'c']

    In an equation for ‘it’: it = succ ['a', 'b',
'c']
```

값의 성질

```
> length 1


<interactive>:17:8:
    No instance for (Num [a0]) arising from the
literal ‘1’

    In the first argument of ‘length’, namely ‘1’

    In the expression: length 1

    In an equation for ‘it’: it = length 1
```

자료형

- 값들은 저마다 독특한 성질이 있다
- 그러나 성질이 흡사한 것끼리 모아서 분류할 수 있다
- ‘다음’이 있고, 두 값이 ‘다음’ 값들이 ‘같으면’ 두 값이 ‘같고’, ‘더할 수’ 있고, ‘뺄 수’도 있고… → **정수** 자료형
- 0개 이상의 값을 ‘포함’할 수 있고, 포함된 값들 사이에 ‘순서’가 있고… → **목록** 자료형

자료형

- 비슷한 성질의 값들을 모아두면 편리하다
- 가령, 덧셈을 할 때 '더할 수 있는' 성질의 값들만 받고 싶다 → **수** 자료형만 받으면 된다
- 가령, 갯수를 구할 때 '값을 포함할 수 있는' 성질의 값들만 받고 싶다 → **목록** 자료형만 받으면 된다

factorial은 정수만 받는다

```
> factorial ['a', 'b', 'c']


<interactive>:26:1:
    No instance for (Num [Char]) arising from a
use of ‘factorial’

    In the expression: factorial ['a', 'b', 'c']

    In an equation for ‘it’: it = factorial ['a',
'b', 'c']
```

값의 자료형

(실제는 이보다 좀더 복잡합니다)

```
> let factorial n =

      product ([1..n] :: [Integer]) :: Integer
```

```
> let x = 10 :: Integer

> factorial x :: Integer

3628800 :: Integer
```

자료형 알아보기

(실제는 이보다 좀더 복잡합니다)

```
> :t x
x :: Integer

> :t factorial
factorial :: Integer -> Integer
```

여기선 다루지 않는 다음 주제

- 대수적 자료형 (algebraic data type)
- Parametric polymorphism
- 타입클래스 (typeclass)
- ⊥ (bottom)
- 기타 등등. 하지만 몰라도 일단 하스켈 코딩 시작 가능

부실해서 죄송합니다

bit.ly/haskell-kr-2015-04-04-type