POGLAVLJE 1

Uvod

Dobro došli u čuvenu knjigu koju u svetu zovu „the Llama book“ (knjiga sa lamom)!

Ovo je četvrto izdanje knjige u kojoj je od 1993. godine uživalo pola miliona čitalaca. Bar se nadam da je tako. Sigurno je da smo mi uživali pišući je.*

Pitanja i odgovori

Verovatno imate neka pitanja o Perlu, a možda i o ovoj knjizi, naročito ako ste je već prelistali da biste videli šta vas čeka. Zato ćemo ovo poglavlje iskoristiti da bismo odgovorili na ta pitanja.

Da li je ovo prava knjiga za vas?

Ako ste slični nama, verovatno već stojite u knjižari†, pitajući se da li da uzmete ovu knjigu sa lamom i naučite Perl ili možda neku drugu knjigu i naučite neki programski jezik koji je dobio ime po zmiji, piću ili nekom slovu abecede‡. Imate još samo dva minuta pre nego što će naići vlasnik knjižare i reći vam da niste u biblioteci§ i da bi trebalo nešto da kupite ili da izađete. Možda biste želeli da za ta dva minuta vidite neki jednostavan program napisan na Perlu da biste saznali koliko je on moćan i šta sve možete uraditi pomoću njega. U tom slučaju, pogledajte kratak vodič kroz Perl u narednom poglavlju.

* Prvu verziju ove knjige napisao je Rendal L. Švarc (Randal L. Schwartz), drugu verziju su napisali Rendal i Tom Kristijansen (Tom Christiansen), treću Rendal i Tom Finiks (Tom Phoenix), a četvrtu Rendal, Tom Finiks i Brajan Di Foj (brian d foy). Kad god u ovom izdanju kažemo „mi“, mislimo na poslednju grupu autora. Ako se pitate kako smo to *uživali* pišući ovu knjigu (prošlo vreme) kada smo i dalje na prvoj strani, odgovor je jednostavan: počeli smo da pišemo od kraja prema početku. Znamo da zvuči čudno, ali kada smo napisali indeks, sve ostalo je bilo lako.

† U stvari, ako ste slični nama, verovatno stojite u biblioteci, a ne u knjižari. Mi smo cicije.

‡ Pre nego što nam napišete poruku i obavestite nas da se radi o pozorišnoj trupi, a ne o zmiji, moramo vam objasniti da mislimo na tehnologiju CORBA.

§ Osim ako ipak *jeste*.

Zašto ima toliko fusnota?

Hvala što ste primetili. Ima mnogo fusnota u ovoj knjizi. Zanemarite ih. One su potrebne jer je Perl prepun izuzetaka od svojih pravila. To je dobro, jer je i život pun izuzetaka od pravila.

To znači da ne možemo napisati „operator fizbin služi za podešavanje kvazistatičkih promenljivih“, a da ne objasnimo izuzetak u fusnoti*. Prilično smo iskreni i zato moramo da pišemo fusnote. Vi možete biti iskreni i ne morate ih čitati. (Čudno je kako je sve to ispalo.)

Mnogi izuzeci imaju veze s prenosivošću. Perl je nastao na sistemima koji su radili pod Unixom i još uvek ima duboke korene u tom operativnom sistemu. Ipak, gde god je moguće, pokušali smo da pokažemo kako se programi mogu ponašati neočekivano, bez obzira na to da li je uzrok rad pod Unixom ili nešto drugo. Nadamo se da će čitaocima koji ne znaju ništa o Unixu ova knjiga poslužiti kao dobar uvod u programski jezik Perl. (Usput će besplatno naučiti i ponešto o Unixu.)

Većina drugih izuzetaka ima veze sa starim pravilom „80/20“. To znači da se 80% ponašanja programskog jezika Perl može opisati u 20% dokumentacije, dok preostalih 20% ponašanja zauzima 80% dokumentacije. Da bi ova knjiga bila mala, mi smo u glavnom tekstu opisali najčešća ponašanja, a u fusnotama smo dali veze do tekstova koji opisuju ostale mogućnosti (fusnote su napisane sitnim slovima tako da na istom prostoru možemo dati više informacija).† Kada pročitate knjigu bez obraćanja pažnje na fusnote, verovatno ćete hteti da još jednom pređete neke odeljke. U tom trenutku (ili ako tokom čitanja postanete neizdržljivo radoznali) možete pročitati i fusnote. Većina su ionako računarski vicevi.

Šta je s vežbama i njihovim rešenjima?

Vežbe se nalaze na kraju svakog poglavlja jer smo, među nama govoreći, ovaj isti materijal već koristili na predavanjima koja je slušalo nekoliko stotina studenata.‡ Vežbe smo pažljivo osmislili kako bismo vam omogućili da napravite i neke greške.

Nije da mi želimo da pravite greške, ali morate imati priliku. To je zato što ćete te greške sigurno praviti tokom programiranja u Perlu, pa je bolje da ih pravite sada. Sve greške koje budete napravili tokom čitanja ove knjige, nećete ponoviti kada budete pisali program za koji imate određen rok. Mi smo uvek tu da vam pomognemo ako nešto krene naopako; u dodatku A nalaze se rešenja svih vežbi i objašnjenja svih grešaka koje ste napravili i onih koje niste. Rešenja pogledajte tek kad uradite vežbe.

* Osim utorcima, kad nema struje, kad lakat držite pod čudnim uglom tokom ravnodnevice ili kad se izraz `use integer` koristi unutar petlje koju poziva prototip procedure u verzijama Perla starijim od verzije 5.6.

† Razmišljali smo da celu knjigu napišemo u fusnotama da bismo smanjili broj strana, ali se pisanje fusnota za fusnote otelo kontroli.

‡ Ne svi odjednom.

Ne gledajte rešenje pre nego što ste se dobro pomučili da uradite vežbu. Više ćete naučiti ako sami uradite vežbu nego ako pogledate rešenje. Ipak, nemojte udarati glavom o zid ako ne možete da rešite problem. Pređite na sledeće poglavlje i nemojte previše brinuti.

Čak i ako nikad ne pravite greške, ipak bi trebalo da pogledate rešenja vežbi. Priložena objašnjenja će istaći neke detalje programa koji isprva nisu bili očigledni.

Šta znače brojevi na početku vežbi?

Svaka vežba ima broj u uglastim zagradama ispred teksta, koji izgleda otprilike ovako:

> [2] Šta znači broj 2 u uglastim zagradama kada se pojavljuje ispred teksta vežbe?

Broj predstavlja našu (vrlo grubu) procenu minuta koje bi trebalo da potrošite na pojedine vežbe. Procena je gruba, pa se nemojte iznenaditi ako vežbu (uz pisanje, testiranje i otklanjanje grešaka) završite za dvostruko manje vremena ili ako je ne završite za dvostruko više. S druge strane, ako ste se baš „zaglavili“, nećemo reći nikome da ste provirili u dodatak A i pogledali rešenje.

Šta da radim ako sam predavač na kursu za programski jezik Perl?

Ukoliko ste predavač koji je odlučio da ovu knjigu koristi kao udžbenik (mnogo predavača je to radilo tokom godina), trebalo bi da znate kako smo se potrudili da svi skupovi vežbi budu toliko kratki da ih studenti mogu uraditi za 45 minuta do sat vremena i da im ostane malo vremena za pauzu. Vežbe iz nekih poglavlja mogu se uraditi brže, a za neke treba više vremena. To je zato što smo tokom pisanja brojeva u uglastim zagradama shvatili da ne znamo da sabiramo. (Srećom, znamo kako da nateramo računare da to rade umesto nas.)

Šta znači reč „Perl“?

Perl ponekad označava „praktičan jezik za zaključivanje i izveštavanje“ (engl. *Practical Extraction and Report Language*), mada ga neki nazivaju „patološki eklektični rasadnik ludosti“. To je retronim, a ne akronim, jer je Lari Vol (Larry Wall), tvorac Perla, prvo smislio izvorno ime, a tek zatim njegovu proširenu verziju. Nema smisla raspravljati o tome koje je ime ispravno; Lari priznaje oba.

U nekim knjigama ćete naići i na reč „perl“, napisanu malim početnim slovom. „Perl“ označava programski jezik, a „perl“ – interpreter koji prevodi i pokreće programe.

Zašto je Lari napravio Perl?

Lari je osmislio Perl sredinom osamdesetih godina prošlog veka kada je želeo da napravi neke izveštaje od datoteka iz Usenetovih diskusionih grupa, a to nije mogao da uradi pomoću Unixovog uslužnog programa *awk*. Pošto je lenj programer*, Lari je odlučio da problem reši pravljenjem alatke opšte namene, koju bi mogao da koristi bar na još jednom mestu. Rezultat je Perl u verziji nula.

Zašto Lari nije koristio neki drugi jezik?

Programskih jezika ima na pretek, zar ne? Ipak, u ono vreme Lari nije mogao da pronađe nijedan programski jezik koji bi odgovarao njegovim potrebama. Da je neki od današnjih programskih jezika tada bio dostupan, možda bi ga Lari koristio. Trebao mu je neki jezik koji omogućava brzo programiranje, poput uslužnog programa *awk*, i koji ima snagu naprednijih alatki kao što su *grep, cut, sort* i *sed*†, a da ne mora da se oslanja na jezike kao što je C.

Perl popunjava prazninu između programiranja niskog nivoa (omogućavaju ga jezici C, C++ ili asembler) i programiranja visokog nivoa (recimo, programiranje na komandnoj liniji). Programi niskog nivoa se po pravilu teško pišu i ružni su, ali su brzi i nemaju ograničenja; teško je nadmašiti brzinu dobro napisanog programa niskog nivoa. Pomoću programa niskog nivoa možete uraditi šta god poželite. S druge strane, programi visokog nivoa obično su spori, ružni i imaju ograničenja; postoji mnogo toga što ne možete uraditi ako za to ne postoji komanda u sistemu. Perl je jednostavan, skoro da nema ograničenja, uglavnom brz i pomalo ružan.

Pogledajte četiri tvrdnje o Perlu:

Prvo, Perl je jednostavan. Kao što ćete videti, to znači da se on lako koristi. Nije ga teško ni naučiti. Ukoliko ste vozač, verovatno ste proveli mnogo nedelja ili meseci učeći da vozite i sada vam je to lako. Kada budete potrošili isto toliko vremena na učenje Perla i on će vam biti jednostavan.‡

Perl skoro da nema ograničenja. Nema mnogo toga što ne možete uraditi pomoću Perla. U Perlu sigurno nećete pisati upravljački program na nivou mikrojezgra s podrškom za prekide (mada se i to može uraditi), ali većina uobičajenih problema – od malih i jednostavnih programa do velikih industrijskih aplikacija – dobri su zadaci za Perl.

* Ne vređamo Larija kada kažemo da je lenj; lenjost je vrlina. Radnička kolica je izmislio neko ko je bio previše lenj da sam nosi stvari; pisanje je izmislio neko ko je bio previše lenj da pamti. Perl je izmislio neko ko je bio previše lenj da obavi posao, a da ne izmisli potpuno nov programski jezik.

† Ne brinite ako ne znate šta znače. Važno je samo da su to programi koje je Lari koristio, ali nisu bili dorasli postavljenim zadacima.

‡ Nadamo se da ćete sa automobilom imati manje problema.

Perl je uglavnom brz. To je zato što ne postoji stručnjak za razvoj Perla koji ga ne koristi, pa svi želimo da on bude brz. Ako bi neko želeo da doda neku zgodnu mogućnost, koja bi usporila druge programe, Lari bi je skoro sigurno odbio dok se ne bi pronašao način da ona bude dovoljno brza.

Perl je pomalo ružan. To je tačno. Simbol izdavačke kuće O'Reilly za Perl je kamila, životinja na naslovnoj strani večne knjige sa kamilom (poznate i pod nazivom *Programming Perl*), rođake knjige sa lamom (i njene sestre, knjige sa alpakom). Kamile su takođe pomalo ružne. Ipak, one vredno rade, čak i u lošim uslovima. Kamile obavljaju posao uprkos svim teškoćama, čak i kada izgledaju loše i ponekad vas pljunu. Perl je pomalo sličan kamili.

Da li je Perl jednostavan ili komplikovan?

Perl se lako koristi, ali se ponekad teško uči. Ovo je, naravno, uopštavanje. Prilikom projektovanja Perla, Lari je morao da napravi brojne kompromise. Kada je imao priliku da napravi neku mogućnost tako da bude jednostavnija za programere, a da se teže savladava, on je skoro svaki put donosio odluke u korist programera. To je zato što ćete Perl naučiti samo jednom, a koristićete ga stalno.* Perl ima veliki broj pogodnosti koje programeru omogućavaju da uštedi vreme. Na primer, većina funkcija ima podrazumevane vrednosti; one često odgovaraju načinu na koji želite da koristite funkciju. Zato ćete često viđati sledeće delove koda u Perlu:†

```
while (<>) {
  chomp;
  print join("\t", (split /:/)[0, 2, 1, 5]), "\n";
}
```

Napisan u potpunosti, bez korišćenja podrazumevanih vrednosti i prečica, ovaj deo koda bio bi deset ili dvanaest puta duži, pa bi vam trebalo više vremena da ga pročitate i napišete. Isto tako, bilo bi ga teško održavati i pronalaziti greške, a imao bi i više promenljivih. Ukoliko znate malo Perla, a ne vidite promenljive u ovom delu koda, to je deo suštine. Sve one imaju podrazumevane vrednosti. Da biste – kao programer – imali ovakvu vrstu jednostavnosti, morate platiti cenu prilikom učenja; morate naučiti kako se koriste podrazumevane vrednosti i prečice.

Dobra analogija je pravilna i česta upotreba složenica napravljenih od više reči u engleskom jeziku. Izrazi „will not" (neću) i „won't" isto znače, ali većina koristi drugi izraz jer je kraći i svi ga znaju. Slično, složenice u Perlu su skraćenice za uobičajene „izraze" tako da se oni mogu brže napisati i održavati u jednom koraku.

* Ako ćete programski jezik koristiti samo nekoliko minuta tokom nedelje ili meseca, verovatno će vam više značiti neki koji se lakše uči, jer ćete skoro sve zaboraviti između jedne i druge primene. Perl je namenjen korisnicima koji programiraju najmanje dvadeset minuta dnevno i to uglavnom u Perlu.

† Nećemo ovde sve objašnjavati, ali ovaj primer čita podatke iz jedne ili više ulaznih datoteka i zapisuje ih u drugom formatu. Sve mogućnosti su objašnjene u ovoj knjizi.

Kada upoznate Perl, verovatno ćete manje vremena trošiti na pisanje deklaracija, a više na krstarenje Webom, jer je Perl odlična alatka za pronalaženje informacija. Perlove kratke konstrukcije omogućavaju pravljenje (s malo truda) zgodnih, malih programa ili alatki opšte namene. Napravljene alatke možete nositi sa sobom, jer je Perl vrlo prenosiv i široko dostupan, pa ćete imati još više vremena za krstarenje Webom.

Perl je jezik visokog nivoa. To znači da je kôd gust; program napisan na Perlu može biti dugačak od jedne do tri četvrtine odgovarajućeg programa napisanog na programskom jeziku C. Zato se programi u Perlu brže pišu, čitaju i održavaju, a ubrzano je i otklanjanje grešaka. Neće proći mnogo vremena, a vi ćete sagledati sve prednosti Perlovih potprograma, koji su dovoljno mali da staju na ekran, pa ne morate pomerati sadržaj da biste videli deo koji vam treba. Budući da je broj grešaka u programu proporcionalan dužini izvornog koda* (a ne funkcionalnosti programa), manji izvorni kôd u Perlu znači i prosečno manji broj grešaka.

Kao i na svim drugim jezicima, i na Perlu se mogu pisati programi koje je nemoguće čitati. Ipak, ako vodite računa, takve programe možete izbeći. Neiskusnim korisnicima programi u Perlu ponekad mogu izgledati besmisleno, ali će za iskusne programere oni predstavljati pravo blago. Ukoliko poštujete pravila iz ove knjige, vaši programi će se lako čitati i održavati, i verovatno neće pobediti na takmičenju za najnerazumljiviji kôd.

Kako je Perl postao toliko popularan?

Posle malo igranja s Perlom i dodavanja raznih mogućnosti, Lari ga je objavio u Usenetovim diskusionim grupama. Korisnici iz skoro celog sveta (njih desetine hiljada) slali su mu svoje komentare, pitajući ga kako da obave zadatke za koje on nije ni mislio da se mogu uraditi u Perlu.

Kao rezultat toga, Perl je nastavio da se razvija. Dobijao je nove mogućnosti, a postao je i prenosiv. Od malog jezika dostupnog na samo nekoliko sistema pod Unixom, Perl se razvio u moćan programski jezik, s hiljadama strana besplatne dokumentacije na Webu, desetinama knjiga, nekoliko velikih Usenetovih diskusionih grupa (i desetak nezavisnih, manjih), s brojnim čitaocima i primenama na skoro svim današnjim sistemima. Ne zaboravite ni ovu knjigu sa lamom.

Gde je Perl danas?

Lari danas više ne piše kôd, ali i dalje upravlja razvojem Perla i donosi sve važne odluke. Perl uglavnom održava grupa korisnika poznata pod imenom Perl 5 Porters. Rad ove grupe možete pratiti preko liste slanja na adresi *perl5-porters@perl.org*.

* Uz oštar skok kada neki odeljak programa prevaziđe veličinu ekrana.

U vreme kada pišemo ovu knjigu (mart 2005. godine), Perlu se spremaju velike izmene. U poslednjih nekoliko godina brojni stručnjaci rade na novoj verziji Perla: Perl 6.

Nemojte, ipak, odustajati od Perla 5, još uvek tekuće i stabilne verzije Perla. Ne očekujemo da će se uskoro pojaviti stabilna verzija Perla 6. Perl 5 neće nestati kada se pojavi Perl 6, a korisnici će verovatno nekoliko godina upotrebljavati obe verzije. Grupa Perl 5 Porters nastaviće da održava Perl 5, a dobre ideje iz Perla 6 naći će put do Perla 5.

Lari je 2000. godine prvi put predložio novu verziju Perla. U narednim godinama pojavio se novi interpreter pod imenom Parrot, ali to nije donelo mnogo promena za prosečne korisnike. Godine 2005, Otrijus Teng počeo je da se zabavlja jezikom Pugs (Perl User Golfing System) kao „lakom" implementacijom Perla 6 u Haskellu. Stručnjaci za razvoj s obe strane, odmah su pritrčali u pomoć. Ne možemo sa sigurnošću tvrditi šta će se dalje desiti, jer je novi jezik još uvek u radnoj fazi, ali možete pisati jednostavne Perl 6 programe u Pugsu. Detaljnije informacije o Perlu 6 potražite na adresama *http://dev.perl.org/perl6* i *http://www.ougscode.org*.

Za šta je Perl dobar?

Perl je dobar za brzo pisanje kratkih programa kada ste u žurbi. Perl je takođe dobar i za pisanje dugačkih i obimnih programa, za koje desetinama programera treba tri godine da ih završe. Naravno, vi ćete verovatno pisati brojne programe na koje ćete utrošiti manje od sat vremena, zajedno s planiranjem i testiranjem.

Perl je optimizovan za rešavanje problema koji u 90% slučajeva imaju veze s obradom teksta. Ovaj opis pokriva većinu programerskih zadataka koji se ovih dana pojavljuju. U savršenom svetu, svi programeri bi znali sve programske jezike, pa biste uvek mogli da izaberete najbolji jezik za svaki projekat. U većini slučajeva, izabraćete Perl.* Iako Tim Berners-Li nije ni razmišljao o Webu kada je Lari napravio Perl, to je bio brak sklopljen na Internetu. Neki tvrde da je primena Perla ranih devedesetih godina prošlog veka omogućila korisnicima da veliki deo sadržaja brzo prebace u format HTML, a Web ne bi postojao bez sadržaja. Naravno, Perl je odličan izbor za pisanje malih CGI skriptova (programa koji rade na Web serveru) i to toliko dobar da neiskusni korisnici često postavljaju pitanja: „Zar CGI nije isto što i Perl?" ili „Zašto biste za pisanje CGI skriptova koristili neki drugi jezik, osim Perla?". Mi mislimo da su ova pitanja zabavna.

Za šta Perl nije dobar?

Budući da je Perl dobar izbor za brojne primene, postoji li nešto za šta on nije dobar izbor? Perl ne bi trebalo da izaberete ako želite da napravite *neprovidan program*.

* Ipak, ne verujte nam na reč. Ako želite da saznate da li je Perl bolji od jezika X, naučite oba i vidite koji ćete češće koristiti. To je onaj koji je najbolji izbor za vas. Na kraju, učenjem jezika X bolje ćete razumeti Perl i obrnuto, tako da nećete uzalud potrošiti vreme.

To je program koji ćete nekome pokloniti ili prodati i u čijem se izvornom kodu ne mogu videti vaši tajni algoritmi, pa vam korisnici ne mogu pomoći da ga održavate i otkrivate greške. Kada korisnicima budete davali programe napisane na Perlu, obično ćete im davati izvorni kôd, a ne neprovidne programe.

Ako želite da pravite neprovidne programe, moramo vam reći da takvi programi ne postoje. Ukoliko korisnici mogu instalirati i pokrenuti vaš program, onda ga lako mogu pretvoriti u izvorni kôd na bilo kom jeziku. Naravno, to ne mora biti isti izvorni kôd koji ste vi koristili, ali će biti neka vrsta izvornog koda. Pravi način da tajni algoritam ostane skriven jeste korišćenje usluga advokata; oni mogu napisati dozvolu u kojoj se kaže „S ovim kodom se može uraditi *ovo*, ali ne i *ovo*. Ako prekršite pravila, imamo dovoljan broj advokata koji će vas naterati da zažalite zbog toga".

Kako da nabavim Perl?

Verovatno ga već imate. Najzad, mi vidimo Perl gde god da odemo. Perl se isporučuje uz brojne sisteme, a administratori sistema ga često instaliraju na sve računare u svojoj mreži. Ukoliko ga ne pronađete u svom sistemu, možete ga nabaviti besplatno.

Perl se distribuira pod dve različite licence. Za većinu korisnika nije bitno o kojoj licenci je reč. Ipak, ako menjate mogućnosti Perla, trebalo bi da pažljivije pročitate uslove navedene u licenci jer postoje manja ograničenja prilikom distribuiranja izmenjenog koda. Korisnici koji ne žele da menjaju Perl, u licenci će naći i sledeći red „It's free – have fun with it" (Perl je besplatan – zabavite se koristeći ga).

Dakle, Perl je besplatan i lepo radi na svim računarima koji se odazivaju na ime Unix i imaju prevodilac za programski jezik C. Dovoljno je da ga preuzmete, unesete par komandi i on će početi da se sam podešava. Još je bolje ako uspete da nagovorite administratora sistema da te dve komande unese umesto vas i da ga instalira.* Pored sistema s Unixom, korisnicima se toliko svideo Perl da su ga prebacili i na druge platforme, kao što su Macintosh†, VMS, OS/2, MS/DOS, sve moderne verzije Windowsa i druge.‡ Većina tih verzija Perla ima instalacioni program koji se lakše koristi nego pod Unixom. Detaljnije informacije potražite u odeljku „ports" o mreži CPAN.

* Ako administratori sistema ne mogu da instaliraju softver, kakva je onda korist od njih? Ukoliko imate problema da ubedite administratora da instalira Perl, ponudite se da mu kupite picu. Jos nismo upoznali administratora sistema koji bi odbio besplatnu picu ili neku sličnu ponudu.

† MacPerl radi pod „klasičnim" operativnim sistemom za Mac. Ako imate operativni sistem Mac OS X, koji je zasnovan na Unixu, onda imate glavni Perl.

‡ U trenutku kada ovo pišemo, ne postoji verzija Perla za lične digitalne pomoćnike Palm. Suviše je veliki, čak i kada se izbaci veliki broj mogućnosti. Ipak, čuli smo glasine da Perl može da radi na ličnim digitalnim pomoćnicima sa operativnim sistemom WinCE.

Šta je CPAN?

CPAN je skraćenica za „sveobuhvatnu arhivsku mrežu za Perl“ (engl. *Comprehensive Perl Archive Network*), jedan od najvećih izvora informacija o Perlu. Ovaj izvor sadrži izvorni kôd samog Perla, verzije Perla za sve sisteme koji ne rade pod Unixom*, primere programa, dokumentaciju, dodatke i arhivirane poruke o Perlu. Ukratko, CPAN je sveobuhvatan izvor informacija o Perlu.

Izvor CPAN je kopiran na stotine računara širom sveta. Pretragu možete započeti na adresama *http://search.cpan.org* i *http://kobesearch.cpan.org/*. Ukoliko nemate pristup Internetu, možda možete nabaviti CD ili DVD sa svim korisnim informacijama iz mreže CPAN. Potražite ih u lokalnim knjižarama koje prodaju tehničku literaturu. Budući da se CPAN dnevno ažurira, potražite neku noviju arhivu informacija. Arhiva od pre dve godine sigurno je zastarela. Još bolje je da sa Interneta snimite kompakt disk s najsvežijim informacijama.

Gde da potražim podršku za Perl?

Pa, pošto imate potpun izvorni kôd, sami ispravljajte greške.

Ovo baš ne zvuči dobro, zar ne? Ipak, korisno je. Budući da izvorni kôd Perla nema vlasnika, svi korisnici mogu ispravljati greške. U stvari, kada otkrijete grešku, velika je verovatnoća da ju je neko već ispravio. U svetu postoje hiljade korisnika koji održavaju Perl.

Ne želimo da kažemo kako Perl ima mnogo grešaka, ali to je program, a svaki program ima najmanje jednu grešku. Da biste videli koliko je korisno imati izvorni kôd Perla, zamislite da ste umesto Perla dobili dozvolu za korišćenje programskog jezika pod imenom Forehead, koji je u vlasništvu velike i moćne kompanije čiji je vlasnik zilioner s lošom frizurom. (Ovo je naravno samo pretpostavka. Svi znaju da ne postoji jezik koji se zove Forehead.) Sada razmislite šta biste uradili kada biste pronašli grešku u Foreheadu. Prvo, možete je prijaviti. Drugo, možete se nadati da će ona biti ispravljena, da će se ispravka pojaviti brzo i da nova verzija neće mnogo koštati. Isto tako, možete se nadati i da nova verzija neće imati dodatne mogućnosti s novim greškama, kao i da velika kompanija neće propasti u antimonopolskom sudskom procesu.

Kada imate Perl, imate i izvorni kôd. U retkom i malo verovatnom slučaju da grešku ne možete ispraviti na drugi način, možete unajmiti jednog ili više programera koji će obaviti posao umesto vas. Isto tako, ako kupite računar na kome Perl ne može da radi, sami možete napraviti verziju za taj računar, a ako vam treba nova mogućnost, znate šta treba da radite.

* Skoro uvek je bolje prevesti programe napisane u Perlu na sistemima koji rade pod Unixom. Drugi sistemi možda nemaju prevodilac za programski jezik C ili druge alatke potrebne za prevođenje.

Postoje li druge vrste podrške?

Naravno. Jedna od naših omiljenih je Perl Mongers. To je svetsko udruženje korisnika Perla; detaljnije informacije potražite na adresi *http://www.pm.org*. Verovatno i u vašoj blizini postoji grupa korisnika u kojoj se nalaze stručnjaci za Perl. Ukoliko takva grupa ne postoji, možete je osnovati.

Naravno, nemojte zanemariti dokumentaciju, kao prvi nivo podrške. Pored takozvanih *man strana*[*] (engl. *manpages*), dokumentaciju možete pronaći i na mreži CPAN (*http://www.cpan.org*) ili na drugim Web lokacijama poput *http://perldoc.perl.org*, na kojoj se nalaze HTML i PDF verzije Perlove dokumentacije, *http://www.perldoc.com*, koja omogućava pretraživanje različitih verzija dokumentacije i *http://faq.perl.org*, koja sadrži najnovije odgovore na često postavljana pitanja o Perlu.

Još jedan dobar izvor informacija o Perlu jeste i knjiga *Programiranje na Perlu* (engl. *Programming Perl*) izdvačke kuće O'Reilly, poznatija pod nazivom „knjiga sa kamilom", zbog kamile koja se nalazi na koricama. Knjiga sa kamilom sadrži potpune referentne informacije, nešto materijala udžbeničkog tipa i obilje raznih podataka o Perlu. Postoji i džepna verzija knjige, *Perl 5 Pocket Reference* (*Džepno uputstvo za Perl 5*) Johana Vromansa (O'Reilly), koju je zgodno imati pri ruci (ili u džepu).

Ukoliko morate postaviti pitanje, postoje brojne diskusione grupe na Usenetu i liste slanja.[†] U bilo koje doba dana ili noći, neki od stručnjaka za Perl spreman je da odgovori na vaše pitanje; sunce nikada ne zalazi u carstvu Perla. To znači da ćete odgovor na pitanje najčešće dobiti za samo nekoliko minuta, ali ako rešenje niste prvo potražili u dokumentaciji ili među odgovorima na često postavljana pitanja, bićete kritikovani.

Zvanične diskusione grupe na Usenetu nalaze se u delu hijerarhije pod nazivom *comp.lang.perl.**. U vreme pisanja ove knjige, postojalo je pet diskusionih grupa, ali se one menjaju s vremena na vreme. Trebalo bi da se prijavite bar na diskusionu grupu *comp.lang.perl.announce.**, koja sadrži važna obaveštenja o Perlu, i gde je naročita pažnja posvećena bezbednosti. Ako ne znate kako se radi s diskusionim grupama, potražite pomoć od lokalnog stručnjaka.

Iz diskusija o Perlu razvilo se nekoliko zajednica na Webu. U jednoj od popularnijih, poznatoj pod nazivom The Perl Monastery (*http://www.perlmonks.org*), učestvuju i brojni autori knjiga i članaka o Perlu, uključujući najmanje dva autora ove knjige. Druge informacije možete potražiti i na adresi *http://learn.perl.org* i odgovarajućoj listi slanja *beginners@perl.org*.

Ukoliko vam zatreba ugovor o podršci za Perl, postoje brojne kompanije koje će vam to naplatiti. Druge grupe za podršku obezbediće vam pomoć besplatno.

[*] Izraz *manpages* je preuzet iz Unixa i označava dokumentaciju. Ukoliko ne koristite sistem pod Unixom, trebalo bi da se *man* strane za Perl nalaze u okviru dokumentacije sistema.

[†] Većina listi slanja može se pronaći na adresi *http://lists.perl.org*.

Šta da radim ako pronađem grešku u Perlu?

Prvo treba da ponovo* pročitate dokumentaciju†. Perl ima toliko specifičnih mogućnosti i izuzetaka od pravila, da postoji i verovatnoća da nije u pitanju greška. Uverite se da nemate stariju verziju Perla; možda ste otkrili grešku koja je ispravljena u novijim verzijama.

Kada ste skoro sigurni da ste otkrili grešku, raspitajte se da li ju je možda još neko otkrio: kolega s posla, programeri koje srećete na sastancima grupe Perl Mongers ili neki od stručnjaka na konferencijama o Perlu. Velika je verovatnoća da je u pitanju neka od specifičnih mogućnosti, a ne greška.

Kada ste potpuno sigurni da ste otkrili grešku, napravite program za njeno testiranje. (Nije valjda da to već niste uradili?) Idealan program bi trebalo da bude mali tako da ga svi korisnici Perla mogu pokrenuti i uveriti se u pogrešno ponašanje koje ste otkrili. Kada napravite program koji jasno prikazuje grešku, iskoristite uslužni program *perlbug* (isporučuje se uz Perl) da biste je prijavili. Na taj način ćete stručnjacima za razvoj Perla poslati poruku e-poštom, pa zato nemojte koristiti *perlbug* pre nego što budete imali spreman program za testiranje.

Posle slanja poruke o otkrivenoj grešci, trebalo bi da dobijete odgovor u roku od nekoliko minuta. Najčešće ćete dobiti jednostavnu zakrpu koju ćete instalirati i nastaviti rad. Naravno, može se desiti (u najgorem slučaju) da ne dobijete odgovor, jer stručnjaci za razvoj Perla nemaju obavezu da čitaju i odgovaraju na poruke o pronađenim greškama, ali budući da svi volimo Perl, niko ne želi da greška ostane neispravljena.

Kako da napišem program na Perlu?

Vreme je da postavite to pitanje (čak i ako niste). Programi napisani na Perlu tekstualne su datoteke; možete ih praviti i menjati pomoću omiljenog programa za uređivanje teksta, tzv. editora teksta (engl. *text editor*). (Ne treba vam nikakvo posebno razvojno okruženje, mada postoje neke komercijalne verzije. Nismo nikada koristili nijedno od njih, pa vam ih ne možemo preporučiti.)

Umesto običnog, trebalo bi da koristite neki od editora teksta namenjen programerima. U čemu je razlika? Editor teksta namenjen programerima omogućava obavljanje programerski specifičnih zadataka poput uvlačenja delova koda ili uparivanja zatvorenih vitičastih zagrada s otvorenim. U sistemima koji rade pod Unixom, dva najpopularnija editora teksta namenjena programerima jesu *emacs* i *vi* (i njihove razne verzije i kopije). BBEdit i Alpha su dobri programi za operativni sistem Mac OS X, a veliki broj korisnika pohvalio je i UltraEdit i Programmer's Favorite Editor (PFE) za Windows. *Man* strana `perlfaq2` sadrži spisak nekoliko drugih editora teksta. Posavetujte se sa lokalnim ekspertom oko editora teksta na vašem sistemu.

* Možda čak dva ili tri puta. U većini slučajeva, u dokumentaciji smo tražili objašnjenja za neočekivano ponašanje Perla i otkrivali nove načine korišćenja koje smo opisivali na prezentacijama ili u člancima.

† Čak i Lari priznaje da s vremena na vreme potraži pomoć u dokumentaciji.

Kada je reč o jednostavnim programima koje ćete pisati za potrebe vežbi iz ove knjige, od kojih nijedan neće imati više od dvadeset ili trideset redova koda, poslužiće bilo koji editor teksta.

Neki početnici su pokušavali da umesto editora teksta koriste program za obradu teksta (engl. *text processor*). Mi to ne preporučujemo, jer to može biti nezgodno (u najboljem slučaju) ili nemoguće (u najgorem slučaju). Ipak, nećemo vas sprečiti da to radite. Ne zaboravite samo da datoteke treba da budu tekstualne; matični format datoteka programa za obradu teksta zasigurno će biti neupotrebljiv. U većini programa za obradu teksta najverovatnije ćete dobiti upozorenje o pravopisnim greškama i obaveštenje da bi trebalo da koristite manje zagrada.

U nekim slučajevima moraćete da pišete programe na jednom računaru, a pokrećete na drugom. Nemojte zaboraviti da datoteke prebacujete u tekstualnom ili ASCII formatu, a ne u binarnom. Ovaj korak je neophodan, jer se tekstualni formati razlikuju na različitim računarima. Ako to ne uradite, nećete dobiti očekivane rezultate. Neke verzije Perla odbiće izvršavanje programa ako primete grešku u oznakama za kraj reda.

Jednostavan program

Poštujući najstarije pravilo, sve knjige koje opisuju programske jezike s korenima u Unixu moraju imati program „Zdravo svete“. Evo njegove verzije u Perlu:

```
#!/usr/bin/perl
print "Zdravo svete!\n";
```

Zamislimo da ste ovaj kôd uneli u editor teksta. (Nemojte još uvek brinuti o tome šta ovaj kôd znači i kako radi. To ćemo videti uskoro.) Program možete snimiti pod bilo kojim imenom. Perl ne zahteva korišćenje posebnih imena datoteka i njihovih nastavaka, a najbolje je da nastavke u imenima datoteka uopšte ne koristite.* Ipak, neki sistemi zahtevaju korišćenje nastavka *.plx*, koji označava Perlovu izvršnu datoteku (engl. *PerL eXecutable*); detaljnije informacije potražite u uputstvu za vaš sistem.

Možda ćete svoj sistem morati da podesite tako da prepoznaje ove izvršne datoteke. Šta ćete uraditi, zavisi od vrste sistema; možda ćete morati da uradite više od snimanja programa na određeno mesto. (Tekući direktorijum je dobro mesto.) U sistemima pod Unixom, izvršni programi se označavaju pomoću komande `chmod`:

```
$ chmod a+x moj_program
```

Znak $ (i razmak) na početku reda označava da je u pitanju odzivnik komandnog okruženja (engl. *shell prompt*), koji će na vašem sistemu verovatno izgledati drugačije.

* Zašto je bolje da se ne koriste nastavci u imenima datoteka? Zamislite da ste napisali program koji računa rezultat kuglanja i da ste svim prijateljima rekli da se on zove *kuglanje.plx*. Jednog dana odlučite da taj program napišete na jeziku C. Da li ćete mu dodeliti isto ime, koje označava da je program napisan na Perlu, ili ćete prijatelje obavestiti da program ima novo ime? (Molim vas, nemojte ga nazvati *kuglanje.c!*) Odgovor je da njih ne zanima na kom jeziku je program napisan, jer ga oni samo *koriste*. Zato je program od početka trebalo da se zove samo *kuglanje*.

Ukoliko ste navikli da u komandi `chmod` umesto simboličkih parametara poput `a+x` koristite brojeve (`755`, na primer), nema problema. U svakom slučaju, sistem će znati da je navedena datoteka program.

Sada ste spremni da ga pokrenete:

```
$ ./moj_program
```

Tačka i kosa crta na početku komande označavaju da program treba tražiti u tekućem radnom direktorijumu. Ta dva znaka nisu potrebna u svim slučajevima, ali bi trebalo da ih koristite na početku svih komandi dok potpuno ne shvatite njihovu funkciju.* Ukoliko je sve proradilo, to je pravo čudo. Češće ćete otkriti da vaš program ima grešku. Izmenite ga i pokušajte ponovo da ga pokrenete, ali nemojte koristiti komandu `chmod` svaki put, jer bi mogla da se „zalepi“ za datoteku. (Naravno, ako je greška to što niste pravilno iskoristili komandu `chmod`, verovatno ćete dobiti grešku s objašnjenjem da je pristup datoteci onemogućen.)

Šta se nalazi unutar programa?

Kao i drugi „slobodni“ jezici, i Perl vam omogućava da po želji koristite beline (razmake, tabulatore i oznake za novi red) kako bi vaš program bio čitljiviji. Ipak, većina programa napisanih na Perlu koristi prilično standardan format, sličan onom koji je korišćen u ovoj knjizi. Mi vam toplo preporučujemo da pravilno „nazubljujete“ programe jer će se oni tako lakše čitati; dobar editor teksta uradiće veći deo posla umesto vas. Dobri komentari takođe olakšavaju čitanje programa. U Perlu se komentari prostiru od znaka # do kraja reda. (U Perlu ne postoje „komentari blokova koda“.)† U programima iz ove knjige nismo koristili mnogo komentara jer se ponašanje programa opisuje u tekstu, ali bi vi u svojim programima trebalo da koristite komentare po potrebi.

Drugi način (čudan, mora se priznati) za pisanje programa „Zdravo svete“ mogao bi biti sledeći:

```
#!/usr/bin/perl
    print    # Ovo je komentar
"Zdravo svete!\n"
  ;    # Nemojte ovako pisati programe na Perlu!
```

Prvi red je poseban komentar. Ako su u sistemima pod Unixom‡ prva dva znaka u prvom redu teksta „`#!`“, to znači da preostali deo reda predstavlja ime programa koji služi za izvršavanje ostatka datoteke. U ovom slučaju program se nalazi u datoteci */usr/bin/perl*.

* Ukratko, sprečava se pokretanje drugog programa (ili dodatka) istog imena. Uobičajena greška među početnicima je da prvom programu daju ime `test`. Većina sistema ima program (ili dodatak) tog imena, koji će biti pokrenut umesto korisnikovog programa.

† Postoje brojni načini da rešite ovaj problem. Pogledajte odgovore na često postavljana pitanja (dobićete ih korišćenjem komande *perldoc perlfaq*).

‡ Većina modernih sistema. Mehanizam „shebang“ (šebang) pojavio se sredinom osamdesetih godina prošlog veka pa je prilično star.

Red koji počinje znacima `#!` najmanje je prenosiv deo programa napisanih na Perlu jer nije isti za sve sisteme. Srećom, najčešće je reč o putanjama */usr/bin/perl* ili */usr/local/bin/perl*. Ukoliko to nije slučaj, moraćete da saznate gde se Perl nalazi u vašem sistemu i da koristite tu putanju. U sistemima pod Unixom mogli biste iskoristiti pomoćni red koda koji će umesto vas pronaći putanju na kojoj se Perl nalazi:

```
#!/usr/bin/env perl
```

Ako se Perl ne nalazi ni u jednom zadatom direktorijumu, podatke o pravoj putanji moraćete da saznate od lokalnog administratora sistema ili od nekog korisnika koji ima isti sistem kao vi.

U sistemima koji ne rade pod Unixom uobičajeno je (pa čak i korisno) da prvi red bude `#!perl`. Ako ništa drugo, ova komanda obaveštava programere da je u pitanju program napisan na Perlu.

Ukoliko je red koji počinje znacima `#!` pogrešan, komandno okruženje će prikazati poruku o grešci. To može biti i neka neočekivana greška (na primer, „file not found“, datoteka nije pronađena). Ipak, ne radi se o tome da vaš program nije pronađen; putanja */usr/bin/perl* ne nalazi se tamo gde treba da bude. Mi bismo rado razjasnili poruku, ali nju ne prikazuje Perl već komandno okruženje. (Budite pažljivi da ne biste umesto reči *usr* uneli reč *user*; tvorci Unixa su bili prilično lenji, pa su izostavili mnogo slova.)

Još jedan problem na koji biste mogli naići jeste potpuno odsustvo podrške za redove koji počinju znacima `#!`. U tom slučaju, sistem će verovatno sâm pokrenuti program, uz rezultate koji će vas razočarati ili zapanjiti. Ukoliko ne razumete šta znače čudne poruke o greškama, potražite ih na *man* strani `perldiag`.

„Glavni“ program se sastoji od uobičajenih izraza u Perlu (nema potprograma, što ćete videti kasnije). Ne postoji „glavna“ funkcija, kao što je slučaj u jezicima poput C-a ili Jave. U stvari, većina programa nema funkcije (u obliku potprograma).

Perl ne zahteva postojanje odeljka za deklarisanje promenljivih kao što je slučaj u nekim drugim jezicima. Ukoliko ste navikli da deklarišete promenljive, ova činjenica vas u početku može iznenaditi. Ipak, ona omogućava da se kratki programi u Perlu pišu vrlo brzo. Ukoliko vaš program ima samo dva reda, sigurno ne želite da jedan od njih potrošite na deklarisanje promenljivih. Ukoliko ipak želite da ih deklarišete, u tome nema ništa loše; u četvrtom poglavlju ćete saznati kako se to radi.

Većinu naredaba čine izrazi koji se završavaju znakom tačka i zarez. Evo jedne koju ste videli nekoliko puta do sada:

```
print "Zdravo svete!\n";
```

Kao što pretpostavljate, ovaj red koda služi za prikazivanje poruke `Zdravo svete!`. Na kraju svake poruke nalazi se prečica `\n`, koja vam je verovatno poznata ako ste koristili jezike kao što su C, C++ ili Java; ona označava prelazak u novi red. Kada se ova prečica nađe na kraju poruke, ispis prelazi u novi red, omogućavajući tako da se

odzivnik pojavi u novom redu, a ne pored poruke. Sve poruke treba završavati oznakom za prelazak u novi red. U sledećem poglavlju naći ćete detaljnije informacije o korišćenju prečice za prelazak u novi red i drugim prečicama koje počinju obrnutom kosom crtom.

Kako se prevodi program napisan na Perlu?

Samo ga pokrenite. Interperter će prevesti i pokrenuti program u jednom koraku.

```
$ perl moj_program
```

Kada pokrenete program, Perlov interpreter će prvo proći kroz ceo izvorni kôd, pretvarajući ga u *bajtkod* (engl. *bytecode*), što je unutrašnja struktura podataka koja predstavlja program. Potom će Perlova mašina za rad s bajtkodom preuzeti kontrolu i pokrenuti napravljeni bajtkod. Ukoliko postoji sintaksna greška u redu s brojem 200, dobićete odgovarajuću poruku pre nego što počne izvršavanje sledeće naredbe.* Ako imate petlju koja se ponavlja 5000 puta, ona će se prevesti odjednom; tako će petlja raditi punom brzinom. Veliki broj komentara i belina neće uticati na brzinu programa. Ukoliko koristite proračune samo s konstantama, rezultat će biti konstanta koja će se izračunati samo na početku petlje.

Da budemo iskreni, prevođenje može da potraje. Nije dobro napisati obiman program na Perlu koji obavlja jednostavan zadatak, jer će vreme prevođenja daleko nadmašiti vreme izvršavanja programa (engl. *runtime*). Ipak, prevodilac je prilično brz; najčešće će vreme prevođenja programa biti mnogo kraće od vremena njegovog izvršavanja.

Izuzetak može biti slučaj kada pišete program koji će raditi kao CGI skript, koji se može pozivati stotinama ili hiljadama puta u minutu. (To je ogromna učestalost. Da se skript poziva nekoliko stotina ili hiljada puta na dan, kao što je slučaj s većinom programa na Webu, verovatno ne bismo toliko brinuli.) Većina programa ima kratko vreme izvršavanja, tako da vreme prevođenja postaje značajno. Ukoliko to i vama predstavlja problem, verovatno ćete želeti da pronađete način da program zadržite u memoriji između poziva. Pomoći će vam dodatak mod_perl za Web server Apache, koji se nalazi na adresi *http://perl.apache.org* ili Perlovi moduli kao što je `CGI::Fast`.

Da li se prevedeni bajtkod može sačuvati da bi se izbeglo ponovno prevođenje ili, još bolje, da li se bajtkod može prebaciti u neki drugi jezik (C, na primer), a zatim prevesti? Ovo je moguće izvesti u nekim slučajevima, ali se programi verovatno neće lakše koristiti, održavati ili instalirati, a biće i sporiji. Perl 6 bi trebalo da se mnogo bolje pokaže u ovom pogledu, ali je još rano da se o tome govori.

* Osim ako je sledeći red neka operacija koja se obavlja u vreme prevođenja, kao što je, na primer, blok `BEGIN`.

Kratak vodič kroz Perl

Hoćete da vidite neki koristan program napisan na Perlu? Evo ga:

```
#!/usr/bin/perl
@lines = `perldoc -u -f atan2`;
foreach (@lines) {
  s/\w<([^>]+)>/\U$1/g;
  print;
}
```

Ovaj kôd napisan na Perlu sigurno vam izgleda čudno kada ga vidite prvi put. (U stvari, on će vam izgledati čudno kad god ga budete videli.) Hajde da objasnimo svaki red koda i da vidimo šta on radi. (Objašnjenja su kratka jer je ovo kratak vodič. Do kraja knjige detaljnije ćete upoznati sve mogućnosti ovog programa. Ne morate ga odmah sasvim razumeti.)

Prvi red počinje znacima `#!`, kao što ste već videli. Možda ćete taj red morati da prilagodite svom sistemu, kao što smo već objasnili.

Drugi red pokreće spoljašnju komandu čije se ime nalazi između obrnutih jednostrukih navodnika (engl. *backticks*) („` `“). (Na tastaturama sa američkim rasporedom, obrnuti jednostruki navodnik obično se nalazi odmah pored tastera 1. Pazite da umesto obrnutih jednostrukih navodnika ne unesete jednostruki navodnik tj. polunavodnik.) Komanda koju smo iskoristili je `perldoc -u -f atan2`; unesite je na komandnu liniju da biste videli njene rezultate. Komanda `perldoc` se na većini sistema koristi za čitanje i prikazivanje dokumentacije za Perl, njegove dodatke i uslužne programe, pa bi trebalo da bude dostupna.* Upotrebljena komanda prikazuje informacije o trigonometrijskoj funkciji `atan2`; ovde je koristimo kao primer spoljašnje komande čije rezultate želimo da upotrebimo.

Rezultat komande navedene između obrnutih jednostrukih navodnika snima se u nizovnu promenljivu (engl. *array variable*) pod imenom `@lines`. Sledeći red koda započinje petlju koja obrađuje svaki red rezultata. Unutar petlje, naredbe su uvučene. Iako Perl to ne zahteva, dobri programeri to uvek rade.

Najstrašnije deluje prvi red unutar tela petlje: `s/\w<([^>]+)>/\U$1/g;`. Bez ulaženja u detalje, reći ćemo samo da ova naredba menja sve redove sa specijalnom oznakom napravljenom od ugaonih zagrada (`< >`), a morao bi da postoji bar jedan takav red kao rezultat komande `perldoc`.

Sledeći red – na opšte iznenađenje – prikazuje sve (verovatno izmenjene) redove. Prikaz će ličiti na rezultat komande `perldoc -u -f atan2`, samo što će biti izmenjeni oni redovi koji sadrže pomenute oznake.

* Ako komanda `perldoc` nije dostupna, to verovatno znači da vaš sistem nema komandnu liniju i da Perl ne može pokretati komande navedene između obrnutih jednostrukih navodnika niti pomoću komande piped-open, što ćete videti u četrnaestom poglavlju. U tom slučaju preskočite vežbe koje koriste komandu `perldoc`.

Tako smo u samo nekoliko redova pokrenuli spoljašnji program, snimili njegove rezultate u memoriju, izmenili ih i prikazali. Ova vrsta programa je prilično uobičajena u Perlu, gde se jedna vrsta podataka pretvara u drugu.

Vežbe

Svako poglavlje ćemo završavati vežbama, a njihova rešenja pronaći ćete u dodatku A. Ne morate pisati programe jer se oni nalaze u tekstu poglavlja.

Ukoliko programi ne rade na vašem računaru, proverite ih još jednom i posavetujte se s lokalnim stručnjakom. Ne zaboravite da svaki program morate malo podesiti, kao što je opisano u tekstu.

1. [7] Unesite program „Zdravo svete“ i probajte da li radi. (Možete ga nazvati kako god želite, ali mi zbog jednostavnosti predlažemo ime `vez1-1`, jer je ovo prva vežba u prvom poglavlju.)
2. [5] Iza komandnog odzivnika unesite komandu `perldoc -u -f atan2` i pogledajte njene rezultate. Ukoliko komanda ne radi, posavetujte se s lokalnim administratorom sistema ili pogledajte u dokumentaciji kako se ova komanda pokreće u vašoj verziji Perla. (Ta informacija će vam trebati i za sledeću vežbu.)
3. [6] Unesite drugi program (iz prethodnog odeljka) i pogledajte njegove rezultate. (Savet: znakove interpunkcije unosite pažljivo, tačno onako kako su napisani.) Vidite li kako su se rezultati komande izmenili?