Alati za razvoj softvera

Programski jezik Go (golang)

Univerzitet u Novom Sadu
Fakultet Tehničkih Nauka

Opšte informacije o jeziku

- 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 slličnosti sa programskim jezikom C, poseduje *memory safety* i *garbage collection* mehanizme, pa ga mnogi neformalno nazivaju "modernim C-om"
- Go ima 3 bitne karakteristike koje ga razlikuje od ostalih programksihg jezika:
  1. **Jednostavan dizajn** — ne postoji mehanizam rukovanje izuzecima niti *generic templates* koji komplikuju dizajn samog programskog jezika (u poslednjoj verziji dodata je minimalna podrška za generics)
  2. **Direktna podrška za konkurentno programiranje** CSP — poseduje rutine (goroutines) koji omogućuvaju konkuretno izvršavanje funkcija bez potrebe da se koristi bilo kakve eksterna biblioteka (npr. pthreads u c-u)
  3. **Backward compatibility** — izmene koje bi narušile kompatibilnost prethodnih verzija se neće uvoditi u sam programki jezik

- Treća stavka nam garantuje da se jednom napisan kod u go programksu 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 Please upgrade your Python as Python 3.5 is no longer maintained.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

Hello world

```
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 deklarisane se moraju koristiti**
- **Uvučeni (import-ovani) paketi se moraju koristiti**
- Ne postoji ; na kraju iskaza
- **Golang nije objektno-orijentisan jezik**
- Vežbanje online

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

Standardni ulaz/izlaz

- Koristi se standardni paket **fmt**
- Rad sa standardnim ulazom/izlazom sličan kao u jeziku C (i u srodnim jezicima)

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

Naredbe grananja — if

- if, else if, else rezervisane reči
- Ne postoji ternarni operator: *var evenOrOdd = a % 2 == 0 ? "Even" : "Odd"*
- Nisu potrebne zagrade u *if* iskazu
- } 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)
}
```

Naredbe grananja — switch

- switch, case, default rezervisane reči
- Nisu potrebne zagrade u *switch* iskazu
- Može da sadrži više elemenata u case razdvojenih zarezom
- Mogućnost kombinovanja dodele vrednosti i provere vrednosti
- Nije potreban *break* unutar case-a
- Case izrazi ne moraju da budu konstante

```
fmt.Print("Go runs on ")
//os:=runtime.GOOS; se izvrsi 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)
}
```

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

Naredbe ciklusa — for

- Nema zagrada kod izraza unutar *for* klauzule
- Ne postoji while petlja ali se može postići pomoću for petlje
- Sadrži for-each petlju za iteraciju kroz kolekcije/nizove (for-range)
- Podržava *break* i *continue* kao i ostali programski jezici

```
//standardna for petlja
for i := 0; i < 10; i++ {
        sum += i
}

// while petlja
for sum < 10 {
sum += i
}

//for-each / for-range
strings := []string{"hello", "world"}
for i, s := range strings {
        fmt.Println(i, s)
}

sum := 0
for i := 1; i < 5; i++ {
        if i%2 != 0 {
                continue // pocinje novu iteraciju
        }
        sum += i
}
```

Funkcije

- Funkcionišu identično kao i u svim drugim strogo tipiziranim jezicima
- Broj parametara, pozicija i njihovi tipovi se moraju poklapati prilkom poziva
- Go omogućava vraćavanje više vrednosti kao rezultat
- Omogućava povratak funkcije kao povratnu vrednost
- Omogućava kreiranje neimenovanih funkcija — closure

```
package main

import "fmt"

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)
}
```

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

// povratna vrednost
func isPrime(n int) bool {
        if n < 2 {
                return false
        }

        for i := 2; i < n; i++ {
                if n%i == 0 {
                        return false
                }
        }
        return true
}
```

```
package main

import "fmt"

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
}
```

Rukovanje greškama

- Go nema mehanizam za hvatanje izuzetaka
- try-catch mehanizam ne postoji
- Go vraća dodatan parametar prilikom poziva funkcije
- Taj dodatan parametar nam kaže ima ili nema greške
- Istu strategiju koristimo i mi kada pravimo naše funkcije
- Onaj ko poziva funkciju koja vraća grešku, treba da proveri da li je do greške doslo
- Korisnik može da vrati greške svojstvene svojim funkcijama — errors paket

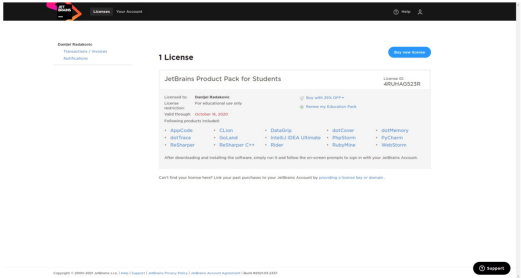
```go
package main

import (
        "fmt"
        "errors"
)

func e(v int) (int, error) {
        if v == 0 {
                return 0, errors.New("Zero cannot be used")
        } else {
                return 2*v, nil
        }
}

func main() {
        v, err := e(0)

        if err != nil {
                fmt.Println(err, v)       // Zero cannot be used 0
        }
}
```

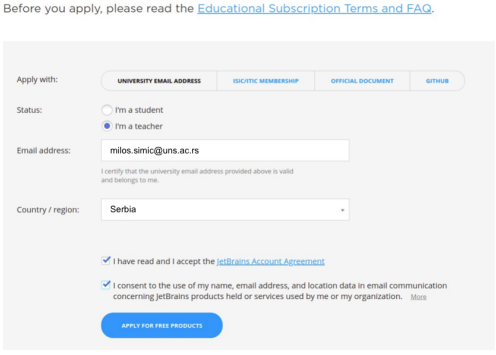
Dodatni materijali

- Instalacija Go-a na raznim platformama
- Postoji nekoliko IDE-a ili editora koje možete koristiti Emacs, Spacemacs, Golad, VSCode, ...
- Tour of Go
- Instalacija Go-a i VSCode plugin-a

Goland i Obnavljanje JetBrains licence

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





Skinuti i instalirati GoLand softver

Kraj predavanja

Pitanja? :)