UNIVERZITET U NOVOM SADU
FAKULTET TEHNIČKIH NAUKA
KATEDRA ZA PRIMENJENE RAČUNARSKE NAUKE

Paralelni i distribuirani algoritmi i strukture podataka

ms Nebojša Horvat

Zimski semestar 2019/2020.

Studijski program: Računarstvo i automatika

Modul: Računarstvo visokih performansi

Go (Golang)

Rekapitulacija

- Funkcija može da vrati proizvoljan broj vrednosti
- Rad sa fajlovima
  - bufio paket
- Strukture i pokazivači
  - Slično kao u jeziku C
- Statički nizovi, slajsevi i mape, funkcija make
- Obrada grešaka-izuzetaka (nema try, catch blokova)
  - Funkcije imaju povratnu vrednost koja je indikator o tome da li je bilo greške

Interfejsi

- Interfejsi predstavljaju imenovane kolekcije potpisa metoda
- Analogni su interfejsima u drugim jezicima
- Dodavanje objektne paradigme na Golang
- Ključna reč “interface”
- Interfejs se ne može da naslediti dugi interfejs
- Međutim mogućeje je kombinovati interfejse i od dva interfejsa napraviti novi koji sadrži sve metode intejfejsa od kojih se sastoji

Interfejsi

```
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 + "]"
}
```

Interfejsi

```
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)
}
```

Zadatak za vežbu

- Implementirati program koji ima interface oblik sa dve funkcije: obim i površina
- Napraviti dve strukture Trougao ii Pravougaonik koje implementiraju interfejs.
- Obratiti pažnju na to da li se koristi vrednost ili pokazivač pri implementaciji funkcija.

Go Rutine (Goroutines)

- Funkcije koje se izvršavaju konkurentno sa drugim funkcijama
- U Golang programima dešava se da istovremeno postoji 1000 go rutina
- “lakše” od niti (thread)
- Komunikacija preko kanala
- Pozivaju se pomoću prefixa “go”

Go Rutine (Goroutines)

- 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.

Go Rutine (Goroutines)

```
func hello(from string) {
    for i := 1; i < 100000000; 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()
}
```

Go Rutine (Goroutines)

```
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)
}
```

Kanali

- Kanali su “cevi” kojima komuniciraju Go rutine
- Rade po principu FIFO
- 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)

Kanali

- data := <- a // Čitanje iz kanala a
- a <- data // Pisanje u kanal a

Go Rutine i kanali - primer

```
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)
}
```

Go Rutine i kanali - primer

```
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) // need to create channel
go sumArrayIntoChannel(s[:len(s)/4], c, 1)
go sumArrayIntoChannel(s[5:10], c, 2)
go sumArrayIntoChannel(s[10:15], c, 3)
go sumArrayIntoChannel(s[15:20], c, 4)
x, y, z, v := <-c, <-c, <-c, <-c // receive from c

fmt.Println(x, y, z, v, x+y+z+v)
```

Zadatak za vežbu

- Implementirati program koji prima n brojeva na ulazu i traži njihov minimum.
- Koristiti go rutine za traženje minimuma u podnizu
- Koristiti rezultate pojedinačnih go rutina da se izračuna globalni minimum.

Upravljanje paketima

- go get [link do repozitorijuma]
- Paketi će biti preuzeti i smešteni u bin folder relativan u odnosu na GOPATH
- Nakon toga preuzeti paketi se mogu koristiti kao I nativni paketi Golang-a

Testiranje

- Paket “testing”
- Unutar istog paketa/foldera kao I testirana jedinica
- Naziv: nazivTestiraneJedinice_test.go
- Pokretanje testa sa : go test
- Postoji mnogo drugih paketa za ovu svrhu
- Autori jezika ne preporučuju tipične Assert konstrukcije
- Test tabele
  - Testiranje za više ulaznih vrednosti

Testiranje

```
import "testing"

func TestSum(t *testing.T) {
    total := Sum(5, 5)
    if total != 10 {
        t.Errorf("Sum was incorrect, got: %d, want: %d.", total, 10)
    }
}
```

Zadatak - Testiranje

- Napisati funkciju koja kreira palindrom od ulaznog stringa (string mora imati neparan broj karaktera)
- Napisati test koji pokazuje da funkcija radi kako je očekivano
- Napisati test koji pokazuje da program vraća odgovarajuću grešku kada se prosledi string koji ima paran broj karaktera

Zadatak za kući

- Implementirati program koji se koristi za evidenciju fudbalskog šampionata:
- Korisnik unosi n klubova kroz standardni ulaz
- Klub sadrži atribute:
  - Naziv, skraćeni naziv, godina osnivanja
  - Naziv mora biti jedinstven
- Nakon unosa klubova, korisnik može da izabere opciju da generiše mečeve
  - Generišu se mečevi tako da svaki tim igra protiv svakog drugog tima 2 puta
    - Na domaćem i na gostujućem terenu
  - Za svaki meč se beleži
    - Id utakmice, domaća ekipa , gostujuća ekipa, vreme održavanja utakmice i rezultat
    - Rezultate generisati na slučajan način pri čemu se svaki rezultat generiše tako da na utakmici ne može biti više od 10 golova,
    - Takođe, u celom šampionatu mora biti postignuto tačno1000 golova
- Podatke o mečevima i klubovima upisati u 2 posebna fala (mecevi.txt, klubovi.txt)
- Omogućiti prikaz tabele klubova sortirane po broju bodova na konzoli kao i upis tabele u fajl (pobeda nosi 3 boda, a nerešen rezultat 1 bod)
  - Tabela sadrži naziv kluba, broj bodova i gol razliku

Zadatak

- Proširiti prethodni zadatak tako da se generisanje svakog rezultata vrši u posebnoj go rutini
- Omogućiti izmenu postojećih rezultata, tako što se korisniku da mogućnost da bira rezultat koji želi da izmeni
  - Nakon izmene podaci se opet beleže u fajl
- Broj golova u šampionatu i maksimalan kao i minimalan broj golova po meču zadaje korisnik