Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 004.3
DOI: https://doi.org/10.24867/07BE14Bucko

KARAKTERISTIKE I UPOTREBLJIVOST PROGRAMSKOG JEZIKA GO U MODERNIM APLIKACIJAMA

CHARACTERISTICS AND USABILITY OF THE GO PROGRAMMING LANGUAGE IN MODERN APPLICATIONS

Milica Bučko, Srđan Popov, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U radu su detaljno istražene i opisane osnovne karakteristike programskog jezika Go počevši od tipova podataka, kreiranja promenljivih preko naredbi toka, paketiranja do konkurentnosti. Razmatrane su prednosti i mane programskog jezika Go u odnosu na programski jezik Java.*

Ključne reči: *Go, Golang, statički tipiziran jezik, Java*

Abstract – *The paper analyzes basic characteristics of the Go programming language starting from data types, creating variables, control flow, packageing till concurencz and parallelism. The advantages and disadvantages of the Go programming language compared to the Java programming language are discussed.*

Keywords: *Go, Golang, static typed language, Java*

1. UVOD

Programski jezik Go, razvila je kompanija Google 2007. Godine u cilju unapređenja produktivnosti u programiranju. Cilj je bio da se jezik Go dizajnira kao statički kompajliran jezik koji se koristi za velike sisteme ali da pri tom programi koji se pišu budu jednostavni za programiranje sa čitljivim kodom, bez prevelikog broja ključnih reči, kao kod dinamički tipiziranih programskih jezika.

Cilj rada je upoznavanje sa programskim jezikom Go, njegovom sintaksom i njegovim karakteristikama. Na studiji slučaja je izvršena komparativna analiza programskog jezika Go sa programskim jezikom Java.

2. PODEŠAVANJE OKRUŽENJA

Procedura instaliranja Go-a na lokalnoj mašini sa Windows operativnim sisemom i podešavanje programskog okruženja putem komandne linije.
Za pokretanje prvog programa pisanog u Go programskom jeziku potrebno je instalirati Go run time i podesiti lokalnu mašinu u cilju pokretanja, buildovanja i izvršavanja programa. Zatim je potrebno uspešno instalirati editor, VSCode. Nakon instalacije VSCode, potrebno ga je konfigurisati kako bi editor znao da rukuje sa kodom.

NAPOMENA:

Ovaj rad proistekao je iz master rada čiji mentor je bio dr Srđan Popov, vanr. prof.

2.1. Komande za pokretanje programa

Nakon konfiguracije okruženja moguće je pokrenuti prvi program napisan u Go-u. Neke od osnovnih komandi za pokretanje projekta su go build, go run, go fmt, go install, go get i go test.

3. KARAKTERISTIKE PROGRAMSKOG JEZIKA GO

3.1. Deklaracija i definicija

Kako je Go statički tipiziran jezik prilikom deklaracije promenljive potrebno je dodeliti tip podatka. Pravilo Go programskog jezika je da se svaka deklarisana promenljiva mora iskorititi, mora biti upotrebljena u toku izvršavanja programa. Ovo ne važi samo za promenljive nego i za pakete. Postoji više načina definisanja promenljivih.

3.1.1. Tip promenljive se eksplicitno navodi

Postoje dva slučaja kreiranja promenljivih kada se tip promenljive eksplicitno navodi. Prvi slučaj je definicija i deklaracija promenljive gde se tip promenljive eksplicitno navodi u istom iskazu. Drugi slučaj je deklaracija promenljive gde se tip promenljive eksplicitno navodi a dodjeljivanje se vrši nakon deklaracije u odvojenom iskazu.

3.1.2. Tip promenljive se ne navodi

U slučaju kada se deklaracija promenljive vrši tako što se tip promenljive ne navodi eksplicitno. Na osnovu vrednosti, kompajler treba sam da zaključi koja je vrednost toj promenljivoj dodeljena.

3.1.3. Deklaracije više promenljivih

U slučaju deklaracije više promenljivih, tip promenljive kompajler zaključuje na osnovu vrednosti koja joj se dodeli. Tip promenljive u jednom iskazu ne mora da bude isti. Jedna od dodatnih opcija koje Go jezik nudi, je lak način deklarisanja promenljive bez navođenja ključne reči var, pomoću operatora :=.

Operator **:=** govori kompajleru da je programer naveo ime i vrednost promenljive a on mora sam da zaključi koji tip će joj dodeliti. Ova operacija se koristi samo kada se prvi put navodi nova promenljiva.

3.2. Tipovi podataka

Tipovi podataka koji se mogu koristiti u programskom jeziku Go svrstavaju se u četiri kategorije: osnovni tipovi

(tekstualni, bulean i numerički), složeni tipovi (nizovi i strukture), referentni tipovi (pokazivači, isečci, mape, kanali i funkcije), interfejsni tipovi (interfejsi).

3.2.1. Osnovni tipovi podataka

Osnovni tipovi podataka podeljeni su od tri grupe: numerička grupa (integer, float i kompleksne vrednosti), bool grupa (true i false) i tekstualna grupa predstavlja niz karaktera.

Operatori definisani u Go jeziku su aritmetički operatori, relacioni operatori, logički operatori, bitski operatori i operatori dodele.

3.2.2. Složeni tipovi podataka

Pod složenim tipovima podataka u programskom jeziku Go podrazumevaju se nizovi i strukture.

Nizovi predstavljaju sekvencu elemenata istog tipa, fiksne veličine. Nije moguće kreirati niz takav da sadrži različite tipove podatala (npr. integer i string). Nije moguće menjati veličinu niza nakon što se jednom definiše. Elementima niza pristupa se pomoću indexa, pri čemu prvi element niza ima indeks 0 a poslednji size-1. Pomoću ugrađene funkcije len moguće je dobiti podatak o veličini niza. Još jedna opcija koju Go jezik podržava je poređenje nizova. Nizovi u tom slučaju moraju biti iste dužine i istog tipa podataka.

Strukture su korisnički definisani tipovi koji sadrže nula ili više elemenata istih ili različitih tipova. Strukture se koriste kada korisnik želi da grupiše povezane podatke u jednu celinu. Svaki element strukture mora imati jedinstven naziv, definisan tip i predstavlja jedno polje strukture. Definisanje strukture zahteva navođenje predefinisane reči **type** ispred same definicije strukture. Poljima unutar strukture moguće je pristupiti navođenjem naziva polja.

Go programski jezik podržava ugnježdavanje struktura. Poređenje struktura je takođe podržano ukoliko su dve strukture koje se porede istog tipa.

3.2.3. Referenti tipovi podataka

Pod referentnim tipovima podataka u programskom jeziku Go podrazumavaju se pokazivači, isečci, mape, kanali i funkcije.

Pokazivač je promenljiva koja pokazuje na memorijsku lokaciju neke druge promenljive. Oni čuvaju memorijsku adresu promenljive na koju pokazuju i svesni su gde je locirana kao i vrednosti koja je sačuvana na toj lokaciji.

Isečak ili **slice** u Go programskom jeziku predstavlja jedan deo ili segment niza. Pogled na elemente niza.

Jedna od najkorisnijih stuktura u programiranju jesu heš tabele (hash table), **mape**. U tabeli se čuvaju parovi ključ – vrednost. Ključ mora biti jedinstven.

Funkcije su delovi koda zaduženi za izvršavanje nekog zadatka na osnovu informacija koje su dobile kao ulazni parametar. Kada se jednom kreira, funkcija može da bude pozvana u bilo kom trenutku proizvoljan broj puta. Opciono je da li će funkcija vraća ti neku vrednost ili ne. U Go jeziku, funkcija se definiše pomoću ključne reči **func**.

3.2.4. Interfejsni tipovi

Interfejs predstavlja kolekciju metoda koje određeni objekat može da implementira, što znači da interfejs definiše ponašanje objekta. Ukoliko objekat implementira metode interfejsa kaze se da objekat implementira taj interfejs.

U interfejsu treba da budu navedena imena metoda, ulazni argumenti i povratni tipovi.
Za deklaraciju interfejsa potrebno je navesti **type** alias zajedno za ključnom reči **interface.**

3.2.5. Konstante

Konstante predstavljaju promenljive čija vrednost ne može biti promenjena kad se jednom dodeli. Konstante u Go programskom jeziku se deklarišu pomoću ključne reči const.

3.3. Naredbe kontrole toka

3.3.1. Naredba IF

U Go programskom jeziku, postoje dve opcije upotrebe naredbe if. Korišćenje naredbe if ikoliko je potrebno proveriti neki uslov bez izvršavanja inicijalizacije i korišćenje naredbe if ukoliko je potrebno navesti naredbu koja je znakom ; razdvojena od uslova.

3.3.2. Naredba FOR

U Go programskom jeziku jedina petlja koja postoji je FOR petlja. Prilikom korišćenja FOR petlje za prelazak na sledecu iteraciju koristi se ključna reč **continue** dok se za izlazak iz petlje koristi ključna reč **break**. Takodje, prilikom upotrebe for petlje koristi se i ključna reč **range** koja omogućava iteriranje kroz sekvencijalne strukture kao što su mape, isešci i nozovi.

3.3.3. Naredba SWITCH

Postoji više načina definisanja naredbe switch. Ova naredba omogućava proveravanje vrednosti po slušajevima. Prilikom korisćenja switch naredne koristi se ključna reč **switch** gde se navodi uslov i ključna reč **case** koja proverava različite slučajeve. Switch naredba izvršava prvi case koji zadovoljava zadati uslov. Ukoliko ni jedan case ne zadovoljava uslov izvršiće se **default** izraz.

3.3.4. Naredba DEFER

Naredba defer odlaže izvršavanje navedene funkcije dok se ne završi izvršavanje funkcije u kojoj se nalazi. U trenutku navođenja naredbe defer određuju se argumenti funkcije dok se samo njeno izvršavanje odlaže. U slučaju greške, naredva defer garantuje da će se funkcija izvršiti. Naredba se koristi za operacije koje se obavljaju u paru.

3.4. Upravljanje greškama

Go jezik ne poseduje opciju try/catch metoda za rukovanje greškama već se greške vraćaju kao regularna povratna vrednost.
Greška, predstavlja vrednost koju funkcija može da vrati u slučaju da se tokom izvršavanja desi nešto neočekivano. Ključna reč za grešku je **error** i njena zero vrednost je **nil**.

Greška je zapravo tipa interfejs, dostupna na globalnom nivou i implementira Error metod koji kao povratnu vrednost vraća poruku o greški u String formatu.

3.5. Sakupljanje smeća

U Go programskom jeziku upravljanje memorijom odnosno sakupljanje smeća (garbage collection) obavlja se automatksi. Prvo, sakupljač skenira celu memoriju i pronalazi sve objekte na koje više niko ne referencira, briše ih i time oslobađa memoriju.
Verzijom Go 1.5 uveden je sakupljač smeća kreiran kao konkurentni, trobojni, označi-počisti(mark-sweep) sakupljač. Trobojni sakupljač je sakupljač kod kog je svaki objekat u jednoj od tri boje, bele, sive ili crne i hip segment koji se posmatra kao jedan povezani graf. Hip segment predstavlja deo memorije u kome se dinamički alociraju podaci tokom izvršavanja programa.

3.6. Paketi

Svaki Go program mora biti deo nekog paketa i najjednostavniji program mora imati package main deklaraciju. Ako je program deo main paketa, prilikom izvršavanja naredbe go install biće kreiran binaran (binary) fajl, koji nakon izvršavanja poziva main funkciju programa.
Postoje dva tipa paketa Executable koji generiše fajl koji može da se pokrene i Reusable-code dependensiji i biblioteke, ono što nam pomaže da ponovo koristimo kod u budućim projektima.
Reč main u package main navodi da je u pitanju Executable tip paketa. Ukoliko se izostavi navođenje main-a fajl se neće izvršiti. Kad god je definisan paket main to znači da je definisan paket koji može biti kompajliran i koji se može izvršiti. Takođe, obavezno je i navođenje funkcije main().

3.7. Konkurentnost i paralelizam

Iako se često konkurentnost i paralelizam posmatraju kao jedan pojam, između njih postoje suštinske razlike. Konkurentnost je mogućnost programa da poseduje više niti i da onog trenutka kad je jedna blokirana program izabere neku drugu i krene je izvrršavati. Ovaj proces se dešava na jednom jezgru. Paralelizam je mogućnost programa da poseduje više niti koje se izvršavaju u isto vreme na više različitih jezgara.

3.7.1. Rutine

Svaka nit izvršavanja predstavlja **rutine** (routine) u Go programskom jeziku. Svaka nova rutinu kreira se pomoću ključne reči „go“.

Pošto je Main rutina glavna nit i direktno od nje zavisi dužina izvršavanja programa, onda se zaključuje da onog momenta kad main rutina dođe do kraja svog izvršavanja, program će se završiti. Program nije u stanju da prepozna da postoje još neke nove rutine koje nisu završile svoj posao. Kako bi se izbegao ovaj problem, Go je uveo princip **kanala** (channels).

3.7.2. Kanali

Kanali (channels) se koriste kako bi se omogućila lakša komunikacija i razmena podataka između rutina. Pošto se podaci razmenjuju putem kanala potrebno je obezbediti da tip kanala i tip podataka koji se razmenjuju bude isti.

Kanal se kreira korišćenjem ključne reči **chan.** Kada je kanal napravljen u njega mogu da se šalju podaci i iz njega mogu da se primaju podaci. Kanal je posrednik u komunikaciji između main rutine i child rutina.

4. POREĐENJE PROGRAMSKOG JEZIKA JAVA SA PROGRAMSKIM JEZIKOM GO

4.1. Kontrola pristupa

Java sadrži private, protected, public modifikatore na osnovu kojih se objedinjuje opseg sa različitim nivoima pristupa za podatke, metode i pakete. Go sadrži exported i unexported identifikatore koji su slični public i private modifikatorima u Javi, ali nisu isti, što znači da Go ne sadrži modifikatore.

4.2. Golang nije objektno orijentisan jezik

Pre svega, Go nema objekte već strukture koje donekle mogu simulirati ponašanje objekata pa odmah u startu nema jednu od najvažnijih osobina objektno orijentisanog jezika, a to je predstavljanje entiteta putem objekata. Još jedna bitna razlika u odnosu na Javu je ta što Go jezik nema nasleđivanje zbog toga što ne podržava tradicionalni polimorfizam koji se ostvaruje putem nasleđivanja. Umesto nasleđivanja, U Go-u se koristi kompozicija.

4.3. Redosled deklaracije

Još jedna od velikih razlika između ova dva programska jezika je redosled definisanja (deklarisanje) varijabli.

Redosled navođenja može biti zbunjujući jer se razlikuje od velikog broja programskih jezika, ali suštinski se ništa ne gubi jer dobijamo isti efekat deklaracije kao i u bilo kom programskom jeziku.

4.4. Pokazivači

Java ne podržava pokazivače jer su kreatori želeli da dobiju na jednostavnosti i da spreče korisnika da direktno upravlja memorijom, te su uveli garbage collector koji se sam brine za oslobađanje zauzete memorije objekta koji više nisu u upotrebi.

Sa druge strane, Go podrazumeva pokazivače koji generalno imaju par svojih prednosti kao što su pristupanje putem adrese promenljive, dinamička alokacija memorije, ubrzavanje izvršavanja programa itd.

4.5. Funkcija kao argument

U Go programskom jeziku, funkcija može da se koristi kao varijabla koja će biti prosleđena nekoj drugoj funkciji.
Počevši od Java 1.8 i korišćenja lambda izraza, dobijena je slična funkcionalnost ali objekti koji se prosleđuju u drugim funkcijama nisu zabravo funkcije nego one-function objekti.

4.6. Funkcije mogu vratiti više argumenata

Jedna od prednosti Go jezika u odnosu na Javu, je ta što je u Go-u podržano da funkcije mogu vraćati više argumenata. Ovu karakteristiku je pre svega omogućila upotreba pokazivača.

4.7. Generics

Generics doprinosi kompleksnosti programa, mnogo konverzija i posla u runtime-u. Iz tog razloga Go ne podržava generics. Shodno tome, u Go jeziku je potrebno mnogo više koda da bi se predstavilo isto ponašanje za različite tipove podataka, nego što je to slučaj sa Javom.

4.8. Anotacije

Za razliku od Java jezika, Go jezik ne podržava anotacije. Ovo je jedna od loših strana Go programskog jezika. Anotacije u Javi su jednostavan i čitljiv način da prikažemo ponašanje za određene delove koda. Ovo direktno zači da svu logiku koje anotacije nose sa sobom u Go jeziku moramo da pišemo ručno, što zahteva dosta više posla i donosi mnogo više problema pri održavanju.

4.9. The profiler

Profiler alatke u Go-u su brže i jednostavnije za korišćenje. Pomažu korisniku da vidi na koji način je alocirana memorija i koliki je stepen iskorišćenosti CPU-a za sve delove programa. To se može ilustrovati putem grafa, pa samim tim je ekstremno olakšan posao ukoliko neki deo programa treba da se optimizuje. Java takođe sadrži profilere, kao što je Java Visual VM ali nisu jednostavni kao Go prifileri.

4.10. Overloading

Overloading nije podržan u programskom jeziku Go. Go zahteva da sve funkcije imaju unikatna imena. Praktično, u Javi možemo imati dve funkcije koje imaju isto ime, a različite tipove parametara i to će biti sasvim ispravno.

4.11. Ostalo

Još jedna od dobrih karakteristika je velika količina informacija koje korisnik može da dobije o ovom jeziku. Go od samog početka ima odličnu dokumentaciju i odlične preporuke na koji način je najbolje koristiti Go i kakav stil koristiti u kojim situacijama.

Go jezik ima veoma jednostavnu sintaksu, za razliku od Java jezika, korisnik ne mora da navodi ; da bi naglasio završetak linije. Sintaksa Go jezika se sastoji iz veoma malog broja ključnih reči.

Za razliku od Java jezika, Go ima ugradjenu podršku za kompleksne brojeve koja je jednostavna i intuitivna.

5. ZAKLJUČAK

Programski jezik Go je relativno nov jezik ali je zahvaljujući svojim dobrim karakteristikama uspeo da se zadrži u svetu programiranja. Jezik se konstantno unapređuje i razvija i svakom novom verzijom uvodi znatna poboljšanja u performansama. Jedna od najvećih prednosti Go jezika jeste brzina kompajliranja koja je mnogo bolja u odnosu na druge jezike.

Google je kreirao ogroman broj zvaničnih tutorijala i omogućio da Golang bude open-source tako da trenutno postoji ogromna količina dodataka i paketa. Ukoliko se radi sa ugradnim (embeded) sistemima koji raspolažu sa ograničenom memorijom onda Go nije najbolja opcija koja se može izabrati za implementaciju. Golang je odličan izbor za web aplikacije koje su danas sastavljene od velikog broja spoljnih servisa i baza podataka. To su aplikacije koje su najčešće organizovane pomoću arhitekture mikroservisa.

Popularnost Go jezika raste i smatra se da će Go jezik u narednih nekoliko godina biti svrstan u najpopularnije jezike za serverske aplikacije.

6. LITERATURA

[1] www.udemy.com/go-the-complete-developers-guide/
[2] medium.com/rungo/

Kratka biografija:

Milica Bučko, je rođena 02.09.1994. godine u Novom Sadu. Osnovnu školu „Vuk Karadžić“, u Bačkoj Palanci, završila je 2009. godine. Gimnaziju „20. oktobar“ u Bačkoj Palanci, završila je 2013. godine. Iste te godine upisala je Fakultet tehničkih nauka u Novom Sadu, smer Računarstvo i automatika. 2017. godine dobija zvanje Diplomirani inženjer elektrotehnike i računarstva. 2017/18. godine upisuje master akademske studije, smer „Primenjene računarske nauke i informatika – Elektronsko poslovanje”.
kontakt: milica.milica.b@gmail.com