LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 3

FUNGSI

Oleh:

MUHAMMAD RUSDIYANTO

2311102053

S1IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

Fungsi merupakan sebuah subprogram atau bagian dari suatu program yang menjalankan suatu perintah spesifik. Fungsi memiliki ciri khas yaitu selalu mengembalikan nilai. Dalam pemrograman, pengembalian nilai ini biasanya diikuti oleh keyword `return`. Karena fungsi mengembalikan nilai, maka fungsi bisa digunakan untuk assignment (memasukkan) nilai ke suatu variabel. Nama fungsi sebaiknya adalah kata yang menggambarkan nilai, seperti median, max, min, dan sebagainya.

Dalam Go, fungsi dideklarasikan seperti ini :

```
func [nama_fungsi]([parameter] tipe_data) [tipe_return] {
  [badan fungsi]
  return [nilai]
}
```

Fungsi juga bisa dideklarasikan tanpa parameter.

Untuk memanggil fungsi dalam Go bisa dilakukan seperti ini :

```
Nama_fungsi([parameter])
```

Pemanggilan fungsi juga bisa dilakukan secara rekursif. Arti rekursif disini adalah fungsi memanggil dirinya sendiri, sehingga proses terjadi terus menerus. Supaya bisa berhenti, maka perlu ditambahkan percabangan/kondisi.

II. GUIDED

Guided 1 | Source Code

```go
package main

import "fmt"

func main() {
    var a, b int
    fmt.Scan(&a, &b)
    if a >= b {
        fmt.Println(permutasi(a, b))
    } else {
        fmt.Println(permutasi(b, a))
    }
}

func faktorial(n int) int {
    var hasil int = 1
    var i int
    for i = 1; i <= n; i++ {
        hasil = hasil * i
    }
    return hasil
}
func permutasi(n, r int) int {
    return faktorial(n) / faktorial((n - r))
}
```

Guided 1 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run guided1.go
5 3
60
```

Guided 1 | Penjelasan

Program di atas adalah program perhitungan permutasi. Proses dalam program ini dimulai dengan input angka a dan b. Setelah input, maka program akan mengecek apakah a lebih besar atau sama dengan b. Jika iya, maka program akan melakukan perhitungan a permutasi b. Jika tidak, maka program akan melakukan perhitungan b permutasi a (dibalik). Perhitungan permutasi dalam percabangan tersebut dilakukan dengan memanggil fungsi `permutasi` sekaligus meneruskan nilai dari kedua angka. Dalam hal ini urutan angka yang diteruskan akan mempengaruhi hasil, sehingga perlu dipastikan bahwa angka pertama adalah angka yang lebih besar.

`Permutasi` sendiri merupakan fungsi yang digunakan untuk menghitung hasil permutasi 2 bilangan (n dan r). Proses dalam fungsi ini cukup sederhana, yaitu mengembalikan hasil perhitungan dari n! / (n-r)!. Perhitungan faktorial dalam fungsi ini dilakukan dengan memanggil fungsi `faktorial`. Fungsi tersebut memiliki 1 parameter `n`, dimana `n` merupakan angka yang akan difaktorialkan. Nilai faktorial didapatkan dengan melakukan perkalian secara berulang dari 1 hingga angka yang akan difaktorialkan. Dalam program ini, hasil perkalian tersebut disimpan di dalam variabel `hasil`. Setelah perulangan selesai (nilai faktorial ditemukan), maka nilai tersebut akan dikembalikan ke fungsi `permutasi` untuk diproses kembali. Lalu, proses dilanjutkan dengan membagi kedua faktorial (n! dan [n-r]!). Setelah dibagi, maka hasil permutasi akan dikembalikan ke fungsi main, dimana hasil tersebut akan ditampilkan.

Guided 2 | Source Code

```go
package main

import "fmt"

// Fungsi buat menghitung faktorial
func factorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
    result := 1
    for i := 2; i <= n; i++ {
        result *= i
    }
    return result
}

// Fungsi buat menghitung permutasi
func permutation(n, r int) int {
    return factorial(n) / factorial(n-r)
}

// Fungsi buat menghitung kombinasi
func combination(n, r int) int {
    return factorial(n) / (factorial(r) * factorial(n-r))
}

func main() {
```

```
    // Input 4 bilangan
    var a, b, c, d int
    fmt.Println("Masukkan bilangan a, b, c, d (dengan
spasi): ")
    fmt.Scanf("%d %d %d %d", &a, &b, &c, &d)

    // Cek syarat a >= c dan b >= d
    if a >= c && b >= d {
        // Menghitung permutasi dan kombinasi a dan c
        permutasiAC := permutation(a, c)
        kombinasiAC := combination(a, c)

        // Menghitung permutasi dan kombinasi b dan d
        permutasiBD := permutation(b, d)
        kombinasiBD := combination(b, d)

        // Output hasil
        fmt.Println("Permutasi(a, c) dan Kombinasi(a, c):",
permutasiAC, kombinasiAC)
        fmt.Println("Permutasi(b, d) dan Kombinasi(b, d):",
permutasiBD, kombinasiBD)
    } else {
        fmt.Println("Syarat a >= c dan b >= d tidak
terpenuhi.")
    }
}
```

Guided 2 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run guided2.go
Masukkan bilangan a, b, c, d (dengan spasi):
5 10 3 10
Permutasi(a, c) dan Kombinasi(a, c): 60 10
Permutasi(b, d) dan Kombinasi(b, d): 3628800 1
```

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run guided2.go
Masukkan bilangan a, b, c, d (dengan spasi):
8 0 2 0
Permutasi(a, c) dan Kombinasi(a, c): 56 28
Permutasi(b, d) dan Kombinasi(b, d): 1 1
```

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run guided2.go
Masukkan bilangan a, b, c, d (dengan spasi):
2 3 4 5
Syarat a >= c dan b >= d tidak terpenuhi.
```

Guided 2 | Penjelasan

Program di atas merupakan pengembangan dari program guided 1. Disamping permutasi, program ini juga bisa melakukan perhitungan

kombinasi. Proses eksekusi kode dalam program ini dimulai dengan input 4 bilangan (a, b, c, d), dimana a lebih besar dari atau sama dengan c dan b lebih besar dari atau sama dengan d. Jika persyaratan tersebut tidak terpenuhi, maka program akan mengeluarkan pesan bahwa syarat tidak terpenuhi. Jika persyaratan terpenuhi, maka program akan kalkulasi a permutasi c, a kombinasi c, b permutasi d, dan b kombinasi d. Perhitungan tersebut dilakukan dengan memanggil fungsi.

III. UNGUIDED

Unguided 1 | Source Code

```
package main

import (
    "fmt"
    "math"
)

func akar(x float64) float64 {
    return math.Sqrt(x)
}

func jarak(a int, b int, c int, d int) float64 {
    return math.Pow(float64(a-c), 2) + math.Pow(float64(b-
d), 2)
}

func didalam(pusat_x int, pusat_y int, r int, sembarang_x
int, sembarang_y int) bool {
    var jarak_titik = akar(jarak(sembarang_x, sembarang_y,
pusat_x, pusat_y))
    if jarak_titik < float64(r) {
        return true
    }
    return false
}

func main() {
    var pusat_x1, pusat_x2, pusat_y1, pusat_y2, r1, r2,
sembarang_x, sembarang_y int
    var hasil1, hasil2 bool
    fmt.Print("Titik pusat (x,y) dan jejari lingkaran 1 : ")
    fmt.Scanln(&pusat_x1, &pusat_y1, &r1)
    fmt.Print("Titik pusat (x,y) dan jejari lingkaran 2 : ")
    fmt.Scanln(&pusat_x2, &pusat_y2, &r2)
    fmt.Print("Titik sembarang (x,y) : ")
    fmt.Scanln(&sembarang_x, &sembarang_y)
    hasil1 = didalam(pusat_x1, pusat_y1, r1, sembarang_x,
sembarang_y)
    hasil2 = didalam(pusat_x2, pusat_y2, r2, sembarang_x,
sembarang_y)
```

```
    if hasil1 && hasil2 {
        fmt.Println("Titik di dalam lingkaran 1 dan 2")
    } else if hasil1 {
        fmt.Println("Titik di dalam lingkaran 1")
    } else if hasil2 {
        fmt.Println("Titik di dalam lingkaran 2")
    } else {
        fmt.Println("Titik di luar lingkaran 1 dan 2")
    }
}
```

Unguided 1 | Output

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run unguided1.go
Titik pusat (x,y) dan jejari lingkaran 1 : 1 1 5
Titik pusat (x,y) dan jejari lingkaran 2 : 8 8 4
Titik sembarang (x,y) : 2 2
Titik di dalam lingkaran 1
```

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run unguided1.go
Titik pusat (x,y) dan jejari lingkaran 1 : 1 2 3
Titik pusat (x,y) dan jejari lingkaran 2 : 4 5 6
Titik sembarang (x,y) : 7 8
Titik di dalam lingkaran 2
```

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run unguided1.go
Titik pusat (x,y) dan jejari lingkaran 1 : 5 10 15
Titik pusat (x,y) dan jejari lingkaran 2 : -15 4 20
Titik sembarang (x,y) : 0 0
Titik di dalam lingkaran 1 dan 2
```

```
PS C:\MyFiles\Visual Studio Projects HERE\alpro2\day3\3> go run unguided1.go
Titik pusat (x,y) dan jejari lingkaran 1 : 1 1 5
Titik pusat (x,y) dan jejari lingkaran 2 : 8 8 4
Titik sembarang (x,y) : 15 20
Titik di luar lingkaran 1 dan 2
```

Unguided 1 | Penjelasan

Program di atas merupakan program yang dapat menentukan apakah suatu titik berada di dalam suatu lingkaran dalam peta kartesius. Proses dimulai dengan input untuk jejari dan titik pusat dari kedua lingkaran, disusul dengan posisi x, y dari titik sembarang itu sendiri. Setelah input didapatkan, maka program akan memanggil fungsi `didalam`. Hasil dari fungsi tersebut akan berupa true atau false (boolean) dan disimpan dan 2 variabel berbeda (hasil1 untuk pengecekan lingkaran dan hasil 2 untuk pengecekan lingkaran 2). Jika kedua variabel bernilai true, maka titik berada di dalam kedua lingkaran. Jika hanya hasil1 yang bernilai true, maka titik berada di lingkaran 1. Jika hanya hasil2, maka titik berada di lingkaran 2. Dan jika kondisi tadi tidak terpenuhi semua, maka titik berada di luar lingkaran 1 dan lingkaran 2.

Fungsi `didalam` adalah sebuah fungsi yang dapat menentukan apakah suatu titik berada di dalam suatu lingkaran. Hal ini dapat dilakukan dengan menghitung jarak titik pusat dengan jejari lingkaran. Untuk menghitung jarak tersebut, program akan memanggil 2 fungsi lain, yaitu `akar` dan `jarak`.

Fungsi jarak merupakan fungsi yang digunakan untuk menghitung jarak titik pusat terhadap jejari. Dalam konteks program ini, a dan b adalah parameter untuk menyimpan koordinat titik sembarang, sementara c dan d adalah untuk menyimpan koordinat titik pusat lingkaran. Perhitungan dilakukan dengan mengonversi tipe data a, b, c, d dari int menjadi float64. Lalu, perhitungan dilakukan dengan rumus $(a - c)^2 + (b - d)^2$. Setelah itu, hasil perhitungan akan dikembalikan.

Fungsi akar merupakan fungsi yang dapat menentukan akar dari suatu bilangan. Dalam konteks program ini, fungsi digunakan untuk menghitung akar kuadrat dari jarak. Proses perhitungannya sendiri dilakukan dengan menggunakan fungsi `Sqrt()` dari library `math`. Setelah hasil ditemukan, maka fungsi akan mengembalikan nilainya.