# Soal Intro Algorithm & Basic Programming

## Soal Prioritas 1 (80)

### Program 1: Menghitung Luas Trapesium

Buatlah sebuah program untuk menghitung luas trapesium.

### Program 2: Menentukan Bilangan Ganjil atau Genap

Buatlah sebuah program untuk menentukan apakah sebuah bilangan adalah bilangan ganjil atau genap.

### Program 3: Menentukan Grade dari Sebuah Nilai

Buatlah sebuah program untuk menentukan grade dari sebuah nilai, dengan ketentuan sebagai berikut:

- Nilai 80 - 100: A

- Nilai 65 - 79: B

- Nilai 50 - 64: C

- Nilai 35 - 49: D

- Nilai 0 - 34: E

- Nilai kurang dari 0 atau lebih dari 100 maka tampilkan 'Nilai Invalid'

### Program 4: Mencetak Angka dari 1 Sampai 100

Buatlah sebuah program yang mencetak angka dari 1 sampai 100 dan untuk kelipatan '3' cetak "Fizz" sebagai ganti angka, dan untuk kelipatan '5' cetak "Buzz”. Sebagai contoh; 1 2 fizz 4 buzz fizz 7 8 fizz buzz …….

## Soal Prioritas 2 (20)

### Program 1: Mencetak Segitiga Asterik

Buatlah sebuah program untuk mencetak segitiga asterik seperti berikut:

```

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

```

### Program 2: Mencetak Faktor Bilangan

Buatlah sebuah program untuk mencetak faktor bilangan dari sebuah angka, seperti berikut:

```

Masukkan sebuah bilangan: 12

Faktor dari 12 adalah: 1, 2, 3, 4, 6, 12

```

## Soal Eksplorasi (20)

### Program 1: Palindrome

Buatlah sebuah program untuk memeriksa apakah sebuah kata adalah palindrome atau bukan, serta coba terapkan scanner untuk menangkap inputan dari console, seperti berikut:

```

Masukkan sebuah kata: level

level adalah sebuah palindrome.

Masukkan sebuah kata: hello

hello bukan sebuah palindrome.

```

# Revisi Soal Intro Algorithm & Basic Programming

## Soal Prioritas 1 (80)

### Program 1: Menghitung Luas Trapesium

Buatlah sebuah program untuk menghitung luas trapesium dengan rumus sebagai berikut:

```

Luas = 0.5 \* (a + b) \* t

```

dengan keterangan sebagai berikut:

- `a` dan `b` adalah panjang sisi sejajar trapesium

- `t` adalah tinggi trapesium

### Program 2: Menentukan Bilangan Ganjil atau Genap

Buatlah sebuah program untuk menentukan apakah sebuah bilangan adalah bilangan ganjil atau genap.

### Program 3: Menentukan Grade dari Sebuah Nilai

Buatlah sebuah program untuk menentukan grade dari sebuah nilai, dengan ketentuan sebagai berikut:

- Nilai 80 - 100: A

- Nilai 65 - 79: B

- Nilai 50 - 64: C

- Nilai 35 - 49: D

- Nilai 0 - 34: E

- Nilai kurang dari 0 atau lebih dari 100 maka tampilkan 'Nilai Invalid'

### Program 4: Mencetak Angka dari 1 Sampai 100

Buatlah sebuah program yang mencetak angka dari 1 sampai 100 dan untuk kelipatan '3' cetak "Fizz" sebagai ganti angka, dan untuk kelipatan '5' cetak "Buzz”. Sebagai contoh; 1 2 fizz 4 buzz fizz 7 8 fizz buzz …….

## Soal Prioritas 2 (20)

### Program 1: Mencetak Segitiga Asterik

Buatlah sebuah program untuk mencetak segitiga asterik seperti berikut:

```

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

```

### Program 2: Mencetak Faktor Bilangan

Buatlah sebuah program untuk mencetak faktor bilangan dari sebuah angka, seperti berikut:

```

Masukkan sebuah bilangan: 12

Faktor dari 12 adalah: 1, 2, 3, 4, 6, 12

```

### Program 3: Palindrome

Buatlah sebuah program untuk memeriksa apakah sebuah kata adalah palindrome atau bukan, serta coba terapkan scanner untuk menangkap inputan dari console, seperti berikut:

```

Masukkan sebuah kata: level

level adalah sebuah palindrome.

Masukkan sebuah kata: hello

hello bukan sebuah palindrome.

```

# Soal Intro Algorithm & Basic Programming

## Soal Prioritas 1 (80)

### Program 1: Menghitung Luas Trapesium

Buatlah sebuah program untuk menghitung luas trapesium.

### Program 2: Menentukan Bilangan Ganjil atau Genap

Buatlah sebuah program untuk menentukan apakah sebuah bilangan adalah bilangan ganjil atau genap.

### Program 3: Menentukan Grade dari Sebuah Nilai

Buatlah sebuah program untuk menentukan grade dari sebuah nilai, dengan ketentuan sebagai berikut:

- Nilai 80 - 100: A

- Nilai 65 - 79: B

- Nilai 50 - 64: C

- Nilai 35 - 49: D

- Nilai 0 - 34: E

- Nilai kurang dari 0 atau lebih dari 100 maka tampilkan 'Nilai Invalid'

### Program 4: Mencetak Angka dari 1 Sampai 100

Buatlah sebuah program yang mencetak angka dari 1 sampai 100 dan untuk kelipatan '3' cetak "Fizz" sebagai ganti angka, dan untuk kelipatan '5' cetak "Buzz”. Sebagai contoh; 1 2 fizz 4 buzz fizz 7 8 fizz buzz …….

## Soal Prioritas 2 (20)

### Program 1: Mencetak Segitiga Asterik

Buatlah sebuah program untuk mencetak segitiga asterik seperti berikut:

```

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

```

### Program 2: Mencetak Faktor Bilangan

Buatlah sebuah program untuk mencetak faktor bilangan dari sebuah angka, seperti berikut:

```

Masukkan sebuah bilangan: 12

Faktor dari 12 adalah: 1, 2, 3, 4, 6, 12

```

## Soal Eksplorasi (20)

### Program 1: Palindrome

Buatlah sebuah program untuk memeriksa apakah sebuah kata adalah palindrome atau bukan, serta coba terapkan scanner untuk menangkap inputan dari console, seperti berikut:

```

Masukkan sebuah kata: level

level adalah sebuah palindrome.

Masukkan sebuah kata: hello

hello bukan sebuah palindrome.

```

# Revisi Soal Intro Algorithm & Basic Programming

## Soal Prioritas 1 (80)

### Program 1: Menghitung Luas Trapesium

Buatlah sebuah program untuk menghitung luas trapesium dengan rumus sebagai berikut:

```

Luas = 0.5 \* (a + b) \* t

```

dengan keterangan sebagai berikut:

- `a` dan `b` adalah panjang sisi sejajar trapesium

- `t` adalah tinggi trapesium

### Program 2: Menentukan Bilangan Ganjil atau Genap

Buatlah sebuah program untuk menentukan apakah sebuah bilangan adalah bilangan ganjil atau genap.

### Program 3: Menentukan Grade dari Sebuah Nilai

Buatlah sebuah program untuk menentukan grade dari sebuah nilai, dengan ketentuan sebagai berikut:

- Nilai 80 - 100: A

- Nilai 65 - 79: B

- Nilai 50 - 64: C

- Nilai 35 - 49: D

- Nilai 0 - 34: E

- Nilai kurang dari 0 atau lebih dari 100 maka tampilkan 'Nilai Invalid'

### Program 4: Mencetak Angka dari 1 Sampai 100

Buatlah sebuah program yang mencetak angka dari 1 sampai 100 dan untuk kelipatan '3' cetak "Fizz" sebagai ganti angka, dan untuk kelipatan '5' cetak "Buzz”. Sebagai contoh; 1 2 fizz 4 buzz fizz 7 8 fizz buzz …….

## Soal Prioritas 2 (20)

### Program 1: Mencetak Segitiga Asterik

Buatlah sebuah program untuk mencetak segitiga asterik seperti berikut:

```

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

```

### Program 2: Mencetak Faktor Bilangan

Buatlah sebuah program untuk mencetak faktor bilangan dari sebuah angka, seperti berikut:

```

Masukkan sebuah bilangan: 12

Faktor dari 12 adalah: 1, 2, 3, 4, 6, 12

```

### Program 3: Palindrome

Buatlah sebuah program untuk memeriksa apakah sebuah kata adalah palindrome atau bukan, serta coba terapkan scanner untuk menangkap inputan dari console, seperti berikut:

```

Masukkan sebuah kata: level

level adalah sebuah palindrome.

Masukkan sebuah kata: hello

hello bukan sebuah palindrome.

```

# Soal Intro Algorithm & Basic Programming

## Soal Prioritas 1 (80)

### Program 1: Menghitung Luas Trapesium

Buatlah sebuah program untuk menghitung luas trapesium.

### Program 2: Menentukan Bilangan Ganjil atau Genap

Buatlah sebuah program untuk menentukan apakah sebuah bilangan adalah bilangan ganjil atau genap.

### Program 3: Menentukan Grade dari Sebuah Nilai

Buatlah sebuah program untuk menentukan grade dari sebuah nilai, dengan ketentuan sebagai berikut:

- Nilai 80 - 100: A

- Nilai 65 - 79: B

- Nilai 50 - 64: C

- Nilai 35 - 49: D

- Nilai 0 - 34: E

- Nilai kurang dari 0 atau lebih dari 100 maka tampilkan 'Nilai Invalid'

### Program 4: Mencetak Angka dari 1 Sampai 100

Buatlah sebuah program yang mencetak angka dari 1 sampai 100 dan untuk kelipatan '3' cetak "Fizz" sebagai ganti angka, dan untuk kelipatan '5' cetak "Buzz”. Sebagai contoh; 1 2 fizz 4 buzz fizz 7 8 fizz buzz …….

## Soal Prioritas 2 (20)

### Program 1: Mencetak Segitiga Asterik

Buatlah sebuah program untuk mencetak segitiga asterik seperti berikut:

```

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

```

### Program 2: Mencetak Faktor Bilangan

Buatlah sebuah program untuk mencetak faktor bilangan dari sebuah angka, seperti berikut:

```

Masukkan sebuah bilangan: 12

Faktor dari 12 adalah: 1, 2, 3, 4, 6, 12

```

## Soal Eksplorasi (20)

### Program 1: Palindrome

Buatlah sebuah program untuk memeriksa apakah sebuah kata adalah palindrome atau bukan, serta coba terapkan scanner untuk menangkap inputan dari console, seperti berikut:

```

Masukkan sebuah kata: level

level adalah sebuah palindrome.

Masukkan sebuah kata: hello

hello bukan sebuah palindrome.

```

# Revisi Soal Intro Algorithm & Basic Programming

## Soal Prioritas 1 (80)

### Program 1: Menghitung Luas Trapesium

Buatlah sebuah program untuk menghitung luas trapesium dengan rumus sebagai berikut:

```

Luas = 0.5 \* (a + b) \* t

```

dengan keterangan sebagai berikut:

- `a` dan `b` adalah panjang sisi sejajar trapesium

- `t` adalah tinggi trapesium

### Program 2: Menentukan Bilangan Ganjil atau Genap

Buatlah sebuah program untuk menentukan apakah sebuah bilangan adalah bilangan ganjil atau genap.

### Program 3: Menentukan Grade dari Sebuah Nilai

Buatlah sebuah program untuk menentukan grade dari sebuah nilai, dengan ketentuan sebagai berikut:

- Nilai 80 - 100: A

- Nilai 65 - 79: B

- Nilai 50 - 64: C

- Nilai 35 - 49: D

- Nilai 0 - 34: E

- Nilai kurang dari 0 atau lebih dari 100 maka tampilkan 'Nilai Invalid'

### Program 4: Mencetak Angka dari 1 Sampai 100

Buatlah sebuah program yang mencetak angka dari 1 sampai 100 dan untuk kelipatan '3' cetak "Fizz" sebagai ganti angka, dan untuk kelipatan '5' cetak "Buzz”. Sebagai contoh; 1 2 fizz 4 buzz fizz 7 8 fizz buzz …….

## Soal Prioritas 2 (20)

### Program 1: Mencetak Segitiga Asterik

Buatlah sebuah program untuk mencetak segitiga asterik seperti berikut:

```

\*

\*\*

\*\*\*

\*\*\*\*

\*\*\*\*\*

```

### Program 2: Mencetak Faktor Bilangan

Buatlah sebuah program untuk mencetak faktor bilangan dari sebuah angka, seperti berikut:

```

Masukkan sebuah bilangan: 12

Faktor dari 12 adalah: 1, 2, 3, 4, 6, 12

```

### Program 3: Palindrome

Buatlah sebuah program untuk memeriksa apakah sebuah kata adalah palindrome atau bukan, serta coba terapkan scanner untuk menangkap inputan dari console, seperti berikut:

```

Masukkan sebuah kata: level

level adalah sebuah palindrome.

Masukkan sebuah kata: hello

hello bukan sebuah palindrome.

```

Git adalah sebuah sistem pengaturan versi (version control system) yang dapat membantu dalam mengelola repositori GitHub Anda. Berikut adalah beberapa perintah umum dalam Git untuk GitHub:

1. git clone [URL]: Menyalin repositori GitHub ke komputer lokal Anda.

Contoh: git clone https://github.com/user/repo.git

2. git add [file]: Menambahkan file ke staging area untuk di-commit.

Contoh: git add file.txt

3. git commit -m "[pesan]": Melakukan commit pada perubahan yang sudah ditambahkan ke staging area dengan memberikan pesan deskriptif.

Contoh: git commit -m "Menambahkan fitur baru"

4. git push: Mengirimkan commit ke repositori GitHub.

Contoh: git push origin master

5. git pull: Mengambil perubahan terbaru dari repositori GitHub dan memperbarui repositori lokal Anda.

Contoh: git pull origin master

6. git branch [nama branch]: Membuat branch baru di repositori Anda.

Contoh: git branch feature-branch

7. git checkout [nama branch]: Pindah ke branch yang ditentukan.

Contoh: gitc heckout feature-branch

8. git merge [nama branch]: Menggabungkan perubahan dari branch yang ditentukan ke branch aktif Anda.

Contoh: git merge feature-branch

9. git status: Menampilkan status perubahan pada repositori lokal Anda.

Contoh: git status

10. git log: Menampilkan daftar commit pada repositori Anda.

Contoh: git log

Time complexity adalah ukuran seberapa cepat program berjalan dan bagaimana kinerjanya tergantung pada ukuran input yang diberikan. Dalam bahasa Go, kompleksitas waktu dapat dihitung dengan mengukur jumlah operasi elementer yang diperlukan untuk menyelesaikan suatu tugas terhadap ukuran input yang diberikan. Time complexity diukur dalam notasi Big O.

Contoh program Go di bawah ini menghitung jumlah elemen dalam array dan memiliki time complexity O(n), di mana n adalah jumlah elemen dalam array tersebut:

```go

package main

import "fmt"

func main() {

arr := []int{1, 2, 3, 4, 5}

sum := 0

for i := 0; i < len(arr); i++ {

sum += arr[i]

}

fmt.Println("Jumlah elemen dalam array:", sum)

}

```

Space complexity mengukur berapa banyak ruang yang dibutuhkan oleh suatu algoritma untuk menyelesaikan tugas terhadap ukuran input yang diberikan. Space complexity juga diukur dalam notasi Big O.

Tipe Data Integer

Tipe data integer terdiri dari dua jenis, yaitu:

1. int: Bilangan cacah (Bilangan positif)

2. uint: Bilangan bulat (bilangan positif maupun negatif)

Penggunaan uint8 dan int8:

- uint8 merepresentasikan angka dengan skala 0 - 255.

- int8 merepresentasikan angka dengan skala -128 - 127.

Ada baiknya menentukan tipe data integer dengan jenis apa yang ingin digunakan untuk menghindari boros memori.

Contoh penggunaan tipe data integer:

```

package main

import "fmt"

func main() {

var satu uint8 = 89

var dua int8 = -89

fmt.Printf("tipe data satu : %T\n", satu)

fmt.Printf("tipe data dua : %T\n", dua)

}

```

Output program:

```

tipe data satu : uint8

tipe data dua : int8

```

Penggunaan uint untuk angka positif dan penggunaan int8 untuk angka negatif.

Tipe Data String

Tipe data string digunakan untuk merepresentasikan teks atau karakter. Contoh penggunaan tipe data string:

```

package main

import "fmt"

func main() {

var m string = "reza"

fmt.Print(m)

}

```

Output program:

```

reza

```

Nil & Zero Value

Nil bukan merupakan tipe data, melainkan sebuah nilai. Variabel yang isi nilainya nil berarti memiliki nilai kosong.

Zero value adalah nilai awal dari sebuah variabel saat belum diberikan nilai secara eksplisit. Berikut adalah zero value dari beberapa tipe data:

- Zero value dari string adalah "" (string kosong).

- Zero value dari bool adalah false.

- Zero value dari tipe numerik non-desimal adalah 0.

- Zero value dari tipe numerik desimal adalah 0.0.

Nil bisa digunakan pada beberapa tipe data seperti pointer, function, slice, map, channel, dan interface kosong atau interface{}.

Sedangkan untuk penggunaan nil pada tipe data selain yang tersebut di atas tidak diperbolehkan.

Contoh penggunaan nil:

```

package main

import "fmt"

func main() {

var ptr \*int = nil

fmt.Println("Nilai ptr: ", ptr)

var arr []int = nil

fmt.Println("Nilai arr: ", arr)

var mp map[string]int = nil

fmt.Println("Nilai mp: ", mp)

var ch chan int = nil

fmt.Println("Nilai ch: ", ch)

var iface interface{} = nil

fmt.Println("Nilai iface: ", iface)

}

```

Output program:

```

Nilai ptr: <nil>

Nilai arr: <nil>

Nilai mp: <nil>

Nilai ch: <nil>

Nilai iface: <nil>

```

Dari contoh di atas, variabel ptr, arr, mp, ch, dan iface memiliki nilai kosong atau nil.

Pada Go, penggunaan zero value juga berguna saat membuat variabel dengan tipe data struct atau array yang membutuhkan inisialisasi awal. Dalam hal ini, zero value akan digunakan sebagai nilai awal dari setiap elemen pada struct atau array tersebut.

Contoh penggunaan zero value pada struct dan array:

```

package main

import "fmt"

type Person struct {

Name string

Age int

}

func main() {

// membuat variabel struct tanpa inisialisasi

var p Person

fmt.Println(p) // outputnya { 0}

// membuat array dengan panjang tertentu tanpa inisialisasi

var arr [5]int

fmt.Println(arr) // outputnya [0 0 0 0 0]

}

```

# Golang: String, Rune, Byte, dan Struct

String adalah tipe data yang paling umum digunakan di Golang. Tipe data ini digunakan untuk merepresentasikan urutan karakter Unicode yang membentuk sebuah string. Rune adalah tipe data yang merepresentasikan sebuah karakter Unicode, yang terdiri dari satu atau beberapa byte. Byte adalah tipe data yang merepresentasikan sebuah byte dalam urutan byte yang membentuk sebuah string. []byte adalah tipe data slice dari byte, yang digunakan untuk merepresentasikan urutan byte yang membentuk sebuah string. []rune adalah tipe data slice dari rune, yang digunakan untuk merepresentasikan urutan karakter Unicode yang membentuk sebuah string.

Stringer adalah sebuah interface yang digunakan untuk mengubah suatu nilai ke dalam bentuk string, sehingga nilainya dapat dicetak atau ditampilkan. Builder adalah tipe data yang digunakan untuk membangun sebuah string secara dinamis, yaitu dengan menambahkan karakter atau urutan karakter ke dalam string tersebut secara bertahap.

## Contoh Sederhana Method

```go

package main

import (

"fmt"

)

// Define a struct named Rectangle

type Rectangle struct {

width float64

height float64

}

// Define a method for Rectangle struct to calculate its area

func (r Rectangle) area() float64 {

return r.width \* r.height

}

func main() {

// Create a new Rectangle object

rect := Rectangle{width: 10, height: 5}

// Call the area method for the Rectangle object

area := rect.area()

// Print the area of the rectangle

fmt.Printf("The area of the rectangle is: %f\n", area)

}

```

## Contoh Sederhana Struct

```go

package main

import (

"fmt"

)

// Define a struct named Person

type Person struct {

Name string

Age int

Address string

}

func main() {

// Create a new Person object

person := Person{Name: "John", Age: 30, Address: "123 Main St"}

// Print the person's information

fmt.Printf("Name: %s, Age: %d, Address: %s\n", person.Name, person.Age, person.Address)

// Change the person's address

person.Address = "456 Oak St"

// Print the updated information

fmt.Printf("Name: %s, Age: %d, Address: %s\n", person.Name, person.Age, person.Address)

}

```

# Pengenalan Algoritma

Algoritma merupakan langkah-langkah atau instruksi yang digunakan untuk menyelesaikan suatu masalah atau tugas. Algoritma dapat digunakan dalam berbagai bidang, seperti pemrograman, matematika, fisika, dan sebagainya.

## Recursive

Recursive merupakan fungsi yang memanggil dirinya sendiri secara berulang sampai mencapai kondisi dasar atau base case. Recursive sering digunakan dalam metode pencarian dan pengurutan data. Namun, penggunaan rekursi yang berlebihan dapat menyebabkan stack overflow dan performa yang buruk.

Contoh:

```go

package main

import "fmt"

func fibonacci(n int) int {

if n <= 1 {

return n

}

return fibonacci(n-1) + fibonacci(n-2)

}

func main() {

fmt.Println("10 bilangan pertama dalam deret Fibonacci:")

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

fmt.Print(fibonacci(i), " ")

}

}

```

## Number Theory

Number Theory merupakan cabang matematika yang mempelajari sifat-sifat bilangan bulat.

Contoh:

```go

package main

import (

"fmt"

"math/big"

)

func main() {

n := big.NewInt(123456789) // bilangan yang akan difaktorkan

fmt.Printf("Faktorisasi prima dari %d adalah:\n", n)

factors := big.NewInt(2) // inisialisasi faktor awal dengan 2

// ulangi pembagian hingga tidak dapat dibagi lagi

for n.Cmp(big.NewInt(1)) > 0 {

for n.Mod(n, factors).Cmp(big.NewInt(0)) == 0 {

n.Div(n, factors)

fmt.Print(factors, " ")

}

factors.Add(factors, big.NewInt(1))

}

}

```

## Searching

Searching merupakan metode untuk mencari data dalam suatu himpunan. Beberapa metode yang sering digunakan dalam searching antara lain linear search, binary search, dan hash table.

Contoh:

```go

package main

import "fmt"

func linearSearch(slice []int, target int) bool {

for \_, num := range slice {

if num == target {

return true

}

}

return false

}

func main() {

// contoh slice

nums := []int{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

// pencarian angka 5

if linearSearch(nums, 5) {

fmt.Println("Angka 5 ditemukan di dalam slice")

} else {

fmt.Println("Angka 5 tidak ditemukan di dalam slice")

}

// pencarian angka 11

if linearSearch(nums, 11) {

fmt.Println("Angka 11 ditemukan di dalam slice")

} else {

fmt.Println("Angka 11 tidak ditemukan di dalam slice")

}

}

```

## Sorting

Sorting merupakan metode untuk mengurutkan data dalam suatu himpunan. Beberapa metode yang sering digunakan dalam sorting antara lain bubble sort, selection sort, insertion sort, merge sort, dan quick sort.

Contoh:

```go

package main

import "fmt"

func bubbleSort(slice []int) {

n := len(slice)

for i := 0; i < n; i++ {

for j := 0; j < n-i-1; j++ {

if slice[j] > slice[j+1] {

slice[j], slice[j+1] = slice[j+1], slice[j]

}

}

}

}

func main() {

// contoh slice yang akan diurutkan

nums := []int{4, 2, 7, 1, 8, 3, 5, 6}

// urutkan slice menggunakan bubble sort

bubbleSort(nums)

// cetak slice yang telah diurutkan

fmt.Println(nums)

}

```

# Materi Problem Solving

Problem Solving adalah suatu proses untuk menyelesaikan suatu masalah dengan cara yang sistematis dan logis. Terdapat beberapa teknik yang digunakan dalam problem solving, di antaranya adalah brute force, divide conquer, greedy, dan dynamic programming.

## Brute Force

Brute Force adalah teknik yang mencoba semua kemungkinan solusi untuk menemukan solusi yang paling optimal. Teknik ini cukup sederhana dan mudah dipahami, tetapi dapat memakan waktu yang lama dan tidak efisien untuk masalah yang kompleks.

Contoh program Brute Force:

```go

package main

import "fmt"

func findMax(a int, b int) int {

var max int = a

if b > a {

max = b

}

return max

}

func main() {

fmt.Println(findMax(10, 20)) // Output: 20

fmt.Println(findMax(0, -5)) // Output: 0

fmt.Println(findMax(100, 100)) // Output: 100

}

```

## Divide Conquer

Divide Conquer adalah teknik yang memecah masalah menjadi sub-masalah yang lebih kecil dan kemudian menyelesaikan sub-masalah tersebut secara terpisah. Setelah semua sub-masalah terselesaikan, solusi akhir dapat digabungkan dari solusi sub-masalah.

Contoh program Divide Conquer:

```go

package main

import "fmt"

func main() {

n := 5

result := factorial(n)

fmt.Printf("Factorial of %d is %d\n", n, result)

}

func factorial(n int) int {

if n == 0 {

return 1

}

return n \* factorial(n-1)

}

```

## Greedy

Greedy adalah teknik yang mencari solusi yang paling optimal pada setiap tahap dengan harapan bahwa solusi lokal tersebut akan menghasilkan solusi yang optimal secara keseluruhan. Teknik ini sederhana dan cepat, tetapi tidak selalu menghasilkan solusi yang paling optimal.

Contoh program Greedy:

```go

package main

import (

"fmt"

"sort"

)

type Item struct {

Value, Weight int

}

type ByDensity []Item

func (a ByDensity) Len() int { return len(a) }

func (a ByDensity) Swap(i, j int) { a[i], a[j] = a[j], a[i] }

func (a ByDensity) Less(i, j int) bool { return a[i].Value/a[i].Weight > a[j].Value/a[j].Weight }

func MaxValue(items []Item, capacity int) int {

sort.Sort(ByDensity(items))

var value, weight int

for \_, item := range items {

if weight+item.Weight <= capacity {

value += item.Value

weight += item.Weight

} else {

remaining := capacity - weight

value += remaining \* item.Value / item.Weight

break

}

}

return value

}

func main() {

items := []Item{

{60, 10},

{100, 20},

{120, 30},

}

capacity := 50

maxValue := MaxValue(items, capacity)

fmt.Printf("Maximum value for capacity %d: %d\n", capacity, maxValue)

}

```

## Dynamic Programming

Dynamic Programming adalah teknik yang menggunakan memoization untuk menyimpan hasil dari sub-masalah yang sudah dipecah sebelumnya dan menghindari perhitungan ulang yang tidak perlu. Teknik ini sangat efisien untuk masalah yang kompleks.

Contoh program Dynamic Programming:

```go

package main

import "fmt"

func Fibonacci(n int) int {

fib := make([]int, n+1)

fib[0] = 0

fib[1] = 1

for i := 2; i <= n; i++ {

fib[i] = fib[i-1] + fib[i-2]

}

return fib[n]

}

func main() {

fmt.Println(Fibonacci(10)) // Output: 55

}

```

Materi ini berfungsi untuk mempercepat dalam pencarian dan memperkecil ruang pencarian.

Program Sequential

Merupakan program yang dieksekusi secara berurutan. Contohnya sebagai berikut:

```

for i:=1 to 10 do

fmt.Println(i)

```

Program di atas akan menampilkan angka 1 hingga 10 secara berurutan.

Program Parallel

Merupakan program yang dieksekusi secara bersamaan. Contohnya sebagai berikut:

```

go func(){

for i:=1 to 10 do

fmt.Println(i)

}()

```

Program di atas akan menampilkan angka 1 hingga 10 secara bersamaan.

Program Concurrent

Merupakan program yang dieksekusi secara bersamaan dan berurutan. Contohnya sebagai berikut:

```

func main(){

go func(){

for i:=1 to 10 do

fmt.Println(i)

}()

fmt.Scanln()

}

```

Program di atas akan menampilkan angka 1 hingga 10 secara bersamaan dan akan menunggu input dari pengguna sebelum program selesai dieksekusi.

Untuk membuat cleancode, sebaiknya kita menggunakan format yang sama untuk setiap contoh program dan memberikan penjelasan yang jelas dan singkat untuk setiap jenis program. Selain itu, sebaiknya kode program ditulis dengan rapi dan mudah dibaca. Contoh cleancode untuk program sequential adalah sebagai berikut:

```

// Program Sequential

// Merupakan program yang dieksekusi secara berurutan

for i := 1; i <= 10; i++ {

fmt.Println(i)

}

```

Contoh cleancode untuk program parallel adalah sebagai berikut:

```

// Program Parallel

// Merupakan program yang dieksekusi secara bersamaan

go func() {

for i := 1; i <= 10; i++ {

fmt.Println(i)

}

}()

```

Contoh cleancode untuk program concurrent adalah sebagai berikut:

```

// Program Concurrent

// Merupakan program yang dieksekusi secara bersamaan dan berurutan

func main() {

go func() {

for i := 1; i <= 10; i++ {

fmt.Println(i)

}

}()

fmt.Scanln()

}

```

Clean code adalah kode yang ditulis secara rapi dan mudah dibaca oleh programmer lain. Kode yang bersih dan mudah dibaca akan memudahkan programmer untuk memahami dan mengembangkan kode tersebut.

Pentingnya clean code karena akan memudahkan programmer dalam bekerja secara tim, meningkatkan efisiensi dan kualitas kode, dan meminimalisir kesalahan dalam pengembangan software.

Beberapa karakteristik dari clean code antara lain:

1. KISS (Keep It Simple, Stupid)

2. DRY (Don't Repeat Yourself)

3. Refactoring

4. Consistency

5. Meaningful Names

6. Small Functions

7. Error Handling

8. Unit Testing

9. Comments

KISS (Keep It Simple, Stupid) mengajarkan kita untuk menulis kode sesederhana mungkin. Hal ini akan memudahkan programmer untuk memahami dan mengembangkan kode tersebut.

DRY (Don't Repeat Yourself) mengajarkan kita untuk menghindari duplikasi kode. Sebuah kode yang memiliki banyak duplikasi akan sulit untuk diubah dan diperbaiki.

Refactoring adalah proses mengubah kode yang sudah ada menjadi lebih baik. Hal ini dilakukan untuk meningkatkan kualitas dan efisiensi kode.

Dengan menerapkan karakteristik-karakteristik tersebut, maka kode yang dihasilkan akan menjadi lebih bersih, mudah dibaca, dan mudah dipahami.

Sebagai contoh, penggunaan variable yang mudah dipahami seperti "user balance" yang dapat diartikan sebagai saldo pengguna. Dalam penggunaan variable tersebut, penamaan variabel tersebut menjelaskan secara jelas dan mudah dipahami oleh programmer lain.

Sebaliknya, penggunaan variable yang tidak mudah dipahami seperti "a" akan membuat programmer lain kesulitan dalam memahami dan mengembangkan kode tersebut.

Dengan mengikuti prinsip clean code, kita dapat meningkatkan kualitas dan efisiensi kode yang kita tulis, sehingga memudahkan dalam pengembangan software.

Introduction Database

Database adalah kumpulan data yang tersimpan dalam suatu sistem yang terorganisir dan dapat diakses oleh pengguna. Database digunakan untuk menyimpan data yang berkaitan dengan suatu organisasi atau perusahaan.

Relationship Database

Relationship Database adalah jenis database yang menggunakan hubungan antara entitas atau objek sebagai dasar untuk menyimpan data. Dalam Relationship Database, data disimpan dalam tabel yang terhubung satu sama lain melalui kunci asing dan kunci utama.

Entity Relationship Diagram

Entity Relationship Diagram (ERD) adalah diagram yang digunakan untuk memodelkan hubungan antara entitas dalam suatu database. ERD biasanya digunakan untuk memvisualisasikan hubungan antara tabel dalam database.

SQL Statement

SQL Statement adalah perintah yang digunakan untuk mengakses atau memanipulasi data dalam database. SQL Statement dapat digunakan untuk membuat tabel, memasukkan data ke dalam tabel, mengubah data dalam tabel, atau mengambil data dari tabel. SQL Statement juga dapat digunakan untuk membuat query untuk menampilkan data dalam format yang diinginkan.

# Materi Database Schema dan Data Definition Language

## Pengertian Database

Database adalah sekumpulan data yang terorganisir dalam suatu sistem yang dapat diakses dan dikelola dengan mudah.

## Relationship Database

Relationship Database adalah hubungan antara tabel-tabel dalam suatu database relasional. Terdapat tiga jenis hubungan dalam database relasional, yaitu:

### 1. One-to-One

Hubungan antara dua tabel dalam basis data relasional di mana satu rekaman dalam tabel pertama hanya berkorespondensi dengan satu rekaman dalam tabel kedua, dan sebaliknya.

Contoh: Tabel "customer" dan "customer\_address".

### 2. One-to-Many

Hubungan antara dua tabel dalam basis data relasional di mana satu rekaman dalam tabel pertama dapat berkorespondensi dengan banyak rekaman dalam tabel kedua. Namun, setiap rekaman di tabel kedua hanya dapat berkorespondensi dengan satu rekaman di tabel pertama.

Contoh: Hubungan antara tabel "orders" dan "order\_details".

### 3. Many-to-Many

Hubungan antara dua tabel dalam basis data relasional di mana banyak rekaman dalam tabel pertama dapat berkorespondensi dengan banyak rekaman dalam tabel kedua, dan sebaliknya.

Contoh: Hubungan antara tabel "students" dan "courses".

## Entity Relationship Diagram

Entity Relationship Diagram (ERD) adalah sebuah diagram yang digunakan untuk merepresentasikan hubungan antara entitas dalam suatu database.

## SQL Statement

SQL (Structured Query Language) adalah bahasa yang digunakan untuk mengakses dan mengelola database. Terdapat beberapa jenis perintah SQL, yaitu:

### 1. Data Definition Language (DDL)

DDL digunakan untuk membuat, mengubah, dan menghapus struktur tabel dalam database. Contoh perintah DDL: CREATE, ALTER, dan DROP.

### 2. Data Manipulation Language (DML)

DML digunakan untuk memanipulasi data dalam tabel, seperti menambah, mengubah, dan menghapus data. Contoh perintah DML: INSERT, SELECT, UPDATE, dan DELETE.

### 3. Data Control Language (DCL)

DCL digunakan untuk mengatur hak akses pengguna dalam database. Contoh perintah DCL: GRANT dan REVOKE.

## Join (Standard SQL ANSI)

Join adalah perintah SQL yang digunakan untuk menggabungkan dua tabel atau lebih berdasarkan kolom yang dimiliki. Terdapat tiga jenis join dalam SQL, yaitu:

### 1. Inner Join

Inner Join digunakan untuk mengambil data yang terdapat pada kedua tabel yang di-join-kan.

### 2. Left Join

Left Join digunakan untuk mengambil semua data dari tabel kiri dan data yang cocok dari tabel kanan.

### 3. Right Join

Right Join digunakan untuk mengambil semua data dari tabel kanan dan data yang cocok dari tabel kiri.

## Pengertian Onion Model

Onion Model adalah sebuah model yang digunakan untuk memisahkan antara data dan aplikasi. Model ini terdiri dari 3 lapisan, yaitu lapisan data, lapisan aplikasi, dan lapisan presentasi. Lapisan data berisi data-data yang akan digunakan oleh aplikasi. Lapisan aplikasi berisi program-program yang akan mengolah data-data yang ada di lapisan data. Lapisan presentasi berisi tampilan-tampilan yang akan ditampilkan kepada pengguna.

## Aggregate Function

Aggregate Function adalah fungsi yang digunakan untuk menghitung nilai-nilai yang berkaitan dengan kumpulan data, seperti nilai rata-rata, nilai maksimum, dan nilai minimum.

# Perbedaan GUI dan CLI

GUI atau Graphical User Interface adalah sebuah antarmuka yang dapat dioperasikan dengan menggunakan klik pada ikon-ikon yang disediakan. Sedangkan CLI atau Command Line Interface adalah sebuah antarmuka yang menggunakan perintah-perintah teks melalui terminal atau command prompt.

Kelebihan menggunakan CLI adalah dapat mengakses semua fitur yang ada di Linux dan pastinya lebih cepat. Sedangkan kelebihan menggunakan GUI adalah mudah dipahami dan lebih mudah dalam pengoperasian.

# Kenapa Menggunakan Basic Command

Menggunakan basic command sangat penting karena kita dapat mengakses semua fitur yang ada di Linux dan pastinya lebih cepat, terutama dalam pengoperasian big data atau server. Selain itu, dengan menggunakan basic command kita dapat membuat script untuk otomatisasi, membantu dalam pemecahan masalah, dan membantu dalam pengoperasian server.

# Macam-Macam User

1. Normal user: user yang biasa digunakan untuk melakukan aktivitas di Linux.

2. Root user: user yang memiliki hak akses penuh.

3. Super user: user yang memiliki hak akses penuh dan dapat mengakses semua fitur dan perintah di Linux.

# Beberapa Basic Command di Linux

1. `pwd`: menampilkan lokasi direktori yang sedang aktif.

2. `ls`: menampilkan isi direktori yang sedang aktif.

3. `cd`: berpindah direktori.

4. `mkdir`: membuat direktori baru.

5. `rmdir`: menghapus direktori.

6. `rm`: menghapus file.

7. `cp`: mengcopy file.

8. `mv`: memindahkan file.

9. `cat`: menampilkan isi file.

10. `grep`: mencari kata tertentu di dalam file.

11. `chmod`: mengubah permission suatu file atau direktori.

Dengan menguasai basic command di Linux, kita dapat dengan mudah mengoperasikan Linux dan melakukan berbagai aktivitas seperti membuat, mengedit, dan menghapus file atau direktori, mencari file dengan cepat, dan mengubah permission suatu file atau direktori.

# Diagram Design

## Karakteristik Sistem Terdistribusi

Sistem Terdistribusi memiliki karakteristik yaitu:

- Terdiri dari beberapa komputer yang bekerja sama untuk menyelesaikan tugas

- Terdapat komunikasi dan koordinasi antar komponen

- Terdapat kemampuan untuk melakukan scaling secara horizontal atau vertikal

## Horizontal Scaling vs Vertical Scaling

- Horizontal Scaling: Menambah jumlah instance pada suatu aplikasi dengan tujuan membagi beban kerja

- Vertical Scaling: Menambah kapasitas pada suatu instance, misalnya dengan menambah RAM atau CPU

## Job/Work Queue

Job Queue atau Work Queue adalah mekanisme untuk menyelesaikan tugas secara terstruktur. Tugas-tugas tersebut ditempatkan pada antrian, dan instance yang tersedia dapat mengambil tugas dari antrian tersebut untuk dieksekusi.

## Load Balancing

Load Balancing adalah mekanisme untuk membagi beban kerja pada beberapa instance agar tidak terjadi overload pada suatu instance. Load Balancer akan memantau beban kerja pada setiap instance dan memutuskan instance mana yang akan menerima permintaan selanjutnya.

## Monolitik vs Mikroservice

- Monolitik: Aplikasi yang dibangun sebagai satu kesatuan yang besar, biasanya sulit untuk melakukan scaling secara horizontal

- Mikroservice: Aplikasi yang dibangun sebagai kumpulan service yang terpisah, memungkinkan scaling secara horizontal

## SQL vs NoSQL

- SQL: Database yang memanfaatkan Structured Query Language sebagai bahasa query, cocok untuk data terstruktur

- NoSQL: Database yang tidak memanfaatkan Structured Query Language, cocok untuk data yang tidak terstruktur

## Caching

Caching adalah mekanisme untuk menyimpan data sementara pada suatu instance, sehingga permintaan selanjutnya tidak perlu memuat data dari sumber asli. Hal ini dapat meningkatkan performa dan mengurangi beban kerja pada sumber asli.

## Database Indexing

Indexing adalah mekanisme untuk mempercepat pencarian data pada suatu database dengan membuat indeks pada kolom-kolom yang sering digunakan.

## Database Replication

Database Replication adalah mekanisme untuk membuat salinan data dari suatu database pada instance lain, sehingga terdapat cadangan data jika terjadi kegagalan pada suatu instance.

## Tools Diagram

Beberapa tools yang dapat digunakan untuk membuat diagram:

- SmartDraw

- Lucidchart

- Draw.io

- Visio

- Whimsical

## Software Design

Beberapa jenis design yang umum digunakan adalah:

- Flowchart

- Use Case

- Entity Relationship Diagram (ERD)

- High-Level System Design (HLS)

## Pengertian dan Konsep Scalability

Scalability adalah kemampuan suatu sistem untuk meningkatkan kinerjanya dengan menambah sumber daya, baik secara horizontal atau vertikal. Scalability juga berhubungan dengan kemampuan sistem untuk merespon perubahan beban kerja atau data dengan cepat dan efektif. Pemilihan arsitektur hardware/software yang tepat juga sangat penting dalam scalability.

## Kelayakan, Ketersediaan, dan Efisiensi

Kelayakan (Reliability), Ketersediaan (Availability), dan Efisiensi (Efficiency) adalah tiga hal yang sering dibahas dalam lingkungan TI dan layanan digital. Kelayakan mengacu pada daya tahan sistem terhadap gangguan atau masalah teknis, ketersediaan mengacu pada responsivitas sistem dalam menyediakan layanan, dan efisiensi mengacu pada penggunaan sumber daya yang ada untuk menghasilkan solusi dan output yang diinginkan.

RESTful API adalah suatu arsitektur untuk membangun aplikasi web yang merupakan representasi dari gaya arsitektur berbasis web yang digunakan dalam pengembangan aplikasi web. RESTful API menggunakan protokol HTTP sebagai dasar komunikasi antara client dan server. Dalam RESTful API terdapat beberapa metode HTTP yang digunakan seperti GET, POST, PUT, DELETE, dan sebagainya.

Postman pada Golang adalah suatu aplikasi yang digunakan untuk menguji dan mengelola API. Postman dapat membantu pengguna untuk melakukan pengujian API dengan mudah dan efisien. Postman juga dapat digunakan untuk membuat request HTTP, menampilkan hasil response, dan melakukan debugging pada API. Selain itu, Postman juga dapat digunakan untuk mengelola koleksi API dan menyimpan konfigurasi API dalam satu tempat.

# Materi – Intro Echo Golangs

## Pengertian API

API atau Application Programming Interface adalah kumpulan aturan, protokol, dan alat yang digunakan untuk membangun perangkat lunak dan aplikasi. API merupakan interface yang berfungsi menghubungkan program dengan sumber daya atau layanan lainnya yang ada pada sistem operasi atau di dalam aplikasi itu sendiri.

Secara singkat, API adalah cara untuk menjembatani interaksi antara dua program. Dengan menggunakan API, pengembang bisa membuat aplikasi yang dapat berinteraksi dengan aplikasi dan layanan lain secara mudah dan efisien.

API biasanya memiliki beberapa endpoint atau titik akhir yang dapat diakses oleh pengembang dengan melakukan permintaan HTTP yang sesuai dengan aturan-aturan yang telah ditetapkan oleh API tersebut. Setelah permintaan diterima oleh API server, data yang diminta akan dikirim kembali ke aplikasi pengguna dalam format yang telah ditentukan sebelumnya.

Contohnya, saat Anda menggunakan aplikasi pemesanan tiket pesawat, aplikasi tersebut berkomunikasi dengan API provider tiket pesawat

untuk mendapatkan informasi harga tiket, jadwal penerbangan, dan lain-lain. Kemudian aplikasi akan menampilkan hasilnya dengan tampilan yang sudah dibuat sebelumnya.

## Format Request dan Response

Format yang umum digunakan dalam request dan response dari API adalah JSON dan XML. Contoh object notation JSON:

```

{

"name": "John",

"age": 30,

"cars": [

{ "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },

{ "name":"BMW", "models":[ "320", "X3", "X5" ] },

{ "name":"Fiat", "models":[ "500", "Panda" ] }

]

}

```

## HTTP Response Code

HTTP Response Code adalah kode status yang diberikan oleh server dalam respons terhadap permintaan yang diberikan oleh client. Beberapa contoh kode respons yang umum digunakan antara lain:

- 200 OK

- 201 Created

- 202 Accepted

- 204 No Content

- 301 Moved Permanently

- 302 Found

- 304 Not Modified

- 400 Bad Request

- 401 Unauthorized

- 403 Forbidden

- 404 Not Found

- 405 Method Not Allowed

- 500 Internal Server Error

- 502 Bad Gateway

- 503 Service Unavailable

## Tools yang Digunakan

Beberapa tools yang digunakan dalam pengembangan API antara lain:

- Postman

- Katalon

- Apache JMeter

- SoapUI

## SWAPI (Star Wars API)

SWAPI (Star Wars API) adalah sebuah layanan web yang menyediakan data terkait Star Wars. API ini dapat diakses oleh developer untuk mengintegrasikan data-data tersebut dengan aplikasi atau website mereka.

Beberapa contoh penggunaan SWAPI antara lain:

- Membuat website dengan tema Star Wars yang menampilkan detail karakter, planet, dan film dari saga Star Wars.

- Membuat aplikasi mobile yang dapat memberikan informasi lengkap tentang suatu karakter atau planet dari Star Wars.

- Menggunakan data dari SWAPI untuk membuat game trivia Star Wars.

Untuk mengakses SWAPI, developer perlu mendaftar akun dan memperoleh API key. Selanjutnya data dapat diambil melalui endpoint-endpoint yang telah disediakan oleh SWAPI. Data yang bisa diambil antara lain karakter, planet, film, dan jenis-jenis kendaraan dalam Star Wars. Data tersebut tersedia dalam format JSON yang dapat dengan mudah diintegrasikan dengan berbagai bahasa pemrograman seperti Python dan JavaScript.

## Third Party Golang

Third Party Library Golang adalah library yang dikembangkan oleh pengembang pihak ketiga untuk membantu proses pengembangan aplikasi menggunakan bahasa pemrograman Go. Beberapa contoh library yang umum digunakan antara lain:

- Echo

- Go kit

- Logrus

- Gorm.io

- Cobra

## Binding Data

Binding data merupakan proses menghubungkan data dari suatu sumber dengan tampilan pada aplikasi. Hal ini dilakukan agar data yang ditampilkan di aplikasi selalu terupdate secara otomatis berdasarkan perubahan data yang terjadi pada sumbernya.

Contoh penggunaan binding data adalah pada aplikasi web yang menampilkan data dari database. Dengan menggunakan teknik binding data, ketika data pada database berubah, maka tampilan data pada aplikasi web juga akan berubah secara otomatis tanpa harus melakukan refresh halaman.

Pada pengembangan aplikasi, binding data dapat diterapkan pada berbagai platform, baik itu aplikasi desktop, web, maupun mobile. Beberapa framework atau library yang biasanya digunakan untuk mengimplementasikan teknik binding data seperti AngularJS, ReactJS, VueJS, EmberJS, dan lain sebagainya.

# ORM dan Struktur Kode

## ORM (Object-Relational Mapping)

ORM adalah suatu teknologi yang memungkinkan kita untuk mengakses database relasional dengan menggunakan objek-objek yang terdefinisi dalam suatu bahasa pemrograman. Dengan menggunakan ORM, kita dapat memanipulasi data pada database dengan lebih mudah karena kita tidak perlu menulis query SQL secara langsung.

## GORM

GORM adalah salah satu ORM untuk Golang yang cukup populer dan banyak digunakan. GORM menyediakan fitur-fitur yang lengkap, seperti dukungan untuk berbagai macam database, relasi antar tabel, validasi data, dan lain-lain.

Untuk menginstall GORM, kita dapat mengikuti langkah-langkah berikut:

1. Buka terminal atau PowerShell.

2. Jalankan perintah `go get -u gorm.io/gorm`.

3. Jalankan perintah `go get -u gorm.io/driver/sqlite` jika kamu ingin menggunakan database SQLite.

## Struktur Kode

Untuk membuat kode kita lebih terstruktur, kita dapat membagi kode menjadi beberapa direktori dan file. Berikutadalah struktur kode yang dapat digunakan untuk membuat CRUD dengan GORM:

- `config`: berisi konfigurasi untuk koneksi ke database.

- `controller`: berisi fungsi-fungsi yang akan dipanggil saat ada request dari client.

- `lib`: berisi helper function atau library yang dapat digunakan di seluruh aplikasi.

- `models`: berisi definisi model atau struktur tabel pada database.

- `routes`: berisi definisi route atau endpoint yang dapat diakses oleh client.

- `main.go`: file utama yang akan memanggil fungsi-fungsi dari package lain dan menjalankan server.

Dengan membagi kode menjadi beberapa package seperti ini, kita dapat memudahkan proses development dan memperbaiki kode jika terjadi kesalahan pada bagian tertentu.

Untuk menjalankan aplikasi, kita dapat mengetikkan perintah `go run main.go` di terminal atau PowerShell. Setelah itu, kita dapat mengakses aplikasi melalui browser atau aplikasi sejenis Postman dengan memasukkan URL endpoint yang telah kita definisikan di dalam `routes`.

# Apa itu Middleware?

Middleware adalah perangkat lunak yang bertindak sebagai jembatan antara aplikasi dan sistem operasi atau infrastruktur yang mendukung aplikasi. Middleware membantu memfasilitasi komunikasi dan pertukaran data antara aplikasi yang berbeda, serta menyediakan fungsi tambahan seperti manajemen sesi, autentikasi, dan caching.

Contoh middleware meliputi server aplikasi, message broker, gateway API, web server, dan banyak lagi. Middleware sangat penting dalam lingkungan komputasi terdistribusi di mana beberapa aplikasi atau sistem harus bekerja bersama untuk mencapai tujuan yang sama. Dengan memungkinkan aplikasi berkomunikasi satu sama lain dan mengakses sumber daya yang sama, middleware memungkinkan pengembangan dan operasi sistem yang lebih efisien dan terkoordinasi.

# Contoh Program Golang Middleware

Berikut adalah contoh program Golang yang menggunakan middleware untuk melakukan logging setiap kali request HTTP diterima:

```go

package main

import (

"log"

"net/http"

"time"

)

func main() {

// Membuat handler untuk endpoint "/hello"

helloHandler := http.HandlerFunc(func(w http.ResponseWriter, r \*http.Request) {

w.Write([]byte("Hello, world!"))

})

// Membuat middleware untuk logging

loggingMiddleware := func(next http.Handler) http.Handler {

return http.HandlerFunc(func(w http.ResponseWriter, r \*http.Request) {

start := time.Now()

next.ServeHTTP(w, r)

log.Printf("[%s] %s %s", r.Method, r.URL.Path, time.Since(start))

})

}

// Menggabungkan handler dan middleware

handler := loggingMiddleware(helloHandler)

// Menjalankan server HTTP di port 8080

log.Println("Server started on port 8080")

err := http.ListenAndServe(":8080", handler)

log.Fatal(err)

}

```

Dalam contoh ini, ketika server HTTP dijalankan, program akan mencetak log setiap kali request

Unit Testing

Unit testing adalah proses menguji sebuah program untuk mengetahui apakah program tersebut sudah benar atau belum. Tujuannya adalah untuk menjalankan program dengan baik dan memastikan bahwa tidak ada kesalahan yang terjadi ketika melakukan perubahan.

Definisi

Unit testing merujuk pada proses pengujian program secara terperinci, dengan menguji setiap unit program secara terpisah. Dalam hal ini, unit program adalah bagian terkecil dari program yang bisa diuji secara terpisah.

Purpose

Tujuan dari unit testing adalah untuk memastikan bahwa setiap bagian program berfungsi dengan baik dan tidak menghasilkan kesalahan. Dalam hal ini, unit testing membantu meningkatkan kualitas program dan mengurangi risiko kesalahan saat melakukan perubahan.

Level

Unit testing dilakukan pada level unit program, yaitu pada setiap bagian terkecil program yang bisa diuji secara terpisah.

Framework

Ada beberapa framework yang bisa digunakan untuk melakukan unit testing, seperti Golang testing, JUnit (Java), NUnit (.NET), dan sebagainya.

Struktur

Unit testing terdiri dari beberapa bagian, yaitu:

- Test case: sebuah tes untuk menguji satu unit program.

- Test suite: kumpulan dari beberapa tes kasus yang dijalankan bersama-sama.

- Test plan: rencana pengujian yang menggambarkan bagaimana pengujian akan dilakukan.

- Test strategy: strategi pengujian yang menjelaskan bagaimana pengujian akan dilakukan.

Cara Melakukan Unit Testing

Untuk melakukan unit testing, bisa dilakukan dengan menempatkan file pengujian pada direktori yang sama dengan file program yang akan diuji. Selain itu, bisa juga dengan menggunakan perintah "go test" pada terminal. Setelah itu, program akan dijalankan dan output yang dihasilkan akan menunjukkan apakah tes berhasil atau gagal.

Kesimpulan

Unit testing sangat penting untuk memastikan bahwa program berfungsi dengan baik dan tidak menghasilkan kesalahan. Dalam hal ini, unit testing dilakukan pada level unit program dan terdiri dari beberapa bagian, yaitu test case, test suite, test plan, dan test strategy. Ada beberapa framework yang bisa digunakan untuk melakukan unit testing, dan pengujian bisa dilakukan dengan menempatkan file pengujian pada direktori yang sama dengan file program atau menggunakan perintah "go test" pada terminal.

untuk melakukan pengeetesan

bisa masuk dalam program di mana taro file di stc

bisa juga dengan cara

GO BUILD INIT MAIN.GO

atau

GO MOD INIT MAIN.GO

Clean Architecture dan Hexagonal Architecture adalah paradigma pemrograman yang membantu pengembang untuk membangun sistem yang terstruktur, modular, dan mudah dipelihara.

Clean Architecture adalah metode desain perangkat lunak yang menekankan pada pemisahan antara kode bisnis, antarmuka pengguna, dan infrastruktur. Clean Architecture memungkinkan pengembang untuk mengembangkan aplikasi yang mudah dipelihara, mudah diuji, dan mudah dikembangkan ulang dengan menggunakan prinsip SOLID.

Sementara itu, Hexagonal Architecture adalah pendekatan pengembangan perangkat lunak yang berfokus pada pemisahan antara inti bisnis dan infrastruktur. Dalam Hexagonal Architecture, inti bisnis adalah inti dari aplikasi dan infrastruktur adalah lapisan yang terpisah dari inti bisnis. Hal ini memungkinkan pengembang untuk membuat aplikasi yang lebih fleksibel, mudah diuji, dan mudah dikembangkan ulang.

Kedua pendekatan ini dapat diterapkan dalam bahasa pemrograman Go (Golang) dengan menggunakan konsep package dan interface. Dalam Go, package dapat digunakan untuk memisahkan kode bisnis dari infrastruktur, dan interface dapat digunakan untuk memisahkan antara inti bisnis dan infrastruktur.

Dengan menerapkan Clean Architecture dan Hexagonal Architecture dalam pengembangan perangkat lunak menggunakan Go, pengembang dapat membuat aplikasi yang lebih mudah dipelihara, mudah diuji, dan mudah dikembangkan ulang. Hal ini akan membantu pengembang untuk menghasilkan perangkat lunak yang lebih baik dan lebih efisien.

# Menjalankan Docker pada Terminal

## Mengecek Versi Docker

Untuk mengecek versi Docker yang terinstall pada komputer, gunakan perintah berikut pada terminal:

```

docker version

```

## Mengecek Images

Untuk mengecek images yang sudah terdownload pada komputer, gunakan perintah berikut pada terminal:

```

docker images

```

## Mengecek Container

Untuk mengecek container yang ada pada komputer, gunakan perintah berikut pada terminal:

```

docker container ls --all

```

Perintah ini akan menampilkan semua container yang pernah dibuat pada komputer.

## Membuat Container

Untuk membuat container baru, gunakan perintah berikut pada terminal:

```

docker container create --name [nama\_container] [nama\_image]

```

Contoh:

```

docker container create --name golangserver1 golang

```

## Mendownload Images

Untuk mendownload images baru dari Docker Hub, gunakan perintah berikut pada terminal:

```

docker pull [nama\_image]

```

Contoh:

```

docker pull golang

```

## Menjalankan Container

Untuk menjalankan container yang sudah dibuat, gunakan perintah berikut pada terminal:

```

docker container start [nama\_container]

```

Contoh:

```

docker container start golangserver1

```

## Menghentikan Container

Untuk menghentikan container yang sedang berjalan, gunakan perintah berikut pada terminal:

```

docker container stop [nama\_container]

```

Contoh:

```

docker container stop golangserver1

```

## Menghapus Container

Untuk menghapus container yang sudah tidak dibutuhkan, gunakan perintah berikut pada terminal:

```

docker container rm [nama\_container]

```

Contoh:

```

docker container rm golangserver1

```

## Menghapus Images

Untuk menghapus images yang sudah tidak dibutuhkan, gunakan perintah berikut pada terminal:

```

docker image rm [nama\_image]

```

Contoh:

```

docker image rm golang

```

## Membuka Port Pada Container

Untuk membuka port pada container, gunakan perintah berikut pada terminal:

```

docker container create --name [nama\_container] -p [port\_host]:[port\_container] [nama\_image]

```

Contoh:

```

docker container create --name golang1 -p 8000:27017 golang

```

Perintah ini akan membuka port 8000 pada host dan menghubungkannya ke port 27017 pada container.

Compute Services pada Golang adalah layanan untuk melakukan komputasi pada aplikasi yang dibangun menggunakan bahasa pemrograman Go. Layanan ini menyediakan berbagai fitur seperti pemrosesan data, pengolahan gambar, dan pemrograman paralel. Dengan menggunakan Compute Services pada Golang, aplikasi dapat melakukan komputasi secara efisien dan cepat, sehingga dapat meningkatkan performa dan efektivitas aplikasi. Selain itu, Compute Services pada Golang juga menyediakan dukungan untuk berbagai platform seperti cloud computing dan mobile devices. Dengan demikian, Compute Services pada Golang dapat membantu pengembang untuk membangun aplikasi yang lebih efisien dan efektif.

## contoh program

```go

package main

import (

"fmt"

"math/rand"

"time"

)

func main() {

rand.Seed(time.Now().UnixNano()) // generate random seed

var nums []int

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

nums = append(nums, rand.Intn(100)) // generate random numbers from 0 to 99

}

fmt.Println("Random numbers:", nums)

fmt.Println("Sum of numbers:", computeSum(nums))

}

func computeSum(nums []int) int {

sum := 0

for \_, num := range nums {

sum += num

}

return sum

}

```

CI/CD adalah singkatan dari Continuous Integration/Continuous Delivery atau Continuous Deployment. Ini adalah metodologi pengembangan perangkat lunak di mana setiap kali ada perubahan kode dalam proyek, kode tersebut diintegrasikan ke dalam repositori utama dan diuji secara otomatis. Setelah pengujian berhasil, perubahan tersebut dikirim ke lingkungan produksi.

Contoh CI/CD adalah ketika seorang pengembang menambahkan kode baru ke dalam repositori utama. Sistem CI/CD akan secara otomatis mengambil kode tersebut, mengintegrasikannya dengan kode yang ada, dan menjalankan serangkaian tes otomatis untuk memastikan bahwa perubahan tersebut tidak memecahkan aplikasi. Jika tes berhasil, perubahan tersebut akan otomatis diterapkan ke lingkungan produksi.

Dalam praktiknya, CI/CD dapat dilakukan dengan bantuan alat seperti Jenkins, GitLab CI/CD, atau CircleCI. Dalam proyek perangkat lunak, CI/CD memungkinkan pengembang untuk mengirimkan perubahan dengan lebih cepat dan lebih aman, karena setiap perubahan diuji secara otomatis sebelum dikirim ke pelanggan. Ini juga memungkinkan tim pengembang untuk bekerja lebih efisien, karena mereka dapat mengintegrasikan kode mereka dengan lebih cepat dan lebih mudah.