Σχεδίαση και Υλοποίηση της Γλώσσας Προγραμματισμού lambda-cases

Δημήτρης Σαριδάκης Μπίτος

Εθνικό Μετσόβιο Πολυτεχνείο

7 Ιουλίου 2024

Μ′ αρέσει πολύ η Haskell

Τα πάντα είναι τιμές (όροι) και έχουν κάποιο τύπο:

- Σταθερές
- Συναρτήσεις
- Είσοδος/Έξοδος
  - Άρα μπορούν να είναι ορίσματα συναρτήσεων, στοιχεία λίστας κτλ

Οι τύποι τα λένε όλα.

Βοηθητικός μεταγλωττιστής:

- Μεταγλωττίζεται; Δουλεύει! (Συνήθως)
- Δεν μεταγλωττίζεται; Οι τάδε τύποι δεν ταιριάζουν.

Γιατί να γράψω κώδικα σε άλλη γλώσσα;

Γιατί δεν είναι η πιο διαδεδομένη γλώσσα;

Building εργαλεία όχι τόσο καλά:

- Φαίνεται να υπάρχει βελτίωση (απ'όσο λένε online)
- Δεν αφορά την διπλωματική

Δύσκολη στην εκμάθηση για αρχάριο. Ίσως παίζουν ρόλο:

- Όχι πολύ περιγραφικές λέξεις κλειδιά
- Όχι πολύ περιγραφικά ονόματα βασικών συναρτήσεων
- Γραμματική λάμδα λογισμού

Τι θα άλλαζα για μένα;

Μπορούν να συμπτυχθούν κομμάτια που γράφω πολύ συχνά;

- Ορισμοί Τιμών
- LambdaCase extension

Μπορούν να αλλάξουν κομμάτια ώστε να μοιάζουν περισσότερο στα αντίστοιχα άλλων γλωσσών όπου είναι πιο κατανοητά;

- Τελεία για attributes/members/fields
- Εφαρμογή συνάρτησης με ορίσματα σε παρένθεση

Υπάρχει κάτι καινούργιο που θα μπορούσα να προσθέσω;

- Ορίσματα στην αρχή ή στην μέση του ονόματος της συνάρτησης
- Ανώνυμες Παράμετροι

Μπορώ να δώσω πιο κατανοητά ονόματα;

Εφαρμογή Συνάρτησης Με Παρενθέσεις

Haskell	lcases
`f x`	`f(x)`
`g x y z`	`g(x, y, z)`
`putStrLn "Hello World!"`	`print("Hello World!")`

`show x`	`(x)to_string`
`mod x y`	`(x)mod(y)`
`map f l`	`apply(f)to_all_in(l)`

Ορισμό apply to all;

Ανώνυμες Παράμετροι

```
f(x, y, _), f x y

f(x, _, z), \y -> f x y z

f(_, y, z), \x -> f x y z

f(_, _, z), \x y -> f x y z
```

Παραδείγματα

και για λίστες tuples

Ορισμοί tuple_type και postfix functions

tuple_type αντίστοιχα:

- structs σε C
- classes σε OOP: μόνο attributes
- records σε Haskell

Δημιουργείται αυτόματα ένα postfix function για κάθε field:

- Κατευθείαν με όρισμα: `some_person.last_name`
- Συνάρτηση Μόνη της: `_.last_name`

Ορισμοί tuple_type και postfix functions

```
tuple_type Name
value (first_name, last_name) : String^2

awesome_guy: Name
  = ("Leonhard", "Euler")

>>> awesome_guy.last_name
  : String
  = "Euler"

>>> _.last_name
  : Name => String

>>> apply(_.last_name)to_all_in(_)
  : ListOf(Name)s => ListOf(String)s
```

postfix functions για product type tuples

Λίστα

Παραδείγματα

".change" postfix function

Συναρτήση αλλαγής στοιχείων

Παραδείγματα

Ορισμοί or_type και prefix functions

Παραδείγματα

Τελεστές

Πίνακας

Εκφράσεις Συναρτήσεων

Εκφράσεις Συναρτήσεων "cases"

pattern matching

LambdaCase extension

Ορισμοί Τιμών

Σύγκριση με Haskell

Εκφράσεις "where"

Παραδείγματα

Τύποι

Αντιστοιχία με Haskell

Πατσούκλια Τύπων

type στην Haskell

Παραδείγματα

Λογική Τύπων

Μηχανισμός ad hoc πολυμορφισμού στην lcases.

Αντιστοιχοί στα type classes.

Ορισμοί Προτάσεων Τύπων

Ατομικές, class

Μετονομασίας

Θεωρήματα Τύπων

instance

Υλοποίηση Parser

Βιβλιοθήκη Parsec

Μετάφραση σε Haskell

Συμπεράσματα

Τι έχει γίνει

Τι θα ήταν καλό να γίνει