**ŽELJKO TOPIĆ**

**THE JULIA PROGRAMSKI JEZIK**

**SADRŽAJ**

1.UVOD

2.OSNOVNI KONCEPTI JEZIKA

* Svojstva jezika
* Primjena programskog jezika Julia

3.JULIA I PYTHON

* Brzina
* Zajednica
* „Libraries“
* Dinamički i statički tipovi
* Indeksiranje niza
* Upravljanje paketima
* Zaključak

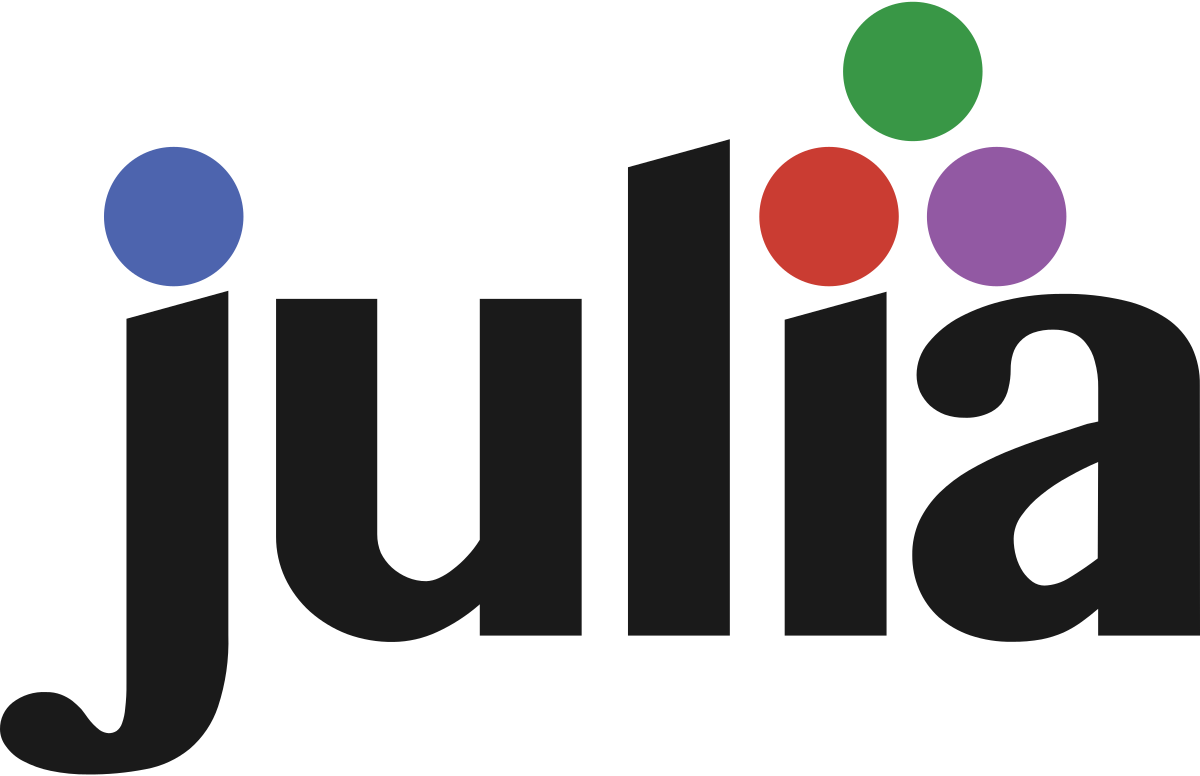
4. PRIMJERI KODOVA

* Komentari
* Tipovi i varijable
* Nizovi i stringovi
* Petlje i ulazi/izlazi
* Ostalo

5.LITERATURA

**UVOD**

Julia je dinamički programski jezik visoke razine i visokih performansi. Iako je jezik opće namjene i može se koristiti za pisanje bilo koje aplikacije, mnoge njegove značajke dobro su prilagođene za numeričku analizu visokih performansi. Performanse su usporedive s tradicionalnim jezicima statičkih tipova(C,C++,Fortran…). Juliu su dizajnirali i programirali: Jeff Bezanson, Alan Edelman, Stefan Karpinski i Viral B. Shah. Kada su stvarali ovaj jezik, željeli su napraviti jezik koji će biti brz kao C, dinamičan kao Ruby , da licence bude slobodna i besplatna te da ima matematičku notaciju kao Matlab. Julia je od početka dizajnirana da bude visokih performansi i da se “kompajlira” na više platformi.



**OSNOVNI KONCEPTI JEZIKA**

Julia je brz programski jezik, čak je usporediv sa C-om i C++ -om, koji su u programiranju standardi za brzinu. Julia je jezik koji je od početka dizajniran za visoke performanse i dinamičnost. Uz sve to ima bogat jezik opisnih tipova podataka, a deklaracije tipova mogu se koristiti za razjašnjenje i učvršćivanje programa. Julia ima sintaksu na visokoj razini, što je čini pristupačnim jezikom programerima iz bilo koje razine ili iskustva. Dostupna je pod licencom MIT, besplatna za upotrebu i sav izvorni kod je dostupan na GitHub-u (<https://github.com/JuliaLang/julia>). Editori koje možete koristi za Juliu su: Juno, Visual Studio Code, Jupyter, JetBrains, Vim, Emacs, SublimeText i NotePad++.

**Svojstva jezika**

Julia programski jezik je lagan i jednostavan za pisat bez velikih ograničenja. Za odlomke funkcija i petlji ne koriste se vitičaste zagrade već samo „end“ na kraju. To što se ne moraju pridjeljivati tipovi varijablama i tipovi funkcijama(void, int, char…) uvelike ubrazava pisanje koda, ali ostavlja veliku mogućnost za razne „bugove“. Julia ima dosta funkcija za rad s nizovima i stringovima, a za nekog tko je se do sada većinom pisao kodove u C -u je velika pomoć (primjeri se nalaze u poglavlju primjeri kodova). Kao što je već rečeno Julia je znanstveni jezik i jako je prilagođena takvom pisanju jezika. Korištenje for petlje unutar deklaracije niza, koja omogućava ubrzavanje pisanja koda jako nedostaje nekim ostalim jezicima tipa C i C++. Pisanje struktura u jeziku Julia je slično ostalim programskim jezicima. Julia za razliku od C++ nema podjelu na klase i strukture, već su sve strukture.

Konstruktori su slični onima u C++ , pristup vrijednostima strukture je pomoću točke, a ubacivanje vrijednosti u niz se izvodi funkcijom push!.

Julia je odličan jezik za korištenje skupova podataka i grafičkih podataka. Uz malo linija koda i pomoć interneta može se dosta naučiti i napraviti. Skupove podataka možete koristiti direktno s interneta što je odlično kada u svakom trenutku morate imati osvježene podatke, možete ih korisiti skinute ili napisati svoje. Pravljenje svojih skupova podataka je poprilično lako i lako se upravlja tim podacima. Kod korištenja grafičkih podataka, nacrta i slično Julia ima neograničene mogućnosti i što je i glavna specifikacija jezika koji se najviše koristi u znanstvene svrhe. Kod ovakvih tipova zadataka Julia dozvoljava mašti da radi što hoće, naročito početnicima u programiranju.

**Primjena programskog jezika Julia**

Programski jezik Julia se najčešće koristi za znanstveno računanje i matematičko programiranje. Julia pruža podršku za analizu velikih podataka obavljajući složene zadatke poput paralelizma i računanja u oblaku koji igraju temeljnu ulogu u analizi velikih količina podataka. Napravljena je s namjerom da se koristi u statistici i strojnom učenju. Julia se posebno iskazala u lineranoj algebri, gdje je pokazala puno bolje rezultate od Pythona.   
Julina sintaksa za matematičke operacije više liči na način na koji se matematičke formule pišu izvan svijeta računarstva, što neprogramerima olakšava posao. Zato većina programera koji koriste Juliu dolaze iz znanstvenog i matematičkog okruženja. Juliu također koristi BlackRock inženjering i britanska osiguravajuća kuća Aviva za procjenu rizika.   
U 2015., Federalna banka New Yorka koristila je Juliu za izradu modela ekonomije Sjedinjenih Država, primjećujući da je određeni model napravio procjenu modela "oko 10 puta brže" od njegove prethodne primjene MATLAB-a.

**PYTHON VS JULIA**

Citat tvoraca Julie:

“*Pohlepni smo: želimo još. Želimo jezik otvorenog koda, s liberalnom licencom. Želimo brzinu C s dinamikmom Ruby. Želimo jezik koji je homoiconski, s istinskim makronaredbama poput Lispa, ali s očitim, poznatim matematičkim zapisima poput Matlaba. Želimo nešto što je korisno za opće programiranje kao Python, jednako lako za statistiku kao R, prirodno za obradu niza kao Perl, snažno za linearnu algebru kao Matlab, dobro za lijepljenje programa zajedno kao ljuske. Nešto što je jednostavno za naučiti, ali najozbiljnije hakere usrećuje. Želimo interaktivan i želimo* ga *sastaviti.”*

Slika na kojoj se prikazuje računalo, tipkovnica

Opis je automatski generiran

**Brzina**

Julini tvorci željeli su napraviti jezik brz kao C. Iako je Python posljednjih godina lakše ubrzati, njegova je izvedba još uvijek daleko od onoga što Julia može učiniti. U 2017. godini Julia se čak pridružila Petaflop klubu - malom klubu jezika koji pri najvišim performansama može prijeći brzinu od jednog petaflopa u sekundi. Osim Julia, u klubu su trenutno samo C, C ++ i Fortran.

**Zajednica**

Sa više od 30 godina Python ima ogromnu zajednicu koja pruža podršku. Teško da postoji Python-ovo pitanje na koje ne možete dobiti odgovor u okviru jedne Google pretrage. Suprotno tome, zajednica Julia prilično je mala. Ali iako je mala na većinu pitanja se također mogu pronaći odgovori naročito na stranici Stack overflow.

**“Libraries”**

Ovo je jedna od najjačih točaka Pythona - njegovi milijuni dobro održavanih knjižnica. Julia nema mnogo knjižnica, a korisnici su se žalili da još uvijek nisu vrlo dobro održavane. Ali kad uzmete u obzir da je Julia vrlo mlad jezik s ograničenom količinom resursa, broj knjižnica koje već posjeduje prilično je impresivan. Osim činjenice da broj knjižnica raste, može se povezivati ​​i s knjižnicama iz C i Fortran.

**Dinamički i statički tipovi**

Python se 100% dinamički izvodi. To znači da program, u vrijeme izvođenja, odlučuje je li varijabla float ili cijeli broj. Iako je ovo iznimno prilagođeno početnicima, ono uvodi i čitav niz mogućih bugova. Python kod se treba testirati u svim mogućim scenarijima - što je prilično glup zadatak koji vam oduzima puno vremena. Budući da su tvorci Julie također željeli da to bude lako naučiti, Julia u potpunosti podržava dinamičko programiranje. Ali za razliku od Pythona, možete uvesti statičke tipove ako želite. Gdje dolazi do značajne uštede vremena.

**Indekisranje nizova**

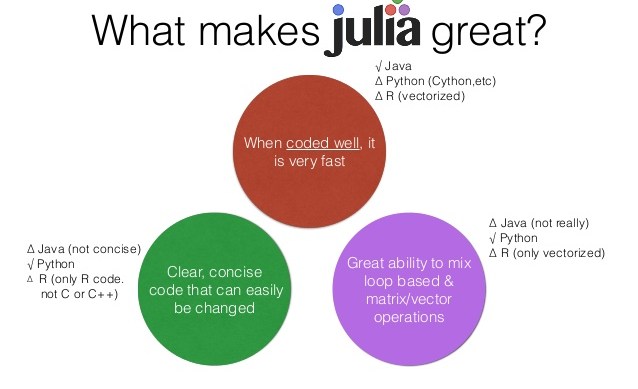
Julia nizovi su 1-indeksirani, tj. nizovi počinju s jedinicom, a ne nulom. To može stvoriti problem programerima koji imaju običaj koristiti druge jezike. Pytonovi nizovi su indeksirani s 0. Općenito, svaki jezik ima 0-indeksiranje za nizove.

**Upravljanje paketima**

Pkg dolazi s vlastitim REPL(Read–eval–print loop) i Julia paketom iz kojeg možete graditi, dodavati, uklanjati i primjenjivati ​​pakete. To je posebno prikladno zbog povezanosti Pkg-a i Gita. Ažuriranje je jednostavno, dodavanje paketa uvijek je jednostavno.

**Zaključak**

Python ima bolje pakete, a s tim obično ako je projekt dovoljno malen, većinom će biti korišten Python, ali za skupove podataka s milijunima promatranja, teško je čak i dobiti takve podatke pročitane u Pythonu. Naravno da Julia ima još neke sitne mane, ali s obzirom na njene resurse i da je ovo tek prvo izdanje(1.0) nadmašila je sva očekivanja.



**PRIMJERI KODOVA**

**Komentari**

# komentar u jednom redu

#= komenatri

u

više

redova =#

**Tipovi i varijable**

Julia koristi dinamičke tipove za pisanje varijabilnih vrijednosti, ali također prihvaća definicije statičkih tipova za pružanje prednosti ukazivanja da su određene vrijednosti određene vrste. Tipovi su slični onima u C-u i C++-u dok je pridjeljivanje malo drugačije.

mutable struct Person

name::AbstractString

male::Bool

age::Float64

children::Int

end

p **=** Person("Julia", false, 4, 0)

people = Person[]

push!(people,Person(“Sofia”, false, 30, 2))

printsum(people)

x = 10 , y = “hello”

#typeof() funkcija vraća tip

println(typeof(x))

global b = k # globalna varijbla

**Nizovi i stringovi**

Nizovi su kao i u C jeziku, s tim da je uz funkciju *push!()* jednostavno ubacivati u nizove i notacija počinje s jedinicom,a ne s nulom kao u većini ostalih jezika. Stringovi su znakovi unutar navodnika kao i u većini jezika. Julia dozvoljava izvlačenje elemenata stringa da bi kreirali podstringove.

# niz int tipa

a = [ 1, 2, 3, 4]

push!(a, 5)

push!(a, 5.4) #error

push!!(a, 5.2) # na ovaj način se ubaciva float tip s tim da ce „output“ izgledati 1.0

# 2.0 3.0 4.0 5.2

#napraviti 2x2x2 niz bilo kojeg tipa

C = ([1 “haha“ ; 3 4.45],[5 6; “zaz“ 8], dims = 3)

C = push!(C, [1 “nn“]) #dodavanje ćelija

C .+ 3 # dodati svim elementima 3

C = [ 2^1 for i = 1:6]

printsum(C)

#output će biti : [2,4,8,16,32,64 ]

#= STRING =#

#stringovi se mogu stvoriti s duplim ili troduplim navodnicima

str = “ Romeo,Romeo zašto si Romeo“

println(str)

str = “ “ “ geek

for

life“ “ “

println(“ \n Creating a multiline string: “)

println(str)

#printanje između četvrtog i trećeg od pozadi

println(str[4 : end -2]) #puno jednostavnije nego u C-u

neke funkcije : split(za pretvaranje stringa u niz), length(veličina), isequal(==), findfirst(), replace(), strip() …

Dodatne manipulacije stringovima: <https://juliabyexample.helpmanual.io/#Strings-Basics> .

**Petlje i ulazi / izlazi**

Osnovna sintaksa za čitanje i pisanje datoteka u Julia programskom jeziku je slična kao i u Pythonu. Petlje kao što su for i while imaju malo drugačiju sintaksu od C-a, iste su kao u Pythonu i dosta su jednostavnije.

fname = “ simple.dat“

open(fname, “r“) do f

for line in eachline(f)

println(line)

end

end

f = fopen( fname, “r“)

fstring = read (f, String)

close (f)

print( fstring)

#= PETLJE =#

for i in 1 : 5 # osnovni primjer for petlje

print( i, “ ,“)

end

println() # 1,2,3,4

while i < 10 # while petlja

i = i +1

end

**Ostalo**

If / else izrazi su više-manje kao i u ostalim jezicima. Uključivanje paketa : pkg.instaled() , pkg.add(“kk“), using kk, import kk. Data frames paketi omogućavaju alate za rad s tabularnim podacima. „Plotting“ u Juliji je moguć samo s dodatnim paketima (Plots.jl). Julia koristi riječnike kao asocijativne kolekcije.

**Literatura**

* <https://julialang.org/>
* <https://docs.julialang.org/en/v1/>
* <https://en.wikipedia.org/wiki/Julia_(programming_language)>
* <https://towardsdatascience.com/the-serious-downsides-to-the-julia-language-in-1-0-3-e295bc4b4755>
* <https://www.infoworld.com/article/3241107/julia-vs-python-which-is-best-for-data-science.html>
* <https://towardsdatascience.com/5-ways-julia-is-better-than-python-334cc66d64ae>
* <https://www.geeksforgeeks.org/julia-vs-python/>
* <https://syl1.gitbook.io/julia-language-a-concise-tutorial/useful-packages/dataframes>