NTP 2020/2021.

Vežbe 8, 9. i 10.

U osmom, devetom i desetom terminu vežbi studenti bi trebalo da se upoznaju sa programskim jezikom *Go*[1], samostalno proučavajući dolenavedene materijale:

- Programski jezik *Go* - prezentacija
- *Go* video-predavanje 1
- *Go* video-predavanje 2
- *Go* video-predavanje 3
- Instalacija *Go* okruženja.
- *A tour of Go* – interaktivni *tutorial*
- *How to Write Go Code* – konvencije i smernice za pisanje i organizaciju *Go* koda.

Zadaci za samostalni rad

Nakon prolaska kroz gorenavedene materijale, ali i uz dodatno samostalno istraživanje, potrebno je uraditi sledeće zadatke (teorijske i praktične):

Teorijska pitanja

1. Da li je programski jezik *Go* kompajliran ili interpretiran programski jezik? Objasniti.
2. Da li je programski jezik *Go* statički ili dinamički tipiziran programski jezik? Objasniti.
3. Objasniti pojmove radni prostor (engl. *workspace*), repozitorijum, modul, paket i izvorna datoteka u kontekstu programskog jezika *Go*.
4. Koja dva tipa izvornih datoteka postoje? Koje su bitne razlike između njih?
5. Kako se gradi *import* prefiks za pakete koji pripadaju standardnoj biblioteci, a kako za pakete koji joj ne pripadaju?
6. Kako su regulisana prava pristupa funkcijama izvan paketa kojima pripadaju?
7. Objasniti postupak pisanja i pokretanja jediničnih testova?
8. Kako je realizovano upravljanje memorijom (engl. *memory management*).
9. Da li programsk jezik *Go* nativno podržava konkurentno programiranje? Objasniti.
10. Šta mora biti prvi iskaz u izvornoj datoteci?
11. Gde se smeštaju izvršne komande nakon instalacije?
12. Šta je ulazna tačka u *Go* program?
13. Šta je naziv, a šta (import) putanja paketa koji ne pripada standardnoj biblioteci? Da li naziv paketa mora biti jedinstven? A putanja paketa?
14. Koliko povratnih vrednosti može da vrati funkcija?
15. Šta su to imenovane povratne vrednosti? Koja je njihova najbitnija uloga? Kada ih je zgodno koristiti, a kada se ne preporučuje njihova upotreba?
16. Šta je *return* iskaz bez argumenata (engl. *naked return*)? Kada se najčešće koristi?
17. Na koja dva načina je moguće deklarisati novu promenljivu? Po čemu se razlikuju?
18. Koja je uloga operatora `:=` u programskom jeziku *Go*? Da li se može koristiti van funkcije? Obrazložiti odgovor.
19. Šta su to nulte vrednosti (engl. *zero values*)?
20. Da li je iskaz `var a int32 = uint32(55)` validan? Obrazložiti odgovor.
21. Šta je inferencija tipova (engl. *type inference*)?

1 U literaturi poznat i kao *Golang*.

22. Na osnovu sledećeg isečka koda potrebno je odrediti tip za promenljive `i`, `j`, `k`, `f` i `g`

```
1  package main
2
3  import (
4        "fmt"
5  )
6
7  func main() {
8        // package, imports, main, etc.
9        var i int
10       j := i
11       k := 42
12       f := 3.142
13       g := 0.867 + 0.5i
14       fmt.Println(i, j, k, f, g)
15 }
```

23. Kako se definišu konstante? Da li je moguća upotreba operatora **:=** prilikom definicije konstanti? Kog sve tipa mogu biti konstatne? Kog su tipa konstante deklarisane bez ekplicitnog navođenja[2] tipa?
24. Šta će biti prikazano na standardnom izlazu nakon izvršavanja sledećeg isečka koda? Zašto?

```
1  package main
2
3  import (
4        "fmt"
5  )
6
7  const Huge = 1e1000
8
9  func main() {
10       fmt.Println(Huge / 1e999)
11       fmt.Println(Huge)
12 }
```

25. Šta je *factored import*? Navesti nekoliko primera njegove upotrebe.
26. Dolenavedeni isečak koda ima jednu grešku. Potrebno je pronaći i ispraviti.

```
1  package main
2
3  import (
4        "fmt"
5        "math"
6  )
7
8  func main() {
9        fmt.Println(math.pi)
10 }
```

27. Kolika je širina[3] sledećih tipova podataka `int`, `uint` i `uintptr`?

2 Npr. konstanta `Huge` iz primera 24.

3 izražena u bitovima

28. Dolenavedeni isečak koda ima nekoliko grešaka. Potrebno ih je pronaći i ispraviti.

```
 1 package main
 2
 3 import (
 4       "fmt"
 5       "math"
 6       "strings"
 7 )
 8
 9 func main() {
10       var a int = 23
11       var x, y int = 3, 4
12       var f float64 = math.Sqrt(float64(x*x + y*y))
13       var z uint = f
14       fmt.Println(x, y, z)
15 }
```

29. Kako se u programskom jeziku *Go* navodi `while` petlja?

30. Potrebno je pronaći i ispraviti sve greške u navedenom isečku koda. Nakon toga odrediti doseg (engl. *scope*) važenja promenljive `v`.

```
 1 package main
 2
 3 import (
 4       "fmt"
 5       "math"
 6 )
 7
 8 func pow(x, n, lim float64) float64 {
 9       if v := math.Pow(x, n); v < lim {
10             return v
11       }
12       else
13             fmt.Printf("%g >= %g\n", v, lim)
14       return lim
15 }
16
17 a := 33
18
19 func main() {
20       fmt.Println(
21             pow(3, 2, 10),
22             pow(3, 3, 20),
23       )
24       fmt.Println(a)
25 }
```

31. Šta će biti prikazano na standardnom izlazu nakon izvršavanja sledećeg isečka koda? Zašto?

```
1  package main
2
3  import "fmt"
4
5  func giveMeOne() int {
6        fmt.Println("Returning 1")
7        return 1
8  }
9
10 func giveMeTwo() int {
11       fmt.Println("Returning 2")
12       return 2
13 }
14
15 func main() {
16       i := 1
17       switch i {
18       case giveMeOne():
19             fmt.Println("1 :)")
20       case giveMeTwo():
21             fmt.Println("2 :)")
22       default:
23             fmt.Printf("Something else: %d", i)
24       }
25 }
```

32. Upotrebom *switch-case* konstrukcije napisati programski kod ekvivalentan datom kodu.

```
1  package main
2
3  import (
4        "fmt"
5        "time"
6  )
7
8  func main() {
9        t := time.Now()
10       if t.Hour() < 12 {
11             fmt.Println("Good morning!")
12       } else if t.Hour() < 17 {
13             fmt.Println("Good afternoon.")
14       } else {
15             fmt.Println("Good evening.")
16       }
17 }
```

33. Koja je uloga `defer` iskaza u programskom jeziku *Go*? Kada se određuju argumenti funkcije ispred čijeg poziva je navedena ključna reč `defer`? U kom redosledu se izvršavaju višestruki `defer` iskazi navedeni unutar iste funkcije?
34. Šta je pokazivač u programskom jeziku *Go*?
35. Kako je realizovana pokazivačka aritmetika u programskom jeziku *Go*?
36. Šta je dereferenciranje pokazivača? Navesti primer.
37. Da li je eksplicitno dereferencinje pokazivača uvek obavezno? Objasniti kroz nekoliko primera.

38. U čemu je razlika između operatora **&** i ***** ?
39. U čemu je razlika između niza (engl. *array*) i isečka (engl. *slice*) u programskom jeziku *Go*?
40. Šta je dužina, a šta kapacitet isečka?
41. Šta predstavlja potporni niz nekog isečka?
42. Objasniti princip rada funkcije `append`?
43. Za šta se koristi funkcija `make`?
44. Šta je nulta vrednost za isečak?
45. Odrediti dužinu, kapacitet i potporni niz isečaka `s1`, `s2`, `s3`, `s4` i `s5` za vreme izvršavanja funkcije `showSlices`.

```
1  package main
2
3  import (
4        "fmt"
5  )
6
7  func showSlices(s1, s2, s3, s4, s5 []int) {
8        fmt.Println(s1, s2, s3, s4, s5)
9  }
10
11 func main() {
12       var a []int
13       var b = make([]int, 5)
14       var c = make([]int, 0, 5)
15       d := b[:2]
16       e := b[1:]
17       e[0] = 23
18       d = append(d, 33)
19       showSlices(a, b, c, d, e)
20 }
```

46. Čemu služi znak **_** u programskom jeziku *Go*? Navesti nekoliko primera upotrebe.
47. Šta je nulta vrednost za mapu?
48. Šta će biti prikazano na standardnom izlazu nakon izvršavanja sledećeg isečka koda? Zašto?

```
1  package main
2
3  import "fmt"
4
5  func main() {
6        var m map[int]float64
7        m = make(map[int]float64)
8        m[1] = float64(2)
9        fmt.Println(m[1], m[2], m[3])
10 }
```

49. Da li su funkcije *first-class* objekti u programskom jeziku *Go*? Objasniti.
50. Da li su funkcije *first-class* objektii u programskom jeziku *C*? Objasniti.
51. Da li programski jezik *Go* podržava leksička zatvorenja? Ukoliko podržava, potrebno je navesti nekoliko primera. Ukoliko pak ne podržava, potrebno je dati objašnjenje zbog čega.
52. Kako su klase realizovane u programskom jeziku *Go*?
53. Šta su metode u programskom jeziku *Go*? Navesti nekoliko primera.
54. Nad kojim tipovima je moguće definisati metode?

55. Koja su dva najbitnija razloga za upotrebu pokazivačkih prijemnika (engl. *pointer receivers*) u odnosu na vrednosne prijemnike (engl. *value receivers*)?
56. Šta će biti prikazano na standardnom izlazu nakon izvršavanja sledećeg isečka koda? Zašto?

```
1  package main
2
3  import "fmt"
4
5  type MyInt int
6
7  func ShowMe(num MyInt) {
8        fmt.Println(num)
9  }
10
11 func (num MyInt) ShowMe() {
12       fmt.Println(num)
13 }
14
15 func main() {
16       val := MyInt(33)
17       ptr := &val
18       ptr.ShowMe()
19       ShowMe(ptr)
20 }
```

57. Šta je interfejs u programskom jeziku *Go*? Navesti nekoliko primera.
58. Da li se interfejs implementira implicitno ili eksplicitno? Objasniti.
59. Da li će sledeći isečak koda izazvati grešku, zašto?

```
1  package main
2
3  import "fmt"
4
5  type I interface {
6        M()
7  }
8
9  type T struct {
10       S string
11 }
12
13 func (t T) M() {
14       fmt.Println(t.S)
15 }
16
17 func main() {
18       t := T{"hello"}
19       var i I = &t
20       i.M()
21 }
```

60. Šta je interfejs vrednosti (engl. *interface values*)?
61. Šta je *nil* interfejs?
62. Da li prijemnik metode može imati `nil` vrednost? Da li *nil* interfejs može biti prijemnik metode? Oba odgovora obrazložiti i potkrepiti odgovarajućim primerima.

63. Šta je prazan interfejs (engl. *empty interface*) i za šta se najčešće koristi?
64. Pronaći i ispraviti grešku u dolenavedenom isečku koda.

```
1  package main
2
3  import "fmt"
4
5  func main() {
6        var i interface{} = "hello"
7        s := i.(string)
8        fmt.Println(s)
9        s, ok := i.(string)
10       fmt.Println(s, ok)
11       f, ok := i.(float64)
12       fmt.Println(f, ok)
13       f = i.(float64)
14       fmt.Println(f)
15 }
```

65. Šta je `Stringer` interfejs i čemu služi? Navesti nekoliko primera njegove upotrebe.
66. Kako su realizovani izuzeci u programskom jeziku *Go*?
67. Na koji način funkcija može prijaviti da se tokom njenog izvršavanja desila greška? Navesti nekoliko primera.
68. Šta je tok (engl. *stream*) podataka?
69. Na koji način je moguće iščitati sadržaj tekstualne datoteke? Koji interfejs se tom prilikom koristi?
70. Na koje načine se prosleđuju argumenti prilikom poziva funkcije u programskom jeziku *Go*?
71. Šta je *Go* rutina (engl. *goroutine*)?
72. Na koje sve načine je moguće izvršiti sinhronizaciju *Go* rutina?
73. Šta su i čemu služe kanali u programskom jeziku *Go*?
74. Šta su baferovani kanali i kako se kreiraju?
75. Da li je obavezno zatvaranje kanala? A tokova?
76. Kada se najčešće zatvaraju kanali?
77. Na koje načine se može obezbediti ekskluzivan pristup nekom resursu?

Praktični zadaci

78. Navesti primer beskonačne petlje u programskom jeziku *Go*.

79. Implementirati funkciju `Sqrt` čiji je zadatak da primenom Njutn-Rapsonove[4] metode izračuna vrednost kvadratnog korena realnog broja, koji prethodno pomenuta funkcija prima kao jedini parametar. Testirati implementaciju `Sqrt` funkcije i proveriti kolika su odstupanja[5] njenih rezultata od rezultata funkcije `math.Sqrt`[6]. Dodatne smernice za implementaciju i testiranje možete pronaći na sledećem linku.

80. Implementirati funkciju `WordCount`, koja kao parametar prima jedan string. Njen zadatak je da u prethodno pomenutom stringu izdvoji sve reči i prebroji koliko puta se svaka od njih pojavljuje. Povratna vrednost funkcije `WordCount` treba da bude mapa čiji parovi imaju sledeći oblik "`reč : broj_pojavljivanja_reči`". Ovako implementiranu funkciju testirati na nekoliko primera. Dodatne smernice za implementaciju i testiranje možete pronaći na sledećem linku.

81. Implementirati funkciju `fibonacci` čija je povratna vrednost druga (unutrašnja) funkcija. Zadatak ove unutrašnje funkcije je da pri svakom pozivu[7] vrati odgovarajući element niza Fibonačijevih brojeva[8], počevši od 0[9]. Dodatne smernice za implementaciju možete pronaći na sledećem linku.

82. Kreirati tip `IPAddr` tako da predstavlja IPv4[10] adresu. Prilikom poziva funkcije `fmt.Print` sa argumentom `IPAddr{192, 168, 0, 1}`, na standardnom izlazu bi trebalo da se prikaže sledeći ispis `192.168.0.1` . Dodatne smernice za implementaciju možete pronaći na sledećem linku.

83. Funkciju `Sqrt` iz zadatka 79. potrebno je proširiti tako da prijavi grešku ukoliko joj se kao argument prilikom poziva prosledi negativan broj. Greška treba da sadrži neodgovarajuću vrednost argumenta koji ju je izazvao. Dodatne smernice za implementaciju možete pronaći na sledećem linku.

84. Implementirati novi tip podatka koji zadovljava `Reader` interfejs i čiji je zadatak da emituje beskonačan tok `ASCII` karaktera `'A'`. Dodatne smernice za implementaciju možete pronaći na sledećem linku.

85. Kreirati tip podatka `rot13Reader` koji zadvoljava `Reader` interfejs. Zadatak tipa `rot13Reader` je da čita podatke iz nekog toka i modifikuje ih primenom[11] Cezarove šifre zamena[12]. Dodatne smernice za implementaciju možete pronaći na sledećem linku.

---

4 https://en.wikipedia.org/wiki/Newton%27s_method
5 Možete isprobati odstupanja za prvih 1000000 prirodnih brojeva.
6 https://golang.org/pkg/math/#Sqrt
7 Pri prvom pozivu vraća broj 0, pri drugom pozivu broj 1, itd.
8 https://en.wikipedia.org/wiki/Fibonacci_number
9 U pojedinoj literaturi se kao prvi Fibonačijev broj navodi broj 1.
10 https://en.wikipedia.org/wiki/IPv4
11 Cezarovu šifru zamene treba primeniti samo na mala i velika slova engleskog alfabeta.
12 https://en.wikipedia.org/wiki/ROT13

86. Upotrebom *Go* rutina, implementirati `Same` funkciji čiji je zadatak da proveri da li su dva sortirana binarna stabla[13] ekvivalentna. Dodatne smernice za implementaciju možete pronaći na sledećem linku.

87. Upotrebom *Go* rutina, implementirati jednostavan veb-*crawler* koji polazeći od zadate veb-stranice rekurzivno obilazi sve susedne stranice[14], vodeći računa da svaku stranicu obiđe tačno jednom. Dodatne smernice za implementaciju možete pronaći na sledećem linku.

88. Generisati dinamički niz sa bar 10000 slučajno odabranih brojeva iz intervala `[0, 255]`. Implementirati funkcije `zbir` i `zbirParalelno` koje treba da odrede zbir brojeva prethodno pomenutog niza sekvencijalno, odnosno paralelno[15] upotrebom *Go* rutina. Odraditi eksperimente jakog i slabog skaliranja funkcije `zbirParalelno`.

13 Vrednost sadržana u nekom čvoru sortiranog binarnog stabla je veća od svih vrednosti sadržanih u levom podstablu tog čvora, a manja od svih vrednosti sadržanih u desnom podstablu prethodno pomenutog čvora. Pretpostavka je da su sve vrednosti sadržane u čvorovima jednog binarnog stabla različite.

14 Stranice A i B su direktni susedi ukoliko u bar jednoj od njih postoji link ka onoj drugoj.

15 Pretpostavka je da će se programski kod izvršavati na višejezgarnom procesoru.