Perl - sjajno rešenje za analitičke probleme

Seminarski rad u okviru kursa
Metodologija stručnog i naučnog rada
Matematički fakultet

Aleksandar Vračarević, Jovan Marić,
Petar Mićić, Tatjana Radovanović
vracarevicaleksandar@gmail.com, maric993@hotmail.com,
micicpetar73@gmail.com, tatjana.tasa.95@gmail.com

30. april 2019

Sažetak

Cilj ovog rada je da čitaocu, kroz primere kôda i pregled glavnih namena programskog jezika Perl, izgradi prvi utisak o jeziku koji je zbog svoje jednostavnosti i čiste sintakse, ali i mnogih mehanizama koji su u njega ugrađeni, uticao na razvoj mnogih jezika. U ovom radu biće prikazano kako se Perl razvijao kroz istoriju, kao i zašto se mnogi programeri opredeljuju za njega.

Sadržaj

1 Uvod

Većina programskih jezika je osmišljena sa ciljem da reši određenu grupu problema, tako je i programski jezik Perl nastao da reši neke česte probleme pri razvoju softvera. Kao takav programski jezik Perl je stekao veliku popularnost i ima širok domen primene. Perl spada u jezike koji se lako uče jer imaju jednostavnu sintaksu nalik na prirodne jezike. Uz podršku velikog broja modula, Perl programi ostvaruju obimnu funkcionalnost uz minimalan skup naredbi. Iako su njegovu popularnost preoteli moderniji programski jezici kao što su C++, Java i Python [14], i dalje predstavlja ozbiljno oružje kod veb programiranja. U ovom radu je napravljen pregled osnovnih mogućnosti Perla, njegovih karakteristika i specifičnosti. Navedena su i popularna okruženja koja Perl koristi i uputstvo za instalaciju.

2 Nastanak i razvoj

Priču o Perlu nije moguće započeti bez njegovog stvaraoca Lerija Vola (eng. *Larry Wall*). Kako sam naglašava u tom trenutku postoje i druga rešenja, ali nijedno od njih ne rešava obradu teksta i generisanje izveštaja na lak i intuitivan način. Zbog prirode problema nameće se jezik koji će pripadati skript paradigmi. Želeo je da naziv bude kratka reč pozitivne konotacije. Opredeljuje se za pearl(srb. *biser*) ali izostavlja slovo a zbog mogućeg podudaranja sa drugim jezikom u izradi[10] .

Tabela 1: Pregled osnovnih doprinosa različitih verzija Perla

Verzija	Datum	Novine
2.0	05.06.1988	-unapređenje regex funkcija -rekurzivne subrutine -blokovske promenljive -foreach petlja -sort operator
3.0	18.10.1989	-podrška za rad u binarnom zapisu -prosleđivanje promenljivih referencom -nove sistemske funkcije
4.0	21.03.1991	-otklanjanje manjih bagova -objedinjena dokumentacija[1]
5.0	18.10.1994	-novi interpretator -objekti -nove funkcije
5.6.0	22.03.2000	-podrška za utf-8 -podrška za 64-bitni sistem -nadogradnja threadova

[1] U vidu knjige Programing Perl Randal Schwartz, Larry Wall, 1991

Perl 1.0 postaje dostupan 18. decembra 1987. godine korisnicima Usneta [2], u sekciji posvećenoj računarima. Jedna od ideja vodilja prilikom izrade ovog programa bilo je da, kao i bilo koji drugi jezik, raste i evoluira 1. Ovakav pristup se ogleda u brojnim izborima koje Vol donosi u fazi projektovanja. Na primer, moguće je dodati nove ključne reči u bilo kom trenutku bez narušavanja starih kôdova [16]. Ovakve odluke imaju posledice na performanse koje su za Vola prihvatljive jer on pre svega pokušava da olakša korišćenje programa njegovim korisnicima.

Početak dvadest i prvog veka obeležen je nastavkom evoluiranja Perla koji se prilagođava svim potrebama modernog programiranja. Podrška za mrežno programiranje kao i za implementaciju funkcionalne paradigme samo su neki od primera. Sve popularniji pretraživač DuckDuckGo većinski je implementiran u Perlu[4].

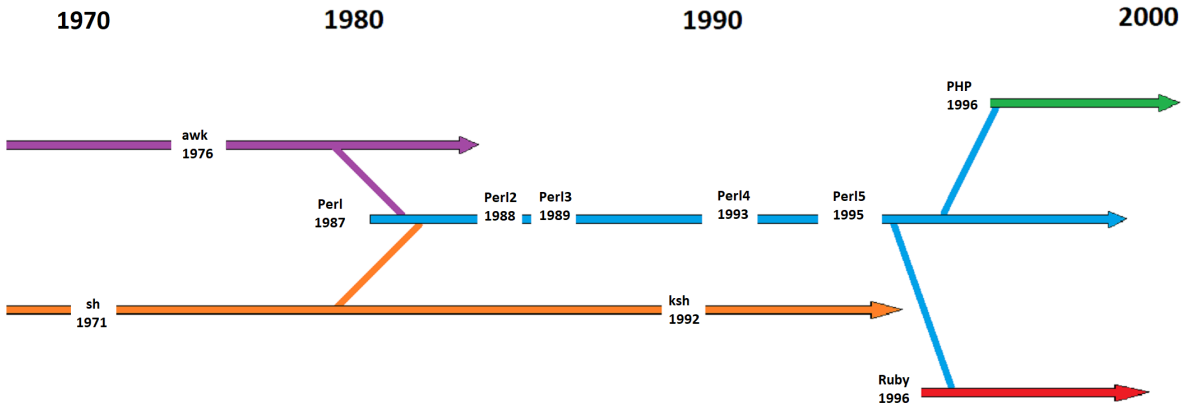
2.1 Mesto u razvojnom stablu i uticaji drugih programskih jezika

Uzimajući u obzir činjenicu šta je osnovna namena ovog programskog jezika nije iznenađujuće da se u Perlu može naći uticaj šel skripte(eng. Shell script), AWK, sed alata, kao i programskog jezika C. Kao što je poznato, C je glavni predstavnik proceduralne paradigme tog vremena[8].

Iako su počeci Perla mnogo prethodili internetu konstantnim evoluiranjem nikada nije prestao da bude relevantan. Verzija 5.004 donosi modul koji obezbeđuje API za pisanje veb aplikacija. Krajem devedesetih nazivan je lepkom koji održava internet. U razvojnom stablu programskih jezika zauzima mesto pretka PHP-a. Sintaksa jezika je slična. Praznine se ignorišu, blokovi su između vitičastih zagrada, naredbe se odvajaju tačka zarezom itd. Sve promenljive u PHP imaju prefiks '$' baš kao i skalarne promenljive u Perlu. U oba jezika evaluacija niske se vrši samo za niske sa dvostrukim navodnicima.

Sredinom poslednje decenije dvadesetog veka Yukihiro Matsumoto želi da objedini moć tekstualnog procesiranja Perla sa Python-om[13]. Godine 1993. objavljuje programski jezik Ruby čija sintaksa podseća na Perl. Sličnosti se još mogu videti i u gotovo identičnom radu sa niskama i regularnim izrazima.

[2] Preteča interneta na Unix sistemima za komunikaciju širom sveta



Slika 1: Drvo razvoja

3 Osnovna namena, svrhe i mogućnosti

Glavna snaga Perla ogleda se u radu sa regularnim izrazima. Postoje brojne modifikacije regex operatora koje olakšavaju korišćenje i pružaju veliki broj opcija programeru. Na usluzi programeru nalaze se i četiri promenljive u kojima se čuvaju rezultati poslednjeg uparivanja. Moguć je pristup sledećim informacijama: da li je došlo do uparivanja, uparenu nisku, nisku pre uparene i nisku posle[8]. Kada drugi programski jezici predstavljaju svoje mogućnosti vezane za parsiranje teksta i regularne izraze uglavnom to čine poredeći se upravo sa Perl-om.
Bezbednost je jedna od najznačajnih komponenti svakog programa. Od naglog razvoja mrežnog programiranja bezbednost eksponencijalno dobija na značaju. Već je pomenuto da se Perl pronalazi kao bekend(eng. *back end*) skripting jezik u mrežnom programiranju. Veliki razlog za to je tejnted mod rada(eng. *Tainted*). Kada Perl operiše u ovom režimu rada svakom podatku koji dolazi od korisnika ili iz okruženja pokretanja, dodaje se malo meta podataka koji ukazuje da ti podaci mogu potencijalno biti problematični. U svakom trenutku u programu se može proveriti poreklo podataka i izdvojiti siguran deo istog[11].
Kao odgovor na sve veću potražnju objektno orijentisanih jezika Perl nudi biblioteku Moose. Moose nudi osnovne mogućnosti kao što su pravljenje klasa, metoda, atributa ali i podržava koncepte kao sto su nasleđivanje i polimorfizam.

4 Osnovne osobine

Prvobitna zamisao autora Perla, Lerija Vola, bila je da napravi alat koji bi na brz i lak način omogućio napredno rukovanje proizvoljnim tekstualnim datotekama. Jezik je osmišljen sa idejom da se veći značaj prida njegovoj praktičnosti(lakoća korišćenja, efikasnost, kompletnost) nego lepoti(minimalnost, elegancija). Autor smatra da je implementacijom Perla uspeo da izdvoji i iskoristi najbolje funkcionalnosti jezika C, awk, sed i sh[15].
Sintaksa je usko povezana sa sintaksom C-a. Umesto da, kao većina Unix

alata, ograničava memoriju, Perl je sposoban da čitav fajl učita kao jednu nisku(ukoliko je dovoljno memorije raspoloživo). Pažljivo osmišljeni mehanizmi poklapanja obrazaca (eng. *pattern matching*) programeru pružaju veoma moćan alat za brzo čitanje velikih datoteka. Osim za obradu teksta, Perl je tokom godina postao odličan alat za obavljanje raznih sistemskih zadataka, veb programiranje, grafičko programiranje, pristup bazama podataka, mrežno programiranje i mnoge druge zadatke.

Bitno je još i naglasiti da je Perl interpretirani jezik što omogućava prenosivost jednom napisanih kôdova na veliki broj različitih platformi[3]. Za razliku od strogo (eng. *strictly*) interpretiranih jezika, Perl program se najpre prevodi u sintaksno drvo i nakon toga se izvršava (više reči o korišćenju Perl interpretatora u 9).

4.1 Tipovi podataka

Perl interpretator se brine o tipovima(što znači da je jezik slabo tipiziran, odnosno programer nije u obavezi da definiše tipove) i o upravljanju memorijom(uz pomoć brojanja referenci). Postoji tri tipa ugrađenih podataka: skalari, nizovi skalara i asocijativni nizovi skalara, poznati i kao heševi (eng. *hashes*). Obični nizovi su uređeni i indeksirani brojevima počevši od nule, dok su heševi neuređene kolekcije skalarnih vrednosti indeksirane dodeljenim niskama (ključevima). Skalarne vrednosti mogu biti brojevi, niske i reference uz mogućnost lake i transparentne konverzije između ove tri varijante[4]. Činjenica da Perl dinamički alocira memoriju omogućava pisanje programa otpornih na greške programera poput prekoračenja bafera ili pristupa nedozvoljenoj memoriji. Interpretacija operatora i vrednosti ponekad zavisi od okolnog konteksta. Postoje dva osnovna konteksta: kontekst liste i kontekst skalara[3].

Primer 4.1 *Naredni primer prikazuje da se ista promenljiva može tumačiti na dva različita načina u zavisnosti od konteksta.*

```
my @niz = ('Zdravo ', 'svete ', ':D!')
# @niz se tumaci u kontekstu liste:
# promenljiva @elementi sadrzi kopiju
    vrednosti promenljive @niz
my @elementi = @niz
# @niz se tumaci u kontekstu skalara:
# promenljva $broj_elemenata sadrzi broj
    elemenata promenljive @niz
my $broj_elemenata = @niz
```

5 Šta je ono što izdvaja Perl?

Kao i svaki programski jezik, Perl ima svoje važne karakteristike koje ga izdvajaju od drugih programskih jezika. Sledi prikaz osnovnih specifičnosti ovog programskog jezika.

[3]Kompletan spisak podržanih platformi može se naći na sledećoj adresi.

[4]Iako skalar ne može neposredno sadržati više od jedne vrednosti, moguće je dodeliti mu referencu na niz vrednosti.

5.1 CPAN

Velika Perl Arhiva[1] (eng. *Comprenhensive Perl Archive Network*) predstavlja online skladište Perl programa, dokumentacije i modula koji olakšavaju implementaciju i održavanje Perl projekata. Arhiva čuva oko 180 hiljada modula otvorenog kôda napisanih od preko 12 hiljada programera.

Glavni zadatak CPAN-a je da omogući programerima da pronađu module koji nisu uključeni u paket osnovnih modula. Paket osnovnih modula se dobija instalacijom Perl kompajlera. Modul pod imenom CPAN.pm omogućava lako preuzimanje i instaliranje drugih Perl modula.

5.2 Parsiranje tekstualnog sadržaja

Prve verzije Perl programskog jezika su u fokus stavljale obradu i manipulaciju tekstualnog sadržaja. Sama skraćenica PERL (Practical Extraction and Reporting Language) prikazuje osnovne namene jezika, a to su izvlačenje nekog sadržaja i generisanje izveštaja na osnovu tog sadržaja.

Naredni primer prikazuje kako se na jednostavan način može izračunati broj pojavljivanja svake reči u tekstu, pri čemu se za reč smatra niska karaktera koja se sastoji samo od malih i velikih slova abecede.

```
while (my $line = <$fh>) {
    chomp $line;
    foreach my $str (split /^[A-Za-z]+$/, $line
    ) {
        $count{$str}++;
     }
}
```

Dakle, prvo se prolazi kroz fajl ($fh predstavlja referencu na fajl koji je prethodno otvoren) liniju po liniju i uklanja se znak za novi red. Potom, za svaku reč (smeštenu u promenljivoj $str) koja odgovara navedenom regularnom izrazu, ažurira se strukturu u kojoj se čuva reč i broj izračunatih pojavljivanja. Rezultat rada navedenog dela kôda je smešten u promenljivoj $count [14].

5.3 Jednostavno korišćenje regularnih izraza

Važnu ulogu pri manipulaciji tekstualnog sadržaja igraju regularni izrazi. Regularni izrazi su šabloni koji mogu biti prepoznati u nizovima karaktera. Rezultat rada operacije prepoznavanja šablona jeste tačno ili netačno, odnosno da li niz karaktera odgovara datom šablonu ili ne. Perl ima ugrađene mehanizme koji čine rad sa regularnim izrazima lakšim, nalik na alate grep, sed i awk.[7]

Osnovni operatori primene regularnih izraza, koje Perl nudi, jesu $=\sim$ i $!\sim$. U izrazu $str $=\sim$ m/regex/ proverava se da li se data niska $str podudara sa regularnim izrazom regex. Regularni izrazi se navode između dve kose crte. U slučaju da je došlo do poklapanja, izraz vraća 1, a nula inače. Operator $!\sim$ radi suprotno, u slučaju podudaranja vraća 0, a 1 inače.

Postoje tri osnovne operacije sa regularnim izrazima:

- Provera podudaranja - m/regex/
- Zamena reči - s/regex/word/
- Translacija - tr/regex/word/

Zamena reči funkcioniše tako što se sva podudaranja regularnog izraza u niski zamene navedenom niskom. Na primer u izrazu 'Volim mačke!' =∼ s/mačke/pse, deo niske mačke će biti zamenjen sa pse, i novodobijena niska "Volim pse"predstavlja rezultat izraza.

Translacija je slična zameni reči, osim što ne koristi regularne izraze prilikom pretrage niske. Na primer, ako je $string = 'Volim macke i pse', rezultat operacije $string =∼ tr/e/o će biti 'Volim macko i pso'.

6 Primeri kôda

Sledeći kôd predstavlja primer ispisa Hello world programa. Komentari se pišu iza tarabe. Tekst se može ispisati tako što se navede iza navodnika ili koristeći funkciju qq. Perl programi se čuvaju u fajlovima sa ekstenzijom .pl.

```
#Komentar
print "Hello, world!\n";
print qq =Did you say "Hello"?\n=;
```

Naredni primer demonstrira program koji računa zbir dva broja. Imena promenljivih počinju znakom $. U komentarima je dato šta koja print naredba ispisuje.

```
$a = 5;
$b = 4;
print qq =a \= $a\n=; # a = 5
print qq =b \= $b\n=; # b = 4
$zbir = $a + $b;
print qq=a + b \= $zbir\n=; #a + b = 9
```

Nazivi nizova u Perlu su u formatu @naziv_niza. Niz se može inicijalizovati tako što se elemeti navedu u zagradama, razdvojeni zarezom. Ukoliko je potrebno ispisati sve elemete niza, to se može učiniti samo navođenjem imena niza. Određenom članu niza se pristupa tako što se njegov indeks navede u uglastim zagradama iza naziva niza. U ovom slučaju ime niza može počinjati i znakom @ i znakom $. Indeks poslednjeg člana niza se dobija tako što se ispred imena niza navede $#.

```
@dan = ("Danas", "je", "lep", "dan");
$broj_reci = @dan; #Broje elementa niza
print "broj reci u recenici \"@dan\" je
    $broj_reci.\n";
print "@dan[2], $dan[3]\n";
print "Indeks posednjeg elementa je $#dan\n";
```

Grananje u programu je mogucé izvršiti koristeći if-else naredbu [5]. Naredni program prvo učitava broj sa standardnog ulaza, a zatim proverava da li je pozitivan ili negativan. Uslov se navodi u zagradi, a if i else blok se navode između vitičastih zagrada.

```
$broj = <STDIN>;
if($broj >= 0){
    print "Broj je pozitivan\n";
}
else{
    print "Broj je negativan\n";
}
```

[5] Grananje se može izvršiti i korišćenjem unless naredbe koja je slična if naredbi, ali se izvršava ako je izraz netačan.

Mogu se zadati i argumente komandne linije. Argumentima komandne linije se pristupa isto kao i elementima niza, samo što se kao naziv niza navodi ARGV. Broj argumenata komandne linije se dobija tako što se indeks poslednjeg elementa uveća za jedan.

```
$br_arg = $#ARGV + 1;
print "Broj argumenata komandne linije je
    $br_arg: @ARGV\n";
```

U Perlu je moguće koristiti for i while petlju na sličan način kao i u programskom jeziku C. Sledećim kôdom je prikazan primer korišćenja while petlje. Program prihvata argumente sa ulaza i ispisuje ih na ekran. Ključna reč my označava da je domen promenljive samo taj blok. Promenljiva $in u while petlji nema uticaj na promenljivu sa istim nazivom van petlje.

```
$in = 5;
while(my $in = <>) {
    print $in; #Ispisuje element koji smo
    uneli
}
print $in; #Ispisuje broj 5
```

Funkcije započinju ključnom rečju sub iza koje sledi naziv funkcije. Telo funkcije se piše između vitičastih zagrada. Argumenti se navode u telu funkcije na sledeći način my(lista_argumenata) = @_; Poziv funkcije se vrši na standardni način.

```
sub obim_kvadrata {
    my($a) = @_;
    return 4*$a;
}
```

Primer 6.1 *Sledeći kôd predstavlja jednostavan način da zamene vrednosti dve promenljive.*

```
$aa = 3;
$bb = 2;
($aa, $bb) = ($bb, $aa);
print "$aa $bb\n";
```

7 Podržane paradigme

Izjava Lerija Vola da *"Perl nema nikakvu agendu, osim da bude maksimalno koristan maksimalnom broju ljudi."*[15] oslikava težnju Perla da bude što opštiji jezik. Ta tvrdnja je potkrepljena i time što su podržane proceduralna, funkcionalna i objektno orijentisana paradigma.

7.1 Proceduralna paradigma

Kao što je već pomenuto u prethodnim poglavljima 3 i 4, uticaj alata awk, sed i programskog jezika C je primetan u Perlu pa se proceduralni način pisanja programa prirodno nameće. Više primera se može naći u poglavlju 6.

7.2 Funkcionalna paradigma

Perl je jezik višeg reda (eng. *higher-order*) koji implementira koncept dinamičkog pravljenja funkcija i prosleđivanja istih drugim funkcijama. Dovoljno je moćan da λ račun izrazi direktno, u samom jeziku, bez potrebe za pisanjem posebnog programa za parsiranje i evaluaciju izraza[9, 5]. Funkcija sa parametrom x i telom B, $\lambda_x.B$ u Perlu se predstavlja kao `sub {my $x = shift; B}`. Primena funkcije P na funkciju Q u lambda računu je jednostavno (PQ), dok je ekvivalentan zapis u Perlu `$P->($Q)`.

U sledećem primeru se vidi kako se može napraviti funkcija koja primenjuje prosleđenu funkciju na neki niz element po element.

```
sub mapfunkcija{
        my (@rez_lista);
        my ($funkcija, @lista) = @_;
        for (my $i=0;$i<=$#list;$i++)
        {
                $rez_lista[$i] = $funkcija->(
    $lista[$i]);
        }
        return @rez_lista;
};
my @niz = (65, -42, -92);
my $mapme = sub {
        my $x = $_[0];
        return $x + 42;
};
my @rezultat = mapfun ($mapme, @niz);
```

Nakon izvršavanja ovog kôda, u promenljivoj `@rezultat` se nalaze brojevi 107, 0 i -50.

7.3 Objektno orijentisana paradigma

Iako Perl nije objektno orijentisan jezik, može instancirati i manipulisati objektima. Klase su predstavljene paketima, koji moraju posedovati `new` podrutinu da bi se mogli instancirati. Objekat neke klase se može predstaviti uz pomoć niza ili heša. Enkapsulacija se postiže izdvajanjem interfejsa u pakete. Mehanizam za postizanje polimorfizma je korišćenje ključne reči `bless`. Ova ključna reč običnu referencu na strukturu podataka označava kao pripadnika nekom paketu i time proširuje mogućnosti te strukture. Nasleđivanje se ostvaruje pomoću specijalnog niza `@ISA`. Ovaj niz čuva spisak imena modula koji se nasleđuju. U narednom primeru biće prikazani neki od ovih koncepata.[12]

```
package Student;
sub new{            # podrutina za
    instanciranje klase
     my ($ime, $indeks, $prosek) = @_;
     my $ref_stud = {
          "ime"     =>     $ime,
          "indeks" => $indeks,
          "prosek" => $prosek,
     };
     bless $ref_stud, 'Student'; # objekat se
    proglasava instancom klase Student
     return $ref_stud;
}
# doseg prethodne klase je ili do kraja
    datoteke, ili do sledece package kljucne
    reci
```

```
package DrugaKlasa; # kraj prethodne klase
sub new{ ... } # definisanje druge klase
$laza_lazic = Student::new('Laza Lazic',
    123456, 9.0);
```

8 Najpoznatija okruženja i njihove karakteristike

Perl je stekao veliku popularnost u ranim danima veba i često se opisuje kao lepak koji održava internet. Stoga će u ovom poglavlju biti opisana dva okruženja za razvoj veb aplikacija uz dodatak već pomenutog Moose okruženja namenjenog da olakša upotrebu Perla u objektno orijentisanom stilu.

8.1 Catalyst

Catalyst je jedno od najrasprostranjenijih Perl okruženja za razvoj veb aplikacija zasnovano na MVC[2] (eng. *Model View Controller*) arhitekturi. Ovo okruženje pruža nivo apstrakcije nad uobičajenim ciklusom zahteva i odgovora. Zahtevi omogućavaju lako gledanje argumenata nadolazećih upita, POST podataka, otpremanje datoteka i zaglavlja dok odgovori pružaju mogućnost postavljanja zaglavlja i vraćanja izlaza klijentu.

8.2 Dancer

Dancer je okvir za pisanje veb aplikacija veoma intuitivne i izražajne sintakse. Modelovan je po uzoru na Ruby radni okvir, Sinatra. Aplikacije se grade navođenjem HTTP glagola, URL-ova (putanja) i metoda za obradu saobraćaja ka tim konkretnim URL-ovima.

```
use Dancer2;
get '/' => sub {
  return 'Zdravo svete!';
};
start;
```

Program se sastoji od HTTP glagola GET nakon čega sledi koreni URL '/' i anonimne podrutine koja vraća nisku. Ukoliko bi se ovaj primer pokrenuo na računaru, niska 'Zdravo svete!' bi se odštampala kada se veb pregledač uputi na adresu http://localhost:3000

8.3 Moose

Iako Perl ima podršku za objektno orijentisanu paradigmu, Moose pokušava da pisanje OO programa učini još lakšim i da programerima pruži jednostavnu deklarativnu sintaksu koja bi eliminisala potrebu za pisanjem konstruktora, destruktora i pristupnih metoda. U pozadini se i dalje dešava nešto slično onome opisanom u 7.3, ali je programer zaštićen od tih detalja implementacije i pruža mu se veći stepen apstrakcije.

```
package Student;
use Moose;
has 'ime' => (
    is => 'rw',
```

```
    isa => 'Str',
);
has 'neki_atribut' => (...)
use Student;
my student = Student->new(
    ime => 'mika mikic',
    neki_atribut => 'neka vrednost', ...
);
```

Nakon navođenja linije `use Moose;`, paket postaje klasa. Linija `is => 'rw'` pravi pristupnu funkciju koja može da čita i upisuje vrednosti u atribut ime. Linijom `isa => 'Str'` naglašava se da atribut ime može da prima samo niske.

9 Instaliranje i pokretanje

Da bi se uspešno pokretao Perl skript potrebno je imati kompajler[6]. U ovom poglavlju biće objašnjeno njegovo instaliranje na Linux i Windows operativnim sistemima.

9.1 Instaliranje i pokretanje na Linux operativnom sistemu

Za instaliranje Perl kompajlera na Linux operativnom sistemu potrebno je u terminalu uneti komandu `sudo apt-get install perl`. Sistem ce zatražiti da se unese lozinka. Kada se instalira Perl kompajler, potrebno je u terminalu pokrenuti sledeću komandu `curl -L http://xrl.us/installperlnix | bash`. Kada sve bude gotovo, komandom `perl -v` se proverava koja je verzija Perla instalirana.

Pokretanje Perl skripta na Linux operativnom sistemu se radi na jednostavan način. Otvori se terminal, a zatim se unese komanda `perl <putanja_do_fajla>`. Ova komanda će pokrenuti izvršavanje skripta.

9.2 Instairanje i pokretanje na Windows operativnom sistemu

Pre instalacije na Windows operativnom sistemu poželjno je prethodno proveriti da nijedna verzija Perla već nije instalirana. Zatim je potrebno skinuti i instalirati Padre, the Perl IDE/editor. Strawberry Perl je deo instalacije, ali takođe se dobijaju i mnogi drugi korisni CPAN moduli. Nakon instalacije, možda će biti zahtevano da se restartuje računar. Potom bi u Start meni/All Programs trebalo pronaći folder sa Perl-om i u njemu kliknuti na Perl komandnu liniju. Komandom `cpan App::cpanminus` instaliraju se moduli iz CPAN-a. Provera koja je verzija Perla instalirana može se uraditi komandom `perl -v`, čime se dobija potvrda i da je instalacija uspešna.

Ukoliko je instaliran Padre, the Perl IDE/editor, komande za pokretanje programa se mogu zadati i iz editora. U meniju se izabere Run, a zatim Run Script, mada može da se pritisne taster F5 i skript će biti pokrenut. Skript je moguće pokrenuti i iz komandne linije na sledeći način `perl <putanja_do_fajla>`.

[6]Strogo govoreći, perl program nije ni kompajler ni interpretator. Naime, Perl program se najpre predstavi sintaksnim drvetom koje nakon toga izvršava perlov izvršni sistem [6]. Imajući ovu činjenicu u vidu, u nastavku rada će biti korišćen naziv kompajler.

10 Zaključak

Ovaj rad prikazuje osnovne osobine programskog jezika Perl, nastanak, istorijski značaj i njegov uticaj. Upoznaje čitaoca sa sintaksom, glavnim elementima, nekim od najčešćih okruženja koje on koristi kao i osnovnim paradigmama koje on podržava. Ono što čini ovaj jezik nezamenljivim oružjem kada je u pitanju tekstualna analiza, jeste ugrađen skup bogatih mehanizama za rad sa regularnim izrazima koji ubrzavaju i olakšavaju razvoj softvera. Oblasti u kojima je Perl pogodan izbor su takođe i mrežno programiranje, programiranje baza podataka, veb programiranje i grafičko programiranje.

Literatura

[1] Comprenhensive Perl Archive Network. on-line at: https://www.cpan.org/.

[2] MVC. on-line at: https://metacpan.org/pod/Catalyst::Manual::About#The-MVC-pattern.

[3] perldoc. https://perldoc.perl.org/perldata.html#Context. Accessed: 2019-04-03.

[4] Duckduckgo github repo. https://github.com/duckduckgo/duckduckgo. Accessed: 2019-04-03.

[5] Sultan Al-Qahtani, Pawel Pietrzynski, Luis Guzman, Rafik Arif, and Adrien Tevoedjre. Comparing Selected Criteria of Programming Languages Java, PHP, C++, Perl, Haskell, AspectJ, Ruby, COBOL, Bash Scripts and Scheme Revision 1.0. https://arxiv.org/abs/1008.3434. Accessed: 2019-04-03.

[6] Tom Christiansen, brian d foy, Larry Wall, and Jon Orwant. perlglossary. https://perldoc.perl.org/perlglossary.html. Accessed: 2019-04-05.

[7] Free Software Foundation. sed, a stream editor, 2000. on-line at: https://www.gnu.org/software/sed/manual/sed.html.

[8] Jeffrey EF Friedl. *Mastering regular expressions*. "O'Reilly Media, Inc.", 2006.

[9] Chuck Liang. Programming language concepts and perl. *J. Comput. Sci. Coll.*, 19(5):193–204, May 2004.

[10] Marjorie Richardson. Larry wall, the guru of perl.

[11] Michael Saltzman. *Modern Perl Programming*. "Onyx Neon Press", 2014.

[12] Sriram Srinivasan. *Advanced PERL Programming*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1st edition, 1997.

[13] Dave Thomas. Programming ruby–the pragmatic programmers' guide. 2001.

[14] Nathan Torkington Tom Christiansen. *Perl Cookbook*. O'Reilly, SAD, 2003.

[15] Larry Wall. Perl, the first postmodern programming language. In *Linux World Conference*, 1999.

[16] Tom Christiansen Wall, Larry and Jon Orwant. *Programming Perl, Third Edition*. "O'Reilly Media, Inc.", 2000.