



PROGRAMSKI JEZIK GO

ZDRAVO SVIMA !

OSNOVNE OSOBINE

- ❖ Statički tipiziran, kompajliran programski jezik
- ❖ Programski jezik otvorenog koda
- ❖ Razvijen u Google-u 2007. godine Robert Griesmer, Rob Pike i Ken Thompson
- ❖ Sličan C programskom jeziku, ali pruža memory safety, ima garbage collection i direktnu podršku za konkurentno programiranje (Communicating Sequential Process (CSP))

1.

WORKSPACE

Let's start with the first set of slides

OBIČNO SE SAV GO KOD SMEŠTA NA JENDO
MESTO KOJE ZOVEMO WORKSAPCE
NA OVOM MESTU SE NALAZI VEĆI BROJ
REPOZITORIJUMA POD SISTEMIMA ZA
KONTROLU VERZIJA (NPR. GIT)

SVAKI REPOZITORIJUM SE SASTOJI OD
JEDNOG ILI VIŠE PAKETA
SVAKI PAKET SE SASTOJI OD JEDNOG ILI
VIŠE GO FAJLOVA U JEDNOM
DIREKTORIJUMU

PUTANJA (PATH) DO DIREKTORIJUMA
PAKETA ODREĐUJE NJEGOV IMPORT PATH

Workspaces

```
bin/
    hello                # command executable
    outyet               # command executable
src/
    github.com/golang/example/
        .git/            # Git repository metadata
    hello/
        hello.go         # command source
    outyet/
        main.go           # command source
        main_test.go      # test source
    stringutil/
        reverse.go        # package source
        reverse_test.go   # test source
    golang.org/x/image/
        .git/            # Git repository metadata
bmp/
    reader.go            # package source
    writer.go            # package source
... (many more repositories and packages omitted) ...
```


GOPATH

- x Podrazumevano \$HOME/go na Unix sistemima ili %USERPROFILE%\go na Windows-u.
- x go env GOPATH komanda daje info. o tekućoj lokaciji tj. Sadržaj GOPATH varijable ili podrazumevanu lokaciju ukoliko nije podešena.



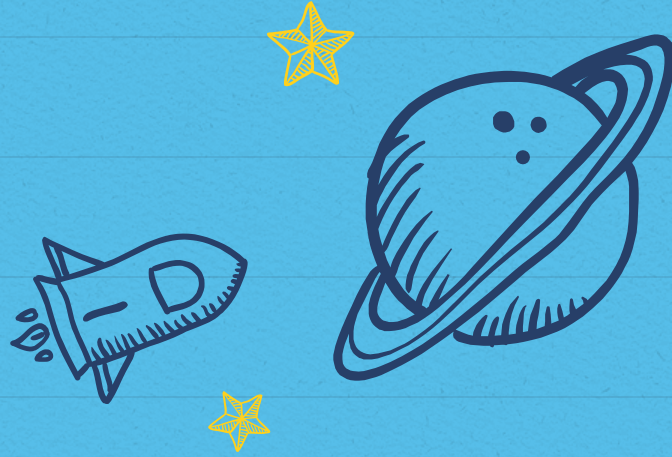
- x Da bi instalirani Go programi bili dostupni:
- x `$ export PATH=$PATH:$(go env GOPATH)/bin`



IMPORT PUTANJA (IMPORT PATH)

- ✗ Kratke putanje za standardnu biblioteku. Na primer: `fmt`, `net/http`
- ✗ Putanja je relativna u odnosu na `$GOPATH/src`
- ✗ Mora biti jedinstvena





PRVI GO PROGRAM


```
$MKDIR $GOPATH/SRC/GITHUB.COM/USER/HELLO
```

Fajl hello.go

package main

```
import "fmt"
```

```
func main(){
```

```
fmt.Printf("Zdravo svete!\n")
```

}

go install

github.com/user/hello

iii

cd

```
$GOPATH/src/github.com/user/hello
```

IMENOVANJE PAKETA

X package name
je ime paketa koje
se koristi
pri *import-u*

X Svi fajlovi koji
pripadaju istom
paketu moraju
koristiti isto ime

X Izvršne komande
moraju biti u
paketu main

X Go konvencija je
da je ime paketa
poslednji element
import putanje.



VARIABLE


var iskaz definiše
varijable. Tip se navodi
na kraju

var se takođe može
koristiti i na nivou paketa
ili funkcije



```
package main  
  
import "fmt"  
  
var c, python, java bool  
  
func main{  
var i int  
fmt.Println(i, c, python, java)  
}
```





LOVE TO
LEARN

KRATKA DEKLARACIJA VARIJABLI

x Unutar funkcija, kraći oblik deklaracije baziran na `:=` dodeli može se koristiti

```
package main
```

```
import "fmt"
```

```
func main(){
```

```
var i, j int = 1, 2
```

```
k := 3
```

```
c, python, java := true, false, "no!"
```

```
}
```



OSNOVNI TIPOVI

- x bool
- x string
- x int int8 int16 int32 int64
- x uint uint8 uint16 uint32 uint64 uintptr
- x byte // alias for unit8
- x float32 float64
- x complex64 complex128



NULTE VREDNOSTI

- x Varijable deklarirane brez inicijalizatorja se inicijalizirajo na podrazumevane nulte vrednosti:
- x 0 za numeričke tipe
- x false za boolean tip
- x "" (prazen string) za string tip



KONVERZIJA TIPOVA

x Izraz oblika $T(v)$ konvertuje vrednost v u tip T

var i int = 42

var f float64 = float64(i)

var u uint = uint(f)

i := 42

f := float64(i)

u := uint(f)



PRIMER

```
package main

import (
    "fmt"
    "math"
)

func main()
{
    var x, y int = 3, 4

    var f float64 = math.Sqrt(float64(x*x + y*y))

    var z uint = uint(f)
    fmt.Println(x, y, z)
}
```


KONSTANTE

Deklarisane kao varijable ali upotrebom ključne reči `const`

MOGU BITI OSNOVNIH TIPOVA

Ne mogu se definisati upotrebom `:=`

100% JE TAKO :D



PRIMER

```
package main import "fmt"

const Pi = 3.14

func main() {
    const World = "世界"
    fmt.Println("Hello", World)
    fmt.Println("Happy", Pi, "Day")
    const Truth = true
    fmt.Println("Go rules?", Truth)
```



ENUMERISANE KONSTANTE (IOTA)

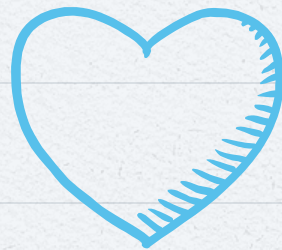
```
const (
    CategoryBooks = iota // 0
    CategoryHealth // 1
    CategoryClothing // 2
)
```

ILI U KOMBINACIJI SA TIPOM

```
type Stereotype int
const (
    TypicalNoob Stereotype = iota // 0
    TypicalHipster // 1
    TypicalUnixWizard // 2
    TypicalStartupFounder // 3
)
```

ISKAZI KONTROLE

TOKA: FOR, IF, ELSE, SWITCH I DEFER



FOR ISKAZ

- x Go ima samo jedan iskaz za petlje – for
- Tri komponente
- x *init iskaz* – izvršava se pre prve iteracije
- x *uslov* – evaluira se pre svake iteracije i u zavisnosti od rezultata ciklus se izvršava ili se petlja prekida
- x *post iskaz* – izvršava se na kraju svake iteracije



FOR

```
package main  
  
import "fmt"  
  
func main() {  
    sum := 0  
    for i := 0; i < 10; i++  
    {  
        sum += i  
    }  
    fmt.Println(sum)  
}
```



PRESENTATION DESIGN

This presentation uses the following typographies:

- ✗ Titles: Amatic SC
- ✗ Body copy: Quicksand

Download for free at:

<https://www.fontsquirrel.com/fonts/amatic>

<https://www.fontsquirrel.com/fonts/quicksand>

You don't need to keep this slide in your presentation. It's only here to serve you as a design guide if you need to create new slides or download the fonts to edit the presentation in PowerPoint®

"BESKONAČNA" PETLJA

```
package main
```

```
func main()
```

```
{
```

```
    for {
```

```
    }
```

```
}
```



SWITCH ISKAZ

- x Kraći način pisanja sekvence if/else iskaza.
Izvršava prvi case blok gde je vrednost jednaka vrednošću izraza uslova
- x Može imati kratak iskaz kao i if



SWITCH

```
package main

import (
    "fmt"
    "runtime"
)

func main() {
    fmt.Print("Go runs on ")
    switch os := runtime.GOOS; os {
    case "darwin":
        fmt.Println("OS X.")
    case "linux":
        fmt.Println("Linux.")
    default:
        // freebsd, openbsd,
        // plan9, windows...
        fmt.Printf("%s.\n", os)
    }
}
```


2.

STRUCT, SLICE I MAP

POKAZIVAČI (POINTERS)

- x Memorijska adresa vrednosti variјable
- x *T je pokazivač na vrednost tipa T
- x & operator vraća pokazivač na zadati argument/vrednost
- x i := 42 p = &i // *p je pokazivač na vrednost 42*
- x * operator označava vrednost na koju pokazivač pokazuje

STRUKTURE (STRUCT)

```
package main
```

```
import "fmt"
```

```
type Vertex struct { X int Y int }
```

```
func main() { fmt.Println(Vertex{1, 2}) }
```

Pokazivači na strukture

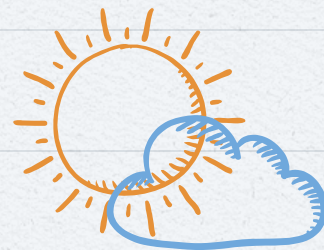
- Poljima strukture se može pristupiti preko pokazivača na strukturu
- Sintaksno, ako imamo pokazivač `p` na strukturu, polju `x` bi mogli pristupiti sa `(*p).x`
- Pošto je ovakva sintaksa teža za korišćenje uvedena je prečica `p.x` tj. nije potrebno eksplicitno dereferenciranje

```
package main

import "fmt"

type Vertex struct {
    X int
    Y int
}

func main() {
    v := Vertex{1, 2}
    p := &v
    p.X = 1e9
    fmt.Println(v)
}
```

NIZOVI (ARRAYS)

- ✗ $[n]T$ – niz od n elemenata tipa T
- ✗ `var a [10]int` – niz od 10 elemenata tipa `int`
- ✗ Nizovi su fiksne veličine



ISEČCI (SLICES)

- Niz je fiksne dužine
- Isečak je "prozor" na niz koji ima dinamičku veličinu
- `[]T` – isečak tipa `T`
- Isečak se formira iznad niza na sledeći način:

```
a[low : high]
```

- Ovo formira polu-otvoren interval elemenata uključujući prvi ali isključujući poslednji

```
package main

import "fmt"

func main() {
    primes := []int{2, 3, 5, 7, 11, 13}

    var s []int = primes[1:4]
    fmt.Println(s)
}
```



KREIRANJE ISEČKA SA MAKE

- **make** funkcija alocira niz sa nultim vrednostima i vraća njegov isečak

```
a := make([]int, 5) // len(a)=5
```

- moguće je definisati i kapacitet

```
b := make([]int, 0, 5) // Len(b)=0, cap(b)=5
```

```
b = b[:cap(b)] // len(b)=5, cap(b)=5
```

```
b = b[1:] // len(b)=4, cap(b)=4
```


MAPE

- Mapiranje ključeva na vrednosti – asocijativni niz
- Nula vrednost je `nil`
- `nil` mapa nema ključeve niti se ključevi mogu dodati
- `make` funkcija vraća inicijalizovanu mapu spremnu za upotrebu

```
package main

import "fmt"

type Vertex struct {
    Lat, Long float64
}

var m map[string]Vertex

func main() {
    m = make(map[string]Vertex)
    m["Bell Labs"] = Vertex{
        40.68433, -74.39967,
    }
    fmt.Println(m["Bell Labs"])
}
```

IZMENA VREDNOSTI MAPA

X Postavljanje vrednosti

```
m[key] = elem
```

x Čitnaje vrednosti

```
elem = m[key]
```

- brisanje

```
delete(m, key)
```

- čitanje i provera da li element postoji

```
elem, ok = m[key]
```

Ako element postoji **ok** će imati vrednost **true** inače **false**



3.

METODE I INTERFEJSI

METODE I INTERFEJSI

- x Go nema klase ali se mogu definisati metode nad tipovima
- x Metoda je funkcija koja ima specijalni *receiver* parametar



Metode su funkcije

- Metode se u osnovi ponašaju kao obične funkcije

```
package main

import (
    "fmt"
    "math"
)

type Vertex struct {
    X, Y float64
}

func Abs(v Vertex) float64 {
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}

func main() {
    v := Vertex{3, 4}
    fmt.Println(Abs(v))
}
```

Pokazivači i funkcije

```
package main

import (
    "fmt"
    "math"
)

type Vertex struct {
    X, Y float64
}

func Abs(v Vertex) float64 {
    return math.Sqrt(v.X*v.X + v.Y*v.Y)
}

func Scale(v *Vertex, f float64) {    // ako uklonimo '*'?
    v.X = v.X * f
    v.Y = v.Y * f
}

func main() {
    v := Vertex{3, 4}
    Scale(&v, 10)
    fmt.Println(Abs(v))
}
```


INTERFEJSI

- x Interfejs tip je definisan skupom signatura metoda
- x Kažemo da tip implementira interfefs ako implementira skup metoda koje interfefs definiše (implicitno)
- x Varijabla tipa interfejsa može sadžati bilo koju vrednost koja implementira dati skup metoda interfejsa

- X [Go Documentation](#)
- X [A Tour of Go](#)
- X [How to Write Go Code](#)
- X [The Go Wiki](#)
- X [Effective Go](#)



POZDRAV!
LUKA PETROVIĆ