Funkcionális programozás a gyakorlatban gyakorlat

Iván Szabolcs – Számítástudomány Alapjai Tanszék,
SZTE TTIK

λ

flatMap

Bevezetés

Mi a cél?

A funkcionális paradigma, melyben az imperatív paradigmában megszokott változók, ciklusok, mutable state helyett (immutable) értékeket, függvényeket és rekurziót használunk, erős típusozottsággal, mellékhatások nélkül, olyan programok fejlesztését teszi lehetővé, melyek – kis túlzással – “ha lefordulnak, akkor helyesek is lesznek”. Mindezt kombinálva a Scala mint JVM programnyelv teljes hozzáférésével a Java csomagrendszeréhez, és hogy egy projekten belül aszerint választhatunk az imperatív Java és a funkcionális Scala implementáció közül, hogy az adott feladatra éppen melyik az alkalmasabb, egy hatékony, gyors és helyes fejlesztési ciklust lehetővé tevő környezetet kapunk.

A kurzus egyrészt betekintést ad a funkcionális paradigmába oly módon, hogy az más FP nyelvekben is felhasználható legyen, másrészt a Scala nyelv egyes specifikus elemeit is ismerteti a Java-Scala integrációval együtt, hogy a kurzust elvégzettek képesek legyenek dolgozni adott esetben egy már létező Scala frameworkkel.

Milyen előzetes tudásra épít?

A tananyag létrejöttét indokló kurzus – tekintve, hogy második féléves kötelező tárgy az üzemmérnök informatikus szakon – minimális előfeltételeket támaszt: a nyelvi elemeket és a paradigma alapkoncepcióit is a nulláról igyekszik felépíteni. Ugyanakkor, a rekurzió megértéséhez valamennyi algoritmus-tervezési érzék szükséges lehet, és az imperatív nyelvi elemekkel is összevetjük a funkcionálisakat, ily módon egy imperatív programozás kurzus (mint amilyen az előfeltétel Programozás alapjai) anyagának ismerete szükséges, ha nem is egy konkrét programozási nyelvé vagy környezeté.

A félév közepe felétől elmozdulva a core FP paradigmától a Scala nyelv elemeire fokozatosan kerül át a hangsúly, ily módon a félév során a Java nyelv elemeinek megismerése is szükségessé válik, melyet a kurzus párhuzamos feltételén, a Programozás I-en tanulnak a szak hallgatói.

Milyen referenciákra épül? Miért hasznos egy újabb tananyag?

A szerző elvégezte a Scala nyelvet és a javac fordítót megalkotó Martin Odersky nevével fémjelzett Functional Programming in Scala specializációt Courserán. A kurzus anyagának egy része – részben azért is, hogy a hallgatók a nemzetközi terminológiát is megismerjék – hasonló ezen kurzuséhoz, azonban attól annyiban mindenképp eltér, hogy olyanok számára is elérhető, akik a Java nyelvvel még csak ismerkednek (a Courserás specializáció bevallottan olyanok számára készült, akik olyan erős Java, OOP és tervezési minta alapokkal rendelkeznek, mely a BProf képzés elejében tartóktól semmiképp nem várható el).

A szerző szerencsésnek tartja, ha a funkcionális paradigmával a hallgatók már a képzés elején megismerkednek, minél később kell “felülírniuk” az imperatív nyelvekben tanult ismereteket és szemléletmódot, annál nehezebbé válik ez a váltás. Ily módon a tananyag – ugyan mélységében természetesen nem kíván versenyezni sem a fentebb említett specializációval, sem egy kifejezetten Scala-specifikus kurzussal – egy olyan “early expose” a hallgatók számára, amilyen az SZTE berkein belül nem létezik (a Funkcionális programozás kurzus állna hozzá legközelebb, de a szerző ismeretei szerint a bolognai átmenet során ez a kurzus 2005 óta lényegében megszűnt, és egy egész más, főként akadémiai szférában alkalmazott programozási nyelven, a Haskellen keresztül ismertette a paradigmát).

Mennyire használható és naprakész tudást ad?

A Java Virtual Machine eszközök milliárdjain fut, androidos telefonoktól kezdve minden számítástechnikai eszköz (kivéve talán a mikrokontrollereket, melyeket leginkább natív C-ben programoznak ma is) képes ezáltal Java bytecode-ot futtatni. Mivel a Scala nyelv fordítója egy teljesen JVM-kompatibilis bytecode-ot készít, ez azt jelenti, hogy a Scalában írt programok minden nehézség nélkül (mind fejlesztői, mind felhasználói szemszögből) futtathatóak szinte tetszőleges architektúrán és környezetben. A tananyagon keresztül szerezhető kompetenciáknak része, hogy az olvasó a végére képes lesz felmérni, hogy egy adott feladatot inkább a szokott imperatív módon, vagy funkcionálisan lenne jobb megoldani, és képes lesz eddigi ismereteit és eszköztárát nagyon hatékonyan bővíteni. A Scala nyelv népszerűségét és használhatóságát mutatja, hogy egy szakmai lista szerint 2021-ben data science és cloud computing (a large scale alkalmazások fejlesztésénél nagyon hasznos, ha a programunk függvényei garantáltan mellékhatás-mentesek, számos hibalehetőség eleve fel sem merül ekkor, az említett két terület pedig a szoftverfejlesztésben nagyon is “hot topic” manapság) célra a top 5 nyelvben szerepel, a linkelt oldal szerint is ”Scala is currently considered to be one of the best programming languages for functional programming”.

Milyen formában adja át a tudást? Milyen eszköztárakat használ?

A tananyag egy önálló weboldalon érhető el. Az anyag kódrészleteket is tartalmaz, gyakori közbeszúrt kérdésekkel, melyekre a válasz lenyitható, önellenőrzés céljára, erősen épít a tantárgy előadásának anyagára, mely szintén ugyanezen weboldalon érhető el. A lényegesebb trükkök, programozói fogások, “best practice”-ek kiemelve szerepelnek; számos aloldalhoz tartozik egy-egy maximum 20 perces videósegédlet (melyek az egész tárgyra vonatkoztatva külön gyűjtve is elérhetőek egy YouTube playlistben, ha valaki egyfajta “podcast”ként szeretné inkább feldolgozni az anyagot).

Hogyan érdemes használni a tananyagot?

A gyakorlati tananyag tizenöt és kisebb-nagyobb alpontra oszlik. Egyszerre érdemes egy alpontot feldolgozni, elolvasva az írásos anyagot és (ajánlottan) átnézve a videóanyagot, oly módon, hogy közben egy fejlesztőkörnyezetben a hallgató ki is próbálja a példakódokat és megpróbálja meg is oldani a kérdésként feltett szoftverfejlesztési feladatokat, és miután ezt (sikeresen vagy sem) megtette, nézi meg a megoldást. A gyakorlati leckék megoldása előtt feltétlenül ajánlott a kapcsolódó előadás lecke vagy leckék alapos átnézése és az ottani kérdések megválaszolása. A gyakorlat feladatainak tényleges megoldása egy fejlesztőkörnyezetben feltétlenül szerves része kell legyen a tanulási folyamatnak ahhoz, hogy a tananyag elsajátítása sikeres legyen, elvégre a tárgy címében is szerepel “a gyakorlatban” szókapcsolat – a feladatok, visszakérdések tényleges kipróbálása, a fejlesztési folyamatban való aktív részvétel nélkül ez nem lehetséges.
Több esetben a feladatok, kérdések nehezek, és egy rövid, idiomatikus válasz megtalálása sokkal tovább tart, mintha a begyakorolt imperatív módon oldaná meg az ember az adott problémát, ez nem szabad elkeserítsen senkit – pontosan az az egyik cél, hogy ráébressze az olvasót a tananyag, a helyes válaszain keresztül, hogy ha megtanulja használni az FP eszköztárát, mennyi “standard” felmerülő problémára kínál out-of-the box, garantáltan helyes és mellékhatás-mentes megoldást, ráadásul egy könnyebben olvasható és így egyszerűbben tesztelhető és karbantartható kódot lehetővé téve.

Témakörök

Szoftverfejlesztés Scalában, pure FP paradigma mellett

- a fejlesztőkörnyezet
- Hello Scala!
- értékek, típusok és inferred típusok
- call-by-name vagy call-by-value paraméterátadás
- a tailrec annotáció és tesztek írása
- hosszabb feladat: numerikus integrálás
- hosszabb feladat: zérushelykeresés húrmódszerrel
- case classok: egy Vektor3D osztály, függvények kompozíciója
- a tailrec buktatói OOP környezetben és a private, protected, final és sealed kulcsszavak
- a lista matchelés
- hosszabb feladat: keresőfa implementálása
- listaműveletek I és II
- hosszabb feladat: Huffman-kódolás
- hosszabb feladat: hash halmaz implementálása

Funkcionális programozás a gyakorlatban

Iván Szabolcs – Számítástudomány Alapjai Tanszék,
SZTE TTIK

Jelen tananyag a Szegedi Tudományegyetemen
készült az Európai Unió támogatásával.
Projektazonosító: EFOP-3.4.3-16-2016-00014