Razvoj programskog jezika F#

Seminarski rad u okviru kursa
Dizajn programskih jezika
Matematički fakultet

Vladan Kovačević
mr17059@alas.matf.bg.ac.rs

20. decembar 2019.

Sažetak

U ovom radu opisan je programski jezik F# kao i programrski jezici koji su uticali na njegov razvoj: ML, OCaml, Erlang, Haskell, C# i Python. Kod ovih programskih jezika, osim kratke istorije i navođenja primena, opisani su bitni koncepti koji su u F#-u nasleđeni, dopunjeni ili uklonjeni. Dato je razvojno stablo koje sadrži F# i jezike koji su uticali na njega. Dati su primeri koda u F#-u i *Erlang*-u.

Sadržaj

1 Uvod

Jedan od osnovnih ciljeva .NET *Framework*-a, moćne *Microsoft*-ove platforme za razvoj softvera bila je kompatibilnost programskih jezika; što znači da program napisan u jednom jeziku može da komunicira sa programom napisanim u drugom jeziku, preko CLI-a[1]. Kako se C# i Visual Basic, dva osnovna jezika .NET-a ne razlikuju mnogo, javila se potreba za novim jezikom, koji će dopuniti .NET dodatnim funkcionalnostima i omogućiti korišćenje funkcionalne paradigme. Iz tih razloga 2005. godine .NET porodica jezika dobija treći veliki programski jezik: F#.[1] Iako je u osnovi F# jezik funkcionalne paradigme, on se smatra programskim jezikom više paradigmi (objektno-orijentisane, imperativne).

Slika 1: Logo programskog jezika F#

2 Osnovno o programskom jeziku F#

Tvorcem F# jezika smatra se Don Sajm (*Don Syme*) koji je ovaj jezik razvio 2002. godine, iako je prva stabilna verzija objavljena 2005. godine. F# je ravijen u Kembridžu, u okviru *Microsoft Research*-a. Po rečima tvorca, F u F# znači *fun* (zabava).
Kako F# potiče iz ML[2] i OCaml jezika, ovaj jezik je u početku imao sva svojstva ML jezika: funkcionalni jezik opšte namene, poseduje sakupljač otpadaka, statički je tipiziran, ima algebarske tipove podataka, rukovanje izuzecima i sl. Kasnije, kako se jezik razvijao, donekle se odvojio od ML porodice i neki ga ne smatraju njenim pripadnikom.[1]
Iako je u osnovni jezik funkcionalne paradigme, F# je i objektno-orijentisan (podržava apstrakciju koda u vidu klasa i objekata, kako bi kod bio čitljiviji), imperativan (podržava pristup memoriji, čitanje i pisanje fajlova, slanje podataka preko mreže i sl.), statički tipiziran (tipovi podataka su poznati pri kompilaciji što omogućava veću sigurnost koda). Kao .NET jezik ima sakupljač otpadaka i podržava mnoge .NET biblioteke kao i ostale .NET koncepte.[8] Iako se F# kompilira, često je korisno izvršavati ga kao skript jezik, pomoću FSI[3] koji omogućava dinamičko izvršavanje .fsx programa tako što vrši dinamičku kompilaciju.

[1] Common Language Infrastructure
[2] Meta Language
[3] F Sharp Interactive

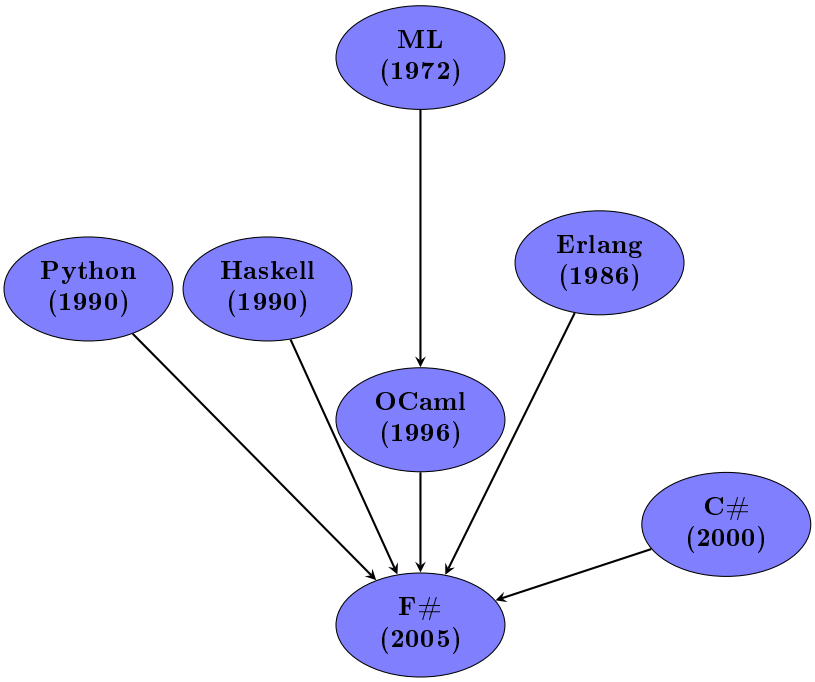
```
let factorial x =
    let rec tailRecursiveFactorial
     x acc =
        if x <= 1 then
            acc
        else
            tailRecursiveFactorial
     (x - 1) (acc * x)

    tailRecursiveFactorial x 1
```

Listing 1: F#

Slika 2: Rekurzivna funkcija za računanje faktorijela u jeziku F#

Funkcionalni jezici imaju matematičku osnovu i zasnovani su na lambda računu. Umesto promenljivih i stanja fokus je na konstantama i funkcijama. Programi se tretiraju kao funkcije; izlazna vrednost jedne funkcije je ulazna vrednost druge itd. Time su eliminisani sporedni efekti i izvršavanje zavisi isključivo od ulaznih vrednosti.
Iako F# jeste jezik opšte namene, njegova moć ogleda se u jednostavnosti, čitljivosti, sigurnosti i brzini izvršavanja. Osim što se koristi u naučnim istraživanjima i analizi podataka, zbog svoje moći paralelnog i interaktivnog izvršavanja koristi se u razvoju algoritama, finansijama, istraživanjima na polju veštačke inteligencije itd.

3 Razvojno stablo



Slika 3: Razvojno stablo jezika F#

U osnovi F# se razvio iz *OCaml*-a, mada na njegov razvoj značajno su uticali i C#, Python, Haskell i Erlang. Razvojno stablo jezika F# može se videti na slici 3.

3.1 ML

Kao što je opisano ranije, ML (*Meta Language*) je funkcionalni jezik opšte namene. Nastao je 1972. godine pod uticajem ISWIM jezika, a razvio ga je Robin Milner. Njegova prva namena bila je automatsko dokazivanje teorema. Njegovi naslednici su SML (*Standard ML*) i OCaml, o kome će biti više reči u narednom odeljku. Takođe je uticao na razvoj jezika Cyclone, Elm, Haskell, C++ itd.
Bitno je naglasiti da ML nije čist funkcionalni jezik, mogući su sporedni efekti pri izvršavanju. Osim ranije navedenih svojstava, ML podržava i parametarski polimorfizam. F# jeste jezik ML porodice i podržava navedene koncepte. ML se danas ne koristi u velikoj meri, a primenjuje se u finansijskim sistemima, bioinformatici, naučnim istraživanjima i sl.

3.2 OCaml

```
let rec factorial x =
    if (x = 1) then else x *
     factorial(x - 1)

let r = factorial 10
```

Listing 2: Ocaml

Slika 4: Rekurzivna funkcija za računanje faktorijela u jeziku OCaml

OCaml (*Objective Caml*) pripada ML porodici jezika. Prva implementacija Caml-a[4] pojavila se 1987. godine, a njegovi tvorci su *Ascánder Suárez*, *Pierre Weis* i *Michel Mauny*. 1990. godine *Xavier Leroy* i *Damien Doligez* razvili su novu implementaciju pod imenom Caml Light, sa bajtkod interpretatorom i brzim sakupljačem otpadaka. 1995. proširuju Caml Light u Caml Special Light, da bi konačno 1996. godine u INRIA u Francuskoj nastao OCaml, moćan, elegantan, sa dodatim konceptima objektno-orijentisane paradigme. 2000. godine *Jacques Garrigue* dopunio je ovaj jezik novim konceptima kao što su polimorfne metode, varijante i opcioni argumenti.
Kao jezik iz ML porodice OCaml jezik karakterišu sakupljač otpadaka, anonimne funkcije, statička tipiziranost, parametarski polimorfizam, prisustvo automatske tipske provere (pomoću ključne reči *auto*), ali pre svega sigurnost i elegantnost. Iako ovi koncepti nisu novi i prisutni su u ostalim jezicima, kod OCaml jezika su značajno razvijeni. Brzina se ostvaruje preko JIT[5] kompajlera, kao i preko preciznog i inkrementalnog sakupljača otpadaka.
U poslednjoj deceniji OCaml postaje sve popularniji, pa se javlja potreba da se dalje razvija i dopunjuje. Jedini problem jeste što uz njegov kompajler dolaze isključivo osnovne biblioteke, međutim, naprednije biblioteke koje dovode ovaj jezik do punog sjaja su raznovrsne i dostupne.[9]Koristi

[4]Categorical Abstract Machine Language
[5]Just In Time

se u akademske svrhe, za učenje programiranja, međutim koriste ga i kompanije kao što je Microsoft, IBM, CEA i druge. F# se direktno razvio iz ovog jezika i time je pored ostalih karakteristika jezika ML porodice preuzeo i sve dopune (kao što je objektna-orijentisanost) iz OCaml. Takođe imaju i veoma sličnu sintaksu.[6]

3.3 Erlang

Erlang je konkurentan, funkcionalan programski jezik nastao iz Prologa. Njegovi tvorci su *Joe Armstrong*, *Robert Virding* i *Mike Williams*, a razvila ga je kompanije Ericsson 1986. godine. Ime je dobio u čast danskog matematičara i inžinjera *Agner Krarup Erlang*-a, a slaže se i kao skraćenica za "Ericsson Language". Ima ogromnu primenu u telekomunikacijama, koriste ga mnoge kompanije kao što su Ericsson, Motorola, WhatsApp, Nortel, T-Mobile i druge. Svoje primenu u ovoj oblasti ima zahvaljujući distribuiranosti i skalabilnosti.
Dinamički je tipiziran i podaci su imutabilni. Dopunjuje ga OTP[6] i u okviru nje se razvija. Filozofiju ovog jezika opisao je njegov tvorac *Joe Armstrong* u svojoj doktorskoj tezi: sve je proces, procesi su strogo izolovani, stvaranje i prekidanje procesa su jeftine operacije, procesi komuniciraju isključivo slanjem poruka, procesi imaju jedinstvena imena, ako znamo ime procesa možemo mu poslati poruku, procesi ne dele resurse, rukovanje izuzecima nije lokalno, procesi obavljaju isključivo svoj zadatak ili se prekidaju. Ovakva orijentacija omogućila je Erlangu "Let it crash"pristup, gde se greške posmatraju kao sasvim uobičajene rantajm pojave, baš iz razloga jer je jeftino stvoriti i prekinuti proces. Takođe, procesi nadgledaju jedni druge.
Iako dosta razvijen i moćan, učenju Erlang jezika treba pristupiti ozbiljno, jer se u svojoj srži dosta razlikuje od ostalih programskih jezika. Erlang pristup programiranju može u značajnoj meri promeniti pogled na programiranje i na to kako bi programi trebalo da funkcionišu. Iz Erlanga, F# je pozajmio koncept obrade poruka koje šalju procesi.[5]

3.4 Haskell

Haskell je strogo tipiziran, čist funkcionalni jezik sa automatskom detekcijom tipova i lenjim izračunavanjem. Kako su početkom 80-ih godina prošlog veka funkcionalni jezici bili na ivici propasti, polovinom 80-ih godina javilo se interesovanje, a i potreba za stvaranjem novog lenjeg funkcionalnog jezika. Veliki tim stručnjaka je radio na tome i kao rezultat i pravo osveženje u funkcionalnom programiranju 1990. godine stvoren je Haskell kao naslednik Mirande. Ime je dobio u čast logičara *Haskell Curry*-a. Treba napomenuti da je ovo jedini lenji funkcionalni jezik koji se danas koristi.
Za razliku od prethodno navedenih jezika, Haskell je čist funkcionalni jezik zasnovan strogo na lambda računu, ne postoje sporedni efekti. Njegova lenjost ogleda se u tome što se funkcije ne pozivaju osim kad je to nužno, čime se izbegavaju suvišna izračunavanja. Ovaj koncept kombinuje se sa pamćenjem poziva funkcije, odnosno izraza.
Iako je Haskell, kao i ostali jezici funkcionalne paradigme bio isključivo korišćen u istraživačke svrhe, potražnja za funkcionalnim jezicima i implementacije osnovnih ideja u ostalim jezicima kao što su Python i Java, dovelo je do toga da Haskell izađe iz akademskih okvira i postane šire

[6] Open Telecom Platform

upotrebljen, kao jedan od najmoćnijih čistih funkcionalnih jezika. Haskell je na F# uticao kao i na bilo koji drugi jezik koji je implementirao ili se oslanja na funkcionalnu paradigmu.[7]

3.5 C#

C# je programski jezik opšte namene. Razvio ga je Microsoft 2000. godine i predstavlja jedan od najpopularnijih jezika današnjice. Mnogi (pogotovo tvorci Jave) ovaj jezik nazivaju "*Microsoft*-ovom Javom", tvrdeći da je ovaj jezik ništa drugo nego kopija Jave. Drugi smatraju da je C# ipak bliži jeziku C++.
Kao i Java, C# je jezik mnogih paradigmi: imperativne, objektno-orijentisane, generičke, funkcionalne, konkurentne itd. Za razliku od Jave, nema primitivne tipove; sve je objekat. Ipak, ne smatra se čisto objektno-orijentisanim jezikom zbog proceduralnih koncepata kao što su pokazivači na funkcije i sl. Funkcionalnost objekata izražena je preko klasa i interfejsa. Statički je tipiziran, izuzetno izražajan i robustan jezik. Kao i Java, prvo se prevodi u bajt kod zatim se pomoću interpretatora izvršava na različitim platformama, što omogućava izutetnu portabilnost programa.
Kao osnovni jezik .NET Framework-a o kojem je bilo reči ranije, podržava sve opisane koncepte i potkrepljen je mnoštvom korisnih biblioteka. Koristi se u razvoju internet i desktop aplikacija. I C# i F# je razvio Microsoft i oba jezika su deo njegovog .NET Framework-a. Veliki broj zajedničkih biblioteka dostupan je za oba jezika i programi pisani u jednom jeziku mogu da intereaguju sa programima pisanim u drugom i obrnuto.[4]

3.6 Python

Prema mnogim izvorima ovo je najpopularniji programski jezik današnjice. Zbog svoje fleksibilnosti i široke primene teško je reći kojoj paradigmi Python pripada.[3] Može se reći da je Python objektno-orijentisani skript jezik. Razvio ga je *Guido van Rossum* 1990. godine i dalje se razvija u okviru *Python Software Foundation*-a. Zahvaljujući *Python community*-u koji broji nekoliko miliona programera širom sveta ovaj jezik se svakodnevno razvija i unapređuje.
Python robustnost, statičko tipiziranje i sigurnost menja za fleksibilnost, dinamičke provere i lokalnu prilagodljivost, međutim, razvija se do te mere da prevazilazi navedene mane i ostvaruje neverovatnu brzinu i efikasnost.[3]
Python-ova sintaksa je izuzetno izražajna i jednostavna. Kao skript jezik omogućava interaktivno izvršavanje, pored toga svi podaci su dinamički tipizirani; ne postoji deklaracija promenljivih, omogućeni su pristup sistemskim funkcijama, manipulacija stringovima, ulazno-izlaznim podacima i sl.
Napisane .py skripte izvršava Python Shell. Mnogobrojne Python biblioteke dodatno proširuju primene ovog jezika. Koristi se kao komandni jezik operativnih sistema, jezik za veb, jezik proširenja, jezik za procesiranje teksta, u veštačkoj inteligenciji, matematici, statistici itd. Po ugledu na Python, F# ostvaruje interaktivno izvršavanje pomoću FSI.[2]

4 Zaključak

Bitno je poznavati najuticajnije i najkorišćenije programske jezike kao što su C++, Java, Python i drugi. Međutim, korisno je i proučiti manje

popularne jezike, jer postoje oni koji unapređuju i implementiraju koncepte za rešavanje specifičnih problema. U manje poznatim jezicima kao što je F# nekad se mogu kriti jednostavna rešenja kompleksnih problema.

Literatura

[1] Dave Fancher. *The Book of F#*. No Starch Press, 2014.

[2] Python Software Foundation. Zvanična stranica programskog jezika python. https://www.python.org.

[3] Milena Vujošević Janičić. Programske paradigme. http://www.programskijezici.matf.bg.ac.rs/dpj/2019/predavanja/skriptogranicenja/skript_ogranicenja.pdf.

[4] Ben Albahari Joseph Albahari. *C# 7.0 in a Nutshell*. O'Reilly, 2018.

[5] Simon St. Laurent. *Introducing Erlang*. O'Reilly, 2017.

[6] OCam. Zvanična stranica programskog jezika ocaml. https://ocaml.org.

[7] Brian O'Sullivan. *Real World Haskell*. O'Reilly, 2009.

[8] Chris Smith. *Programming F#*. O'Reilly, 2010.

[9] Jason Hickey Yaron Minsky, Anil Madhavapeddy. *Real World OCaml*. O'Reilly, 2014.