00:52 34
All iCloud
Suggested Title: « records » Edit
2 November 2022 at 00:51

- records
- cardinality
- ~~List~~ .all
  .reverse
  .map
  .filter
- ~~renaming import modules~~
- sum types
- ~~exposing~~
- ~~howBig~~
- predicate
- function types
- ~~point-free style~~
- ~~just, maybe~~

- (_) → anything besides what is enum. def.
- [x] / x::[] > list with only 1 element
- predicate = function which returns a boolean
- type variables
  - range over types
  - indicate same type between multiple arg.
  - only described in lowercase
- Msg type = the state of the app
- Html.test.Selectors (equal / has
- type classes (False)
  - used to def classes
  - must be in the module
- Wrong use records: field in func signature

Lab 1

- type inference
- function signature : write its name
- constants = functions which take no parameters
- variable shadowing is an error
- IF without ELSE => error
- expressions return a value while statements have side effects
- NO looping mechanisms

→ fact 5 - 1 = fact 5 - 1
fact(5 - 1) = fact 4

00:52 34
All iCloud

- predicate
- function types
- ~~point-free style~~
- ~~just, maybe~~

- Msg type = the state of the app
- Html.text. Selectors [equal
  has

- type classes (False)
  [used to def classes
  [must be in the module

- Wrong use records: field in func signature

Lab 1

- type inference
- function signature: write its name
- constants = functions which take no parameters
- variable shadowing is an error
- IF without ELSE => error
- expresions return a value while statements have side effects
- NO looping mechanisms
  → fact 5 - 1 = fact 5 - 1
  fact(5 - 1) = fact 4
- tail recursive [returns something computed directly or ↑ ret. by its recursive call
  [use additional accumulator
  [pass 1 as acc each time

Lab 2

TUPLES
- heterogenous = can contain different types of data

00:52 34
All iCloud

Lab 2

TUPLES
- heterogenous = can contain different types of data
- at most 3

ex: (1, ("Hello", 2), [3, 4])

RECORDS
- collection of named fields
- similar to STRUCT

ex: fullName : { first : String , last : String }
| → String
| fullName person = person.first ++ " "
| ++ person.last

TYPE ALIASES
- give a new name to existing types

ex: type alias User = { first : String
, last : String }

fullName User → String
User "Ariana" "Lazăr"

TYPE DEFINITIONS
- allow creation of new types

ex: type Color = R | G | B (1)

Color: type name
R | G | B: type constructors

- can define one type but all kinds

00:52 34
All iCloud

- can define one type but diff kinds of type cons ex: type Point = Point Int Int ②
  ↓ also cons name

FOR SUM TYPES
↓
cardinality = nr. of variants
ex: FOR ① is 3
ex: Int has card = 2^{32}

PRODUCT TYPES
cardinality = product of the cardinality of each field
ex: For ② is $2^{32} * 2^{32}$

LET ... IN
- declare bindings and use them in local scope
- used to avoid shadowing
  ex: doubble n = n * 2
  let n = 10 in double n
- keep helper functions local

PATTERN MATCHING
- works like switch
- extract data from variants
- order matters (TOP-DOWN)
- at least one pattern must match

00:52 34
All iCloud

Cannot define Int as sum type

Cardinality (Bool) = 2

Lab 3

- lowercase letters = type variables
- lowercase names = type constraints

TYPE VARIABLE ?

TYPE CONSTRAINTS

- appendable (STRING and LISTS)
- number (INT and FLOAT)
- comparable (nr, char, Str...)

MAYBE

- enumerated types → well def res

```
type Maybe a = Just a
             | Nothing
               ↳ no result
```

RESULT

- signal the failure → success

```
type Result error value = Ok value
                        | Err error
                          ↓
```

00:52 34
All iCloud

Err error
↓
error type

Ex: safeArea : Shape → Result String Float

```
safeArea shape =
  case shape of
    Circle radius →
      if radius < 0 then
        Err "radius < 0"
      else
        OK (pi * radius * radius)
```

LISTS

- singly linked lists
- cannot have a list of diff types

Definition: [1, 2, 3]

1::2::3::[]

Functions

- length
- sum
- head: 1st elem
- tail: items after 1st
- range: List.range 1 5
  [1,2...5]
- reverse
- append keyword = "++"

00:52 34
All iCloud

- concat
- "::" = CONS operator (used to add an element to the front of the list)

ex: 1 :: [2,3] => [1,2,3]

- drop = drops the specified nr. of elem. from the beggining of the list and returns a list with remaining elements
- take = lets the specified nr. of elements in a new list
- zip = new lists which contain the first from each tuple
  2nd -11-
  3rd -11-
  ⋮
- unzip = new lists st: primul cu primul
  2nd cu 2nd
  ⋮
- map = applies the given function to each element and puts the result in a new list

ex: lengths = List.map String.length guardians
List.filter (\x -> x < 6) lengths

\x -> String.contains "-" x && as long

00:52 34
All iCloud

`\x -> String.contains "-" x` = `String.contains "-"` } as long as x is the last arg.

- filter

ex: `isValid char = char /= '-'`
`String.filter isValid "222-5"`
↓
if isValid = true => keep
if not => delete

`List.filter (\x -> x < 6)`

- foldl = reduce a list

`List.foldl (\x a -> x + a) 0 [1,2,3,4]`

combining function | initial value | list that needs folding

Result: 10

next acc

Passes the first element as 1st arg
TAIL REC

- foldr = reduce a list from right

`(\x a -> x ^ a) 0 [1,2,3,4]`

NOT TAIL REC

- all, any

00:52 34
All iCloud

- all + any

  all isEven [2,4] == True
  any isEven [2,3] == True

- EXPOSING A FUNCTION

  module Date exposing (function1)

- EXPOSING a type + all variants

  module Date exposing (function1, Month(...))

- EXPOSING only a type

  module Date exposing (Date)

- Renaming: import Date as D

  D.daysInMonth

- Debug.todo + String => (can replace any func or arg; you want to implem. a func. but won't use it right away)

- Higher order functions
  → take f as args or ret a f.
  → parantheze

00:52 34
All iCloud

a curve. next movie run a right away)

- Higher order functions
  → take f as args or ret a f.
  → parantezi
- POINT FREE STYLE
  - hide param f is applied to
- LAMBDA
  - anonymous f used as args to other f