Razvoj programskog jezika Rust

Seminarski rad u okviru kursa
Dizajn programskih jezika
Matematički fakultet

Milica Kleut i Aleksandar Petrović

20. decembar 2019

Sažetak

Tema ovog rada jeste programski jezik Rust. Cilj je zainteresovati čitaoca da nauči još jedan programski jezik nastao skorijeg datuma, koji se pokazao kao jako poželjan i udoban za programiranje. Kako su na nastanak Rusta uticali drugi programski jezici moramo se dotaći i njih, tako da ćemo dati njihov kratak pregled i ono čime su oni najviše uticali na Rust.

Verovatno je najlakše učiti programski jezik čija je sintaksa slična sintaksi nekog čitaocu već poznatog jezika. S obzirom na to da Rust-ova sintaksa podseće na C-ovsku, poznavaocima C-a ne bi trebalo da predstavlja problem da nauče sintaksu Rusta. Pored C-a glavni prethodnici su: C++, Cyclone, Haskell, ML, SALS i OCaml. Kako bi na najbolji način bili predstavljeni odnosi Rusta i njegovih prethodnika u nastavku će biti dato razvojno stablo.

Kako bi ste bolje razumeli ovaj jezik na početku ćemo vas uvesti u Rust a onda dati razvojno stablo. U drugoj fazi rada ćemo za svaki od prethodnika iz stabla dati ponaosob njegove karakteristike i kratak istorijski pregled. Ovakav redosled pisanja rada je isplaniran u cilju da čitalac proba da poveže ono što je na početku naučio o Rustu sa onim što će u nastavku čitati o drugim jezicima i da pokuša da pravi paralelu.

Interesantno je da je ovaj jezik za kratko vremen pokupio mnoge simpatije pa je danas popularan u krugovima programera. U odnosu na prethodnike uvodi neke novitiete i tako poboljsava i olaksava koncept programiranja.

Naravno, kako sve ima svoje prednosti i mane potudićemo se da tokom rada uočimo šta je to zbog čega je Rust korak ispred u odnosu na druge jezike a sa druge strane zbog čega su neki jezici ispred njega. Ovde smo došli do osetljive teme - "Koji jezik je bolji od kog, a koji jezik je najbolji?". U ovakvom razmatranju je jedino jedna stvar bitna a to je koji je cilj programera i kakva paradigma mu je potrebna da bi svoj problem rešio na najbolji način. U tom smislu čemo razmatrati koji je jezik "najbolji"i u kojim situacijama treba izabrati baš Rust.

Na kraju imamo mali osvrt na rad i rezime onoga sto je najbitnije.

Sadržaj

1 Uvod

Jedan od najvažnijih, a ujedno i najtežih zadataka svakog programera, bio on početnik ili iskusan programer, je odlučiti se za jedan programski jezik koji će koristiti. Postavljaju se razna pitanja među kojima su glavna: Koji jezik nam omogućuje dovoljno funkcionalnosti, a da je ujedno i pregledan, da ima lepu sintaksu? Koji će nam omogućiti da sa što manje muke napravimo moćan program? Puno je kandidata, puno konkurenata, a među njima nailazimo na programski jezik *Rust*. Jedan je od omiljenih programskih jezika danas. Čak se i postavlja pitanje, da li je on najbolji programski jezik koji postoji? Jedne od njegovih glavnih prednosti su njegove sjajne performanse, visok nivo kontrole i to što je memory safe. Logo jezika Rust prikazan je na slici 1.

Slika 1: Logo programskog jezika Rust

2 O Rustu

Rust je sistemski programski jezik nastao 2010. godine od strane Graydon Hoare-a u Mozilla Research-u. Njemu su na ovom projektu pomagali Dave Herman, Brendan Eich i ostali. Graydon Hoare, *"a language engineer by trade"*, kako sam sebe naziva je zapoceo razvoj programskog jezika Rust još u 2006. godini. Mozila se zainteresovala za ovaj jezik pa su započeli razvoj ovog jezika i krenuli sa eksperimentom "*Servo Parallel Browser Project*".

Rust je postao omiljen programski jezik na sajtu Stack Overflow Developer Survey(slika 2), i omiljen je svake godine od 2016.

Rust je multi paradigmatski jezik fokusiran na sigurnost (posebno sigurnu konkurentnost) za pisanje aplikacija visokih performansi, koje su obično pisane u C-u ili C++-u, ali su nailazile na problem sa memorijskim pristupima, koji bi često izazivali segmentation fault. Iako sintaksa samog jezika Rust jako podseća na sintaksu programskog jezika C postoje znacajne razlike:

-Pattern matching i algebarski tipovi podataka(enumi)

-Task-based konkurentnost. Lakši taskovi se mogu pokrenuti paralelno bez deljenja memorije

-Funkcije visokih naredbi (closures)

-Polimorfizam, kombinuje interfejs sličan Javinom i tipove klase kao u Haskell-u

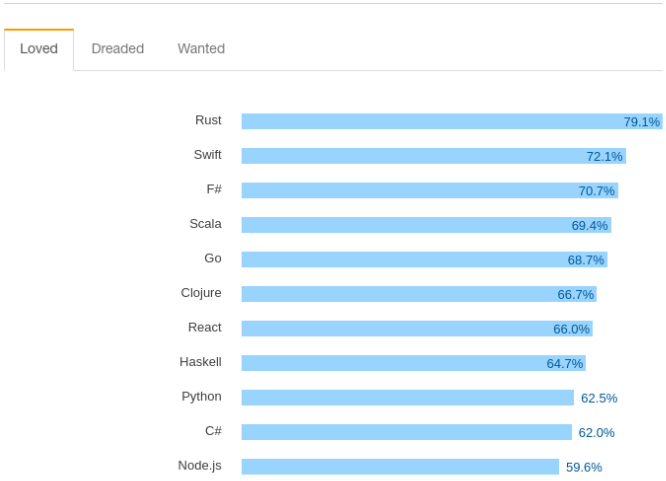
-Generici

-Nema buffer overflow-a

-Nepromenljiv

-Sakupljač otpadaka bez blokada

Svakako da na prvi pogled sintaksa jako podseća na C-ovsku ali u dubljem smislu se može reći da je bliža sintaksi ML i Haskell programskih



Slika 2: Tabela koja pokazuje listu najomiljenijih jezika, na sajtu Stack Overflow

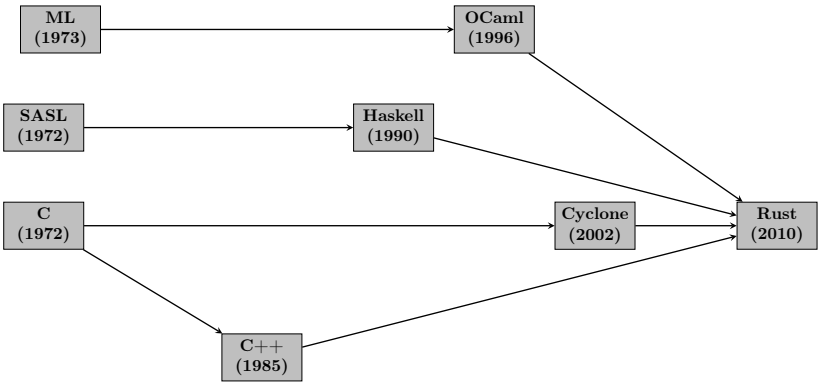
jezika. Ono što je još bitno napomenuti pored sigurnosti jeste da Rust posvećuje pažnju kontrolisanju deljenja memorije i konkurentnosti. Kada pričamo o upravljanju memorijom pravimo razliku u odnosu na jezike kao što su naprimer Go i Java. Rust ne koristi automatski sakupljač smeća već umesto toga upravlja memorijom i drugim resursima pomoću RAII (resource acquisition is initialization) konvencije sa fakultativnim brojanjem referenci. Takođe uvodi dodatnu sintaksu za upravljanje životnim vekovima promenljivih. Napravimo sada mini pregled Rusta i drugih programskih jezika. Upoređivanje performansi stvarnih aplikacija na više jezika je teško. Obično nemamo istu stručnost u više jezika, a na performanse utiču algoritmi i strukture podataka koje programer odluči da koristi. Ali se veruje da Rust deluje ravnopravno sa C++-om. Ima mnogo bolje rezultate od ostalih jezika zasnovanih na JIT-u, kao što su Lua ili Python. Razmotrimo sledeću tabelu(Slika 3) kako bi se uverili u to.

Language	Time (sec)	Memory (mb)
C++ Gcc	1.94	1.0
Rust	2.16	4.8
Java	4.03	513.8
LuaJIT	12.61	1.0
Lua 5.1	182.74	1.0
Python	314.79	4.9

Slika 3: Prikazana tabela

3 Razvojno stablo

Na slici ispod možemo videti kakav je odnos Rusta i drugih programskih jezika koji su sa njim usko povezani. Dakle najuticajni su: **ML (1973)** (kao što smo već pomenuli da je sintaksa Rusta bliska ML jeziku), **OCaml (1996)** (kao jezik koji je nastao iz ML-a bio je pogodan za jezik na koga se ugleda Rust), **SASL (1972) i Haskell(1990)**, **C (1972) i C++(1985)** (kao jezici koji su jako popularni i udobni za programiranje i čija je sintaksa jako čitljiva programerima), **Cyclone (2002)**. U nastavku ćemo dati jasnije veze između ovih jezika kao i osnovne informacije o svakom od njih. Razvojno stablo prikazano je na slici 4.



Slika 4: Razvojno stablo jezika Rust

3.1 C

Programksi jezik **C** je jedan od najpoznatijih programskih jezika, nastao krajem prošlog veka - tačnije 1972. Dizajnirao ga je američki informatičar Denis Riči (1941–2011) u SAD u Belovim telefonskim laboratorijama[1]. Značajan doprinos nastanku jezika su dali Ken Tompson (autor programskog jezika B) i Martin Ričards (autor programskog jezika BCPL). Ideja je bila da se napravi jezik koji će biti opšte namene. U početku je bio namenjem za pisanje sistemskog softvera ali je vremenom počeo da se koristi i za pisanje aplikativnog softvera.

Uticao je na mnoge druge jezike, na primer na **C++** a C++ i C su zajedno uticali na razvoj Rusta. C++ se smatra proširenjem jezika C, pa odatle jasno potiče i ime jezika. Glavna ideja je bila proširiti C tako da ima sposobnost objektno orijentisanog programiranja. Na ovom mestu je zanimljivo pomenuti i kako je zapravo C dobio ime. Kako svaki jezik ima nekog prethodnika ondosno nečiji je naslednik, tako je C naslednik jezika B pa je tako i dobio ime. C je prisutan na velikom broju platformi - od mikrokontrolera do superračunara.

Programi koji su pisani u njemu su jako bliski načinu rada hardvera pa je za programiranje u C-u naophodno dobro poznavanje rada procesora, memorije itd. Ima poprilično direktan pristup memoriji i lako se prevodi na mašinski jezik. Moderniji programski jezici imaju ugrađeno automatsko upravljanje memorijom (engl. garbage collection) što je neuporedivo lakse nego kad programer mora sam eksplicitno upravljati memorijom pa je to jedan od razloga što je danas većina novih korisničkih aplikacija

napisana u nekom modernijem jeziku. Danas se relativno retko ukazuje potreba za pisanjem novih korisničkih aplikacija u C-u.

Glavno područje njegove upotrebe su sistemski programi na strani servera, programi prevoditelji (engl. compilers) i jezgra operativnih sistema (engl. operating system kernels), gdje je potreba za najvećom mogućom brzinom izvođenja, efikasnom kontrolom resursa i kontrolom hardvera od primarne važnosti.

C možemo svrstati u proceduralne i imperativne programske jezike. Ima mali broj ključnih reči a mnoge funkcionalnosti su date kroz bibliotečke funkcije (na primer, ne postoje naredbe jezika kojima bi se učitavali podaci sa tastature računara ili ispisivali na ekran, već se ovi zadaci izvršavaju pozivanjem funkcija iz standardne biblioteke). Kod C-a se potencira prenosivost kôda (tzv. portabilnosti). Dakle kodovi napisani u C-u se mogu prevoditi na različitim platformama.

Programski jezik C se dosta menjao tokom godina te je u više navrata neformalno i formalno standardizovan. Pomenućemo neke od najvažnijih standardizacija.

Prva važnija verzija poznata je pod nazivom **"K&R C"**, što je engleska skraćenica prezimena dvaju autora najpoznatijeg C priručnika *"The C Programming Language"*, a to su Brian Kernighan i Dennis Ritchie. Prvo izdanje te knjige datira iz 1978. godine. Knjiga je napisana vrlo sažeto i precizno. Među programerima je poznata kao K&R ili kao white book. Godinama je služila kao neformalna specifikacija jezika C.

Drugo izdanje iz 1988. godine opisuje **"ANSI C"**, standard kojeg je 1983. godine definirao američki nacionalni institut za standardizaciju(ANSI) pod zvaničnim nazivom *ANSI X3.159-1989 „Programming Language C“* , a koji je i danas najbolje podržan. Ova verzija jezika C se obično jednostavnije naziva **ANSI C** ili **C89**. Ovaj jezik predstavlja nadskup K&R jezika C i uključuje mnoge konstrukte do tada nezvanično podržane od strane velikog broja prevodilaca.

Standard skraćeno poznat kao **C99** je usvojen 1999. godine. Uvodi male izmene u odnosu na prethodni standard koje su uglavnom inspirisane modernijim programskim jezicima kakav je C++.

Zatim 2011. godine pojavljuje se novi standard, objavljen kao *ISO-/IEC 9899:2011*. Skračeno ga zovemo **C11** tj. ranije poznat kao **C1X**. Uvodi unapređenu podršku za Unicode, generičke makroe i za anonimne strukture, precizno opisuje memorijski model za višenitna izračunavanja itd.

I za kraj poslednji standard je **C18**. Objavljen je 2018. godine kao *ISO/IEC 9899:2018*. Ovde su date tehničke ispravke i objašnjenja standarda C11 a inače ne uvodi nikakve novitete.

C je jako popularan jezik u krugu programera. Iako se danas ne koristi toliko, imao je veliki uticaj na ostale jezike pa ga zato često pominjemo i zato je pogodno znati ga.

3.2 C++

C++ je viši programski jezik nastao iz programskog jezika C[3]. Razvijen je u Belovim laboratorijama za objektno orijentisano programiranje u projektu pod rukovodstvom danskog naučnika [1]Bjarnea Stravsturpa. Kao što je pomenuto u prethodnom poglavlju C++ je nastao kao proširenje jezika C pa je njegovo prvobitno ime bilo „C sa klasama“ (engl. C with classes). Današnji naziv C++ je zvanično dat 1983. godine. Naziv je osmislio Rik Masciti. Bio je inspirisan time da ++ u programiranju oznacava inkrementaciju, tj. jedan korak više tj. poboljsanje, napredak pa kako je C++ upravo poboljsanje C-a dolazimo do današnjeg naziva C++. Prihvaćen je kao komercijalni proizvod 1985. godine. Tada jezik još nije zvanično standardizovan. Jezik je ažuriran 1989. godine tako da uključuje zaštićene i statičke članove.

Stravsturp je izabrao baš C za bazu jer je jezik opšte namene, brz je i široko rasprostranjen. C-u je dodavao funkcije slične onima koje ima Simula. Uticaj su imali i ALGOL 68, Ada, CLU i ML. Iz svakog jezika je uzimo korisne funkcionalnosti u cilju dobijanja objektno orijentisanog jezika koji uključuje četiri osnovna koncepta objektno-orijentisanog programiranja: enkapsulaciju, apstrakciju, nasleđivanje i polimorfizam. Uvodi neke novitete kao što su dodatne rezervisane reči i definisan logički tip podatka **bool** koji u C-u ne postoji[2].

Prvi kompajler C-a sa klasama nazvan je Cfront, koji je izveden iz C prevodioca zvanog CPre. Bio je osmišljen tako da se C++ kod prevede u običan C. Zanimljivo je da je Cfront napisan uglavnom C++-om, što ga čini *self-hosting* kompajlerom (kompajlerom koji može sam sebe kompajlirati). Cfront će kasnije biti napušten 1993. godine pošto je bilo teško u njega integrisati nove funkcije. Ipak, Cfront je izvršio ogroman uticaj na implementaciju budućih prevodilaca i na Unix operativni sistem. Tako je nastao jedan od danas najmoćnijih jezika ali i najkompleksnijih.

3.3 Haskell

Haskell[5] je čisto funkcionalni, statički tipiziran jezik. Nazvan je po Haskelu Bruks Kariju (engl. *Haskell Brooks Curry*), čiji rad u oblasti matematičke logike služi kao osnova za sve funkcionalne jezike. Programski jezik Haskell je zasnovan na lambda računu, i zbog toga se lambda koristi kao logo ovog jezika. Kod je kratak, jasan i lako održiv. Haskell nam nudi mali procenat grešaka i veliku pouzdanost. Pogodan je za pisanje velikih softverskih sistema zato što čini njihovo održavanje lakšim i jeftinijim.

Ako bismo tražili sličnosti Haskell-a i Rust-a tj. ako bismo tražili šta je Rust nasledio od Haskell-a prve dve stavke koje moramo uočiti su visoke performanse i konkurentnost. Naravno postoji još stvari ali su ove dve svakako ključne.

[1]Bjarne Stravsturp (*Bjarne Stroustrup*), rođen 11. juna 1950. godine u Orhusu, je danski informaticar i matematicar, koji je poznat po tome što je tvorac programskog jezika C++. Magistrirao je matematiku i informatičke nauke 1975. godine, na univerzitetu u Orhusu u Danskoj. Doktorirao je kompjuterske nauke 1979. godine na Kembridž univerzitetu u Engleskoj. Dizajnirao i implementirao programski jezik C++. Autor je više knjiga o programskom jeziku C++.

3.4 Cyclone

Cyclone je nastao kao projekat firme AT&T Labs i John Gregory "Greg"Morrisett-a u Kornelu(eng. *Cornell*) [4]. Kod ovog programskog jezika dosta podseća na kod programskog jezika C. Odlikuju ga pokazivači i pokazivačka aritmetika, strukture, nizovi, manuelno kontrolisanje memorije, goto, C-ovo pretprocesiranje i sintaksa. Dodatno Cyclone može da se pohvali stvarima kao što su pattern matching, algebarski tipovi podataka, izuzeci, region-based kontrolisanje memorije i opcioni sakupljač otpadaka.

Cyclone je bezbedan, ne dozvoljava greške koje postoje u C-u: buffer overflow, format string attacks, pristup pokazivača, itd. U njegovom razvoju su učestvovali Dan Grossman, Mike Hicks, Trevor Jim i drugi.. Međutim, istraživanja projekta su gotova i developeri su prešli na neke druge stvari. Nekoliko ideja je sa Cyclone-a prebačeno na Rust.

3.5 OCaml

OCalm (*Objective Caml*) je multiparadigmatski jezik koji se smatra najbitnijom implementaciojom programskog jezika Calm, koji je kreiran 1996. godine[8]. Njegovi tvorci su Xavier Leroy, Jérôme Vouillon, Damien Doligez, Didier Rémy, Ascánder Suárez i drugi. Calm spada u grupu ML jezika. Jezici izvedeni iz ML-a najpoznatiji su po sistemima statičkog tipa i kompajlerima koji zaključuju tip. Tipovi podataka promenljivih i potpisi funkcija ne moraju biti eksplicitno deklerisani, za razliku od nekih drugih jezika kako što su Java i C#.

Proširuje karakteristike objektno orijentsianog programiranja. Zaključujemo da pripada objektno orijantisanoj paradigmi ali pored nje pripada i funkcionalnoj i imperetivnoj. Ima veliku standardnu biblioteku.

Poželjan je za softverski inženjering. Od drugih jezika se verovatno najviše razlikuje po svom naglašavanju performansi. Njegov statički sistem sprečava runtime tipove neusklađenosti i na taj način vrši sigurnosne provere koje inače opterečuju performanse dinamički tipiziranih jezika a istovremeno garantuje sigurnost izvršavanja, osim kada je isključena provera granica.

Ocaml lanac alata uključuje interpreter najvišeg nivoa, bajtkod kompajler, optimizacioni kompajler za izvorni kod, reverzibilni dibager i paket menadžer (OPAM). OPAM je razvijen od strane OCamlPro-a i jadnostavan je način za instaliranje OCaml-a i mnogih njegovih alata i biblioteka.

Neki od softvera napisanih u OCalm-u :

-0install, multi-platform paket menadžer

-Coccinelle, uslužni program za prevođenje izvornog koda C programa

-Coq, formalni sistem za upravljanje dokazima

-FFTW, biblioteka za računanje diskretnih Furijeovih transformacija

-Frama-C, služi za analiziranje C programa

-GeneWeb, besplatan i open-source multi-platformski softver

-Mldonkey, aplikacija za deljenje datoteka zasnovana na EDonkey mreži

-Tezos, samoizmenjiva platforma koja koristi XTZ kao domaću valutu

-Ocsigen, OCaml mrežni okvir

-Opa, besplatan i open-source programski jezik za razvoj veba

i još mnogo drugih....

Mnoge komnije koriste OCalm[7]. Neke od njih su Facebook, Docker, Bloomberg L.P., Citrix, Aesthetic Integration, Ahrefs, American Museum of Natural History...

3.6 ML

ML (eng. *MetaLanguage*) je programski jezik razvijen na Univerzitetu u Edinburgu 1970-ih godina[6]. Na njegovom razvijanju radio je Robin Milner, inspirisan sintaksom ISWIM-a. ISWIM je apstraktan programski jezik dizajniran od strane P.J.Landin; akronim predstavlja *"If You See What I Mean"*. Svrstavamo ga u grupu imperativnih i funkcionalnih paradigmi. U nastavku ćemo dati osnovne karakteristike jezika, navesti neke jezike iz ove familije, i videti koji su njegovi prethodnici i naslednici.

U osnovi ML-a jeste lambda račun koji je strogo tipiziran. U funkcijama se koritiiti poziv po vrednosti (енг. *call-by-value*). Dakle funkciji se prosleđuju vrednosti promenljivih ali one same ne mogu biti promenjene. Parametarski polimorfizam se u ML-u koristi da funkcije ili tipovi podataka mogu biti napisani tako da se može prenositi vrednost. To su generičke funkcije, odnosno tipovi. Zatim ima funkcije višeg reda - uzima jednu ili više funkcije kao argumente i vraća funkciju kao rezultat. U lambda računu su sve funkcije višeg reda. Koristi automatsko upravljanje memorijom, tj. poseduje sakupljač smeća (eng. *garbage collection*). Poseduje statički doseg za promenljive.

Glavni jezici iz ove familije su Caml i Standard ML (SML). Ima ih još, kao što su F#, koji se koristi kao podrška Microsoftovoj .NET platformi.

ML je uticao na Haskell, Cyclone, Nemerle, Rust. Na ML je najviše uticao programski jezik Lisp.

3.7 SASL

SALS je programski jezik koga je dizajnirao Dejvid Turner na Univerzitetu St Andrews 1972. godine a kasnije je radio na jeziku Miranda na Univerzitetu u Kentu. SALS je skracenica od *St Andrews Static Language* ili *St Andrews Standard Language*. Pripada strogo jednoj paradigni - funkcionalnoj paradigmi. Godine 1976. Turner ga je redizajnirao i dogradio kao nestriktan (lenji) jezik. SALS je bio temenj za kasnije Turnerove jezike - KRC i Miranda.

Kompanija Burroughs Corporation je koristila SALS za pisanje kompajlera i operativnog sistema.

SALS nije bio mnogo popularan programski jezik ali je bio dobar kao uzor pri dizajniranju novih jezika.

4 Zaključak

U ovom tekstu smo prikazali razvoj programskog jezika Rust. Takođe smo prikazali i jezike koji su najviše uticali na njegov nastanak i razvoj. Čitajući o njima možemo napraviti paralelu određenog jezika prethodnika i Rusta. Treba da pokušamo da uočimo sta im je to slično tj. šta je tvorac Rusta smatrao zančajnim i poželjnim pri pravljenju dobrog programskog jezika a šta je izbacio, ili probao da popravi. Na početku rada dato je razvojno stablo gde smo mogli da vidimo hronološki šta se dešavalo. Očekuje se da će se stablo od Rusta produžavati tj. da će Rust uticati na nove jezike. Cilj rada je bio zaintrigirati čitaoca i ohrabriti ga da se upusti u učenje novog, jako popularnog programskog jezika. Dodatno su u tekstu ubačene tabele koje pokazuju prednosti Rusta. Iz priloženog možete zaključiti i sami zbog čega odabrati baš Rust!

Za dodatne informacije i praćenje daljeg razvoja najbolje je koristiti zvaničnu stranu programskog jezika Rust [9].

Literatura

[1] C. PDF knjiga "Programiranje 1", Metematički fakultet u Beogradu, Beograd 2019. http://poincare.matf.bg.ac.rs/~janicic//books/p1.pdf.

[2] C++. Stranica o programskom jeziku C++. http://www.cplusplus.com/info/description/.

[3] C++. Zvanična stranica programskog jezika C++. http://www.cplusplus.org/.

[4] Cyclone. Zvanična stranica programskog jezika Cyclone. https://cyclone.thelanguage.org/.

[5] Haskell. Zvanična stranica programskog jezika Haskell. https://www.haskell.org/.

[6] ML. . https://github.com/SMLFamily/Successor-ML.

[7] OCaml. Spisak kompanija koje korite Ocalm, preuzeto sa zvanične stranice. https://ocaml.org/learn/companies.html.

[8] OCaml. Zvanična stranica programskog jezika OCaml. https://ocaml.org/.

[9] Rust. Zvanična stranica programskog jezika Rust. https://www.rust-lang.org/.