

Go (Golang) programski jezik

1. Uvod

Go je programski jezik otvorenog koda razvijen 2009. godine od strane Google-a, odnosno od strane *Robert Griesemer-a*, *Rob Pike-a* i *Ken Thompson-a*. Go ima velike sličnosti sa C programskim jezikom koji poseduje *memory safety* i *garbage collection* mehanizme, pa ga mnogi neformalno nazivaju "modernim C-om". Go ima 3 bitne karakteristike kojeg ga razlikuje od ostalih programskih jezika:

- **Jednostavan dizajn** - ne postoji mehanizam rukovanje izuzecima niti *generic templates* koji komplikuju dizajn samog programskog jezika.
- **Direktna podrška za konkurentno programiranje (CSP)** - poseduje rutine (*goroutines*) koji omogućavaju konkurentno izvršavanje funkcija bez potrebe da se koristi bilo koja eksterna biblioteka (npr. *pthread*s u C-u).
- **Backward compatibility** - izmene koje bi narušile kompatibilnost prethodnih verzija se neće uvoditi u sam programski jezik. Ovo garantuje da se jednom napisan kod u Go programskom jeziku moći izvršavati dugi niz godina bez potrebe za bilo kakvim izmenama. Mnogi jezici ovo ne podržavaju:
 - *DEPRECATION: Python 3.5 reached the end of its life on September 13th, 2020. Please upgrade your Python as Python 3.5 is no longer maintained. pip 21.0 will drop support for Python 3.5 in January 2021. pip 21.0 will remove support for this functionality.*
 - *The version of Java (1.8.0_282) you have used to run this analysis is deprecated and we stopped accepting it. Please update to at least Java 11. You can find more information here:*
<https://sonarcloud.io/documentation/upcoming/>

2. Go programski jezik

2.1 Hello World from Go

```
package main // deklaracija paketa

import "fmt" // paket za standardni ulaz/izlaz

// ulazna tacka programa
func main() {
    fmt.Println("Hello World from Go")
}
```

Uvodne napomene:

- Ne postoji *null* vrednost, umesto toga je *nil*.
- **Promenljive koje su deklarisanе se moraju koristiti.**
- **Uvučeni (import-ovani) paketi se moraju koristiti.**
- Ne postoji ; na kraju iskaza
- Vežbanje online: <https://tour.golang.org/>

2.2 Deklaracija promenljivih

```
var a int // Neinicijalizovana int promenljiva ima vrednost 0

// Deklaracija i inicijalizacija
var b int = 5
var c, d int = 5, 6
s := "Hello World"
e := 5
```

Prosti tipovi podataka:

- bool
- string
- int int8 int16 int32 int64
- uint uint8 uint16 uint32 uint64 uintptr
- byte = uint8
- float32 float64
- complex64 complex128

2.3 Standardni ulaz/izlaz

- Koristi se paket **fmt**.
- Rad sa standardnim ulazom/izlazom sličan kao u jeziku C

```
fmt.Scanf("%d", &a) // citanje sa standardnog ulaza
fmt.Printf("Number is %d \n", c)
fmt.Println("Some text")
```

2.4 Naredbe grananja - if, switch

- **if, else if, else** rezervisane reči.
- **Ne postoji** ternarni operator: `var evenOrOdd = a % 2 == 0 ? "Even": "Odd"`.
- Nisu potrebne zagrade u *if* izrazu.
- **}** mora da bude u istom redu kao i **else**.

```

if b % 2 == 0 {
    fmt.Printf("%d is an even number",b)
} else { // } mora da bude u istom redu kao i else
    fmt.Printf("%d is an odd number",n)
}

```

```

fmt.Print("Go runs on ")
//os:=runtime.GOOS; se izvrši neposredno pre switch dela
switch os := runtime.GOOS; os {
    case "darwin":
        fmt.Println("OS X.")
    case "linux":
        fmt.Println("Linux.")
    default:
        fmt.Printf("%s.", os)
}

```

- Nije potreban *break* unutar *case*-a.
- *Case* izrazi ne moraju da budu konstante.

2.5 Naredbe ciklusa - for

- Nema zagrada kod izraza unutar **for** klauzule.
- for petlja:

```

for i := 0; i < 10; i++ {
    sum += i
}

```

- beskonačna for petlja:

```

for {
}

```

- Ne postoji while petlja ali se može postići pomoću for petlje

```

for sum < 10 {
    sum += i
}

```

2.6 Funkcije

```
func add(x int, y int) int {
    return x + y
}

func main() {
    var result = add(42, 13)
    var resultPlus5 = result + 5
    fmt.Printf("Result is %d, %d", result, resultPlus5)
}
```

Primer - faktorijel

```
func fact(n int) int {
    //} in the same line as else
    if n < 1 {
        return 1
    } else {
        return n * fact(n-1)
    }
}
```

Primer - prost broj

```
func isPrime(n int) bool {
    //6k + 1 || 6k -1 except 2 and 3
    if n < 2 {
        return false
    }
    for i := 2; i < n; i++ {
        if n%i == 0 {
            return false
        }
    }
    return true
}
```

Primer - closure

```
i := 42
f := func() {
    j := i/2 // ima pristup i
    fmt.Println(j)
}

f() // 21
```

```
func f() func() int {
    i := 0
    return func() int {
        i+=10
        return i
    }
}

func main() {
    a := f()
    b := f()

    fmt.Println(a()) // 10
    fmt.Println(b()) // 10

    a() // pozovi ponovo
    fmt.Println(a()) // 30
    fmt.Println(b()) // 20
}
```

Primer - currying

```
func multiply(a int) func(int) int {
    return func(i int) int {
        return a * i
    }
}

func main() {
    multiplyBy4 := multiply(4)
    fmt.Println("5 * 4: ", multiplyBy4(5))
}
```

2.7 Rukovanje fajlovima i greškama

```
os.Create(filePath)
file, err := os.OpenFile(filePath, os.O_RDWR, os.ModeAppend)
defer file.Close()
if err != nil {
    log.Printf("can't open file")
}
writer := bufio.NewWriter(file)
charsWritten, err := writer.WriteString("1. row\n")
if err != nil {
    log.Fatal(err)
}
err = writer.Flush()
```

2.8 Strukture

- Slično kao u C programskom jeziku.
- Koriste se za modelovanje stanja objekta.

```
type Person struct {
    firstName string
    lastName  string
    balance   float64
    personID  string
}
```

2.9 Pokazivaci

- Slično kao u C programskom jeziku.
- Tip podatka koji kao vrednost drži adresu.
- Za razliku od jezika C, Go nema pokazivačku aritmetiku.

```
func pointersExample() {
    i := 42
    p := &i // pokazivac na i
    fmt.Println(*p) // deferenciranje pokazivaca
    *p = 21 // postavljanje vrednosti i
    fmt.Println(*p)
    fmt.Println(i) // ispisi i
}
```

2.10 Staticki i dinamički nizovi

- Statički nizovi su fiksne veličine

```
var names [2]string
names[0] = "Marc"
names[1] = "John "
fmt.Println(names[0], names[1])
fmt.Println(names)
```

- Statički nizovi nisu uvek pogodni jer se veličina mora znati unapred. Zbog toga se koriste dinamički nizovi, *slice* - identičan listi u drugim programskim jezicima.

```
slice := make([]int, 5) // dinamički kreiraj slice
fmt.Println(slice) // [0 0 0 0 0]
slice = append(slice, 3)
fmt.Println(slice) // [0 0 0 0 0 3]
```

2.11 Interfejs

- Interfejsi predstavljaju imenovane kolekcije potpisa metoda.
- Analogni su interfejsima u drugim jezicima.
- Dodavanje objektno paradigme na Golang.
- Ključna reč **interface**.
- Interfejs se ne može da naslediti drugi interfejs. Međutim moguće je kombinovati interfejse i od dva interfejsa napraviti novi koji sadrži sve metode interfejsa od kojih se sastoji.
- Omogućavaju **duck typing**, (odnosno **structural typing**).

```
type Osoba interface {
    toString() string
}

type Student struct {
    ime, prz, brIndeksa string
}

type Radnik struct {
    ime, prz, jmbg string
}

func (s Student) toString() string {
    return "Student[" + s.ime + " , " + s.prz + " , " + s.brIndeksa + "]"
}

func (r Radnik) toString() string {
    return "Radnik[" + r.ime + " , " + r.prz + " , " + r.jmbg + "]"
}
```

```
func display(o Osoba) {
    fmt.Println(o.toString())
}
func main() {
    s := Student{"marko", "markovic", "ee-222/2012"}
    r := Radnik{"rastko", "raicevic", "055121312321312"}
    display(s)
    display(r)
}
```

2.11 Konkurencija

- Funkcije se mogu izvršavati konkurentno sa drugim funkcijama koristeći go rutine (goroutines).
- Go rutine su lakše od niti ([link](#)). Sinronizacija između rutina se vrši pomoću kanala.
- Pozivaju se pomoću prefiksa **go**.
- Main funkcija se izvršava u sopstvenoj Go rutini koja se zove *main Goroutine*.
- Pokretanje Go rutine odmah vraća kontrolu pozivaocu
 - Ne čeka se kraj izvršavanja Go rutine.
 - Sve povratne vrednosti Go rutine se ignorišu.
- Main Go rutina mora biti pokrenuta da bi se bilo koja druga Go rutina izvršavala.
- Prekid *main* Go rutine će prekinuti izvršavanje svih ostalih Go rutina.

```
func hello(from string) {
    for i := 1; i < 1000000000; i++ {
    }
    fmt.Println("Hello from : " + from)
}
func main() {
    hello("program")
    go hello("Go routine")
    go func(caller string) {
        fmt.Println("Anonymous f: called by " + caller)
    }("Go routine")
    fmt.Scanln()
}
```

- Kanali predstavlja mehanizam pomoću kojeg komuniciraju Go rutine.

- Kanal u suštini predstavlja strukturu koja funkcioniše po FIFO principu (nešto poput reda).
- Svaki kanal ima tip podatka koji mu je pridružen
 - To ograničava kanal na tip podatka kojim se može komunicirati unutar njega
 - `a := make(chan int)` - kreiranje kanala u kojem je moguće smeštati promenljive tipa `int`.
 - `data := <- a` - čitanje iz kanala
 - `a <- data` - pisanje u kanal

```
func sumArrayIntoChannel(s []int, c chan int, ordNum int) {
    fmt.Printf("Started routine %d\n", ordNum)
    sum := 0
    for _, v := range s {
        sum += v
    }
    c <- sum // send sum to c
    fmt.Printf("Finished routine %d\n", ordNum)
}

func main() {
    s := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10,
              11, 12, 13, 14, 15, 16, 17, 18, 19, 20}

    c := make(chan int)
    go sumArrayIntoChannel(s[:5], c, 1)
    go sumArrayIntoChannel(s[5:10], c, 2)
    go sumArrayIntoChannel(s[10:15], c, 3)
    go sumArrayIntoChannel(s[15:], c, 4)
    x, y, z, v := <-c, <-c, <-c, <-c // citaj iz kanala
    fmt.Println(x, y, z, v, x+y+z+v)
}
```

3. Dodatni Materijali

- <http://www.igordejanovic.net/courses/tech/GoLang/index.html>

3.1 Obnavljanje JetBrains licence

- Neophodno je ulogovati/registrovati se na svoj *uns.ac.rs* nalog.
- Odabrati opciju *Renew my Licence Pack*.

The screenshot shows the JetBrains account interface. At the top, there's a navigation bar with 'Licenses' and 'Your Account' tabs. The user is logged in as 'Danijel Radakovic'. The main content area is titled '1 License' and displays details for the 'JetBrains Product Pack for Students'. The license is issued to 'Danijel Radakovic' for educational use only, valid through 'October 16, 2020'. A list of included products is shown: AppCode, CLion, DataGrip, dotCover, dotMemory, dotTrace, GoLand, IntelliJ IDEA Ultimate, PhpStorm, PyCharm, ReSharper, ReSharper C++, Rider, and RubyMine. A 'Buy new license' button is in the top right. Below the license details, there's a note about signing in after installation and a link to find the license key.

- Popuniti odgovarajucu formu za produženje licence.

JetBrains Products for Learning

Before you apply, please read the [Educational Subscription Terms and FAQ](#).

The screenshot shows the 'JetBrains Products for Learning' application form. It has several sections: 'Apply with:' with options 'UNIVERSITY EMAIL ADDRESS', 'ISIC/ITC MEMBERSHIP', 'OFFICIAL DOCUMENT', and 'GITHUB'; 'Status:' with radio buttons for 'I'm a student' and 'I'm a teacher' (selected); 'Email address:' with the input 'danijelradakovic@uns.ac.rs'; a certification statement 'I certify that the university email address provided above is valid and belongs to me.'; 'Country / region:' with a dropdown menu showing 'Serbia'; two checkboxes for 'I have read and I accept the JetBrains Account Agreement' and 'I consent to the use of my name, email address, and location data in email communication concerning JetBrains products held or services used by me or my organization.'; and a blue 'APPLY FOR FREE PRODUCTS' button at the bottom.

- Skinuti i instalirati GoLand softver.