Monade

Monade so matematični pojem, ki posplošuje algebrajske strukture, kot so grupa, kolobar in vektorski prostor. Hkrati pa se monade uporablja tudi v teoretičnem računalništvu in deklarativnem programiranju za opisovanje *računskih učinkov*. V tej lekciji bomo spoznali osnovno uporabo monad v programiranju.

Haskell je čist programski jezik in vse računske učinke predstavi z monadami. Programer lahko v Haskellu definira nove monade in z njimi simulira razne računske učinke. Tudi drugi programski jeziki imajo monade, le da niso neposredno dostopne programerju. Jeziki kot so OCaml, Java, C++ uporabljajo le eno monado, ki predstavlja vse računske učinke, podprte v jeziku (I/O, izjeme, stanje).

Računski učinki

Računski učinki so širok pojem, ki ga je težko povsem opredeliti. Lahko rečemo, da je računski učinek vsak pojav v programu, ki ni le preprosto procesiranje informacije. Med računske učinke štejemo:

- **manjkajoča vrednost**: program lahko ugotovi, da rezultat ne more izračunati, na primer, ker ena od vmesnih operacij ni definirana
- **izjeme**: program lahko nenadoma prekine izvajanje in javi napako ali vrže izjemo (`throw`, `catch`)
- **vhod/izhod**: program lahko komunicira z zunanjim svetom preko vhodnega in izhodnega kanala (`input`, `print`)
- **stanje**: program ima trenutno stanje, ki se spreminja, ko program teče (`set`, `get`)
- **nedeterminizem**: program lahko izbira med večimi možnostmi
- **verjetnostno računanje**: program lahko izbira med večimi možnostmi z verjetnostno porazdelitvijo
- **timeout**: izvajanje se lahko prekine, če traja predolgo

Poznamo pa še veliko drugih računskih učinkov.

Monade

Zgoraj našteti računski učinki so zelo raznoliki, a kljub temu imajo skupno strukturo, ki jo zajema pojem monade. Razložimo ga iz stališča programerja, ki želi *simulirati* računske učinke v "na roko", se pravi v čistem jeziku.

Najprej ločimo med **(čisto) vrednostjo** (pure value) in **izračunom** (computation). Čista vrednost je podatek ali koda, ki nima nobenih računskih učinkov, izračun pa je koda, ki predstavlja rezultat *in* računske učinke.

Izračune predstavimo s podatkovnim tipom, ki izraža naravo računskih učinkov, ki jih lahko izračun sproži. Vsak računski učinek ali kombinacija učinkov ima svoj podatkovni tip.

Primer: Izračun, ki vrne rezultat tipa `a` ali pa razultata ni, predstavimo s tipom `Maybe a`.

Primer: Izračun, ki lahko izpisuje na standardni izhod, predstavimo s tipom `(String, a)`.

Primer: Izračun, ki lahko bere s standardnega vhoda, predstavimo s tipom `String -> a`.

Primer: Izračun, ki ima stanje tipa `s` in rezultat tipa `a`, prestavimo s tipom `s -> (s, a)`.

Nadalje ima vsak od teh tipov še dve operaciji:

1. Vedno lahko konstruiramo *čisti izračun*, ki predstavlja vrednost brez računskih učinkov.
2. Izračune lahko *komponiramo* tako kot funkcije, pri čemer se računski učinki ustrezno komponirajo.

Pri tem veljajo nekatere zakonitosti, ki jih bomo spoznali kasneje.

Monade v Haskellu

Vse do sedaj našteto je v Haskellu predstavljeno z razredi tipov:

- `Functor`
- `Applicative`
- `Monad`

Spoznali jih bomo na primerih in si ogledali v živo, kako delujejo.