LAPORAN PRAKTIKUM
PEMROGRAMAN BERORIENTASI OBJEK

MODUL III
FUNGSI

Oleh :

NAMA : FAISAL KHOIRUDDIN

NIM : 2311102046

Kelas : IF-11-02

S1 TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

I. DASAR TEORI

1. Definisi Function

Fungsi merupakan satu kesatuan rangkaian instruksi yang memberikan atau menghasilkan suatu nilai dan biasanya memetakkan input ke suatu nilai yang lain. Oleh karena itu, fungsi selalu menghasilkan/mengembalikan nilai. Suatu subprogram dikatakan fungsi apabila::

1. Ada deklarasi tipe nilai yang dikembalikan, dan
2. Terdapat kata kunci return dalam badan subprogram.

Maka fungsi digunakan jika suatu nilai biasanya diperlukan, seperti:

- Assignment nilai ke suatu variabel
- Bagian dari ekspresi
- Bagian dari argumen suatu subprogram, dsb.

Karena itu selalu pilih nama fungsi yang menggambarkan nilai, seperti kata benda dan kata sifat. Contoh nama-nama fungsi: median, rerata, nilaiTerbesar, ketemu, selesai, …

2. Deklarasi Function

Deklarasi fungsi sama dengan prosedur, yaitu berada pada blok yang terpisah dengan program utama.

Notasi Algoritma

```
function <nama function> (<params>) -> <type>
kamus
    {deklarasi variabel lokal dari fungsi}
    ...
algoritma
    {badan algoritma fungsi}
    ...
    return <value/variabel>
endfunction
```

Notasi dalam bahasa Go

```
func <nama function> (<params>) <type> {

    /* deklarasi variabel lokal dari fungsi */
    ...
    /* badan algoritma fungsi*/
    ...
    return <value/variabel>

}
```

Pada bagian deklarasi terlihat setelah parameter terdapat tipe data dari nilai yang dikembalikan, sedangkan pada bagian badan fungsi terdapat return dari nilai yang dikembalikan. Berikut adalah contoh fungsi untuk menghitung volume dari tabung apabila jari-jari alas dan tinggi tabung diketahui.

Notasi Algoritma

```
function volumeTabung(jari_jari,tinggi : integer) -> real
kamus
    luasAlas, volume: real
algoritma
    luasAlas <- 3.14 * (jari_jari * jari_jari)
    volume <- luasAlas * tinggi
    return volume
endfunction
```

Notasi dalam bahasa Go

```
func volumeTabung(jari_jari,tinggi int) float64 {
    var luasAlas,volume float64
    luasAlas = 3.14 * float64(jari_jari * jari_jari)
    volume = luasAlas * tinggi
    return volume
}
```


3. Cara Pemanggilan Function

Sama halnya dengan prosedur, pemanggilan fungsi cukup dilakukan dengan penulisan nama fungsi beserta argumen yang diminta oleh parameter dari fungsi. Perbedaannya dengan prosedur adalah fungsi bisa di-assign ke suatu variabel, menjadi bagian dari ekspresi, dan argumen dari suatu subprogram

Notasi Algoritma

```
1   program ContohProsedur
2   kamus
3       r,t : integer
4       v1,v2 : real
5   algoritma
6       r <- 5;
7       t <- 10
8       v1 <- volumeTabung(r,t)                          {cara pemanggilan #1}
9       v2 <- volumeTabung(r,t) + volumeTabung(15,t)     {cara pemanggilan #2}
10      output(volumeTabung(14,100))                     {cara pemanggilan #3}
11  endprogram
```

Notasi dalam bahasa Go

```
12  func main() {
13      var r,t int
14      var v1,v2 float64
15      r = 5
16      t = 10
17      v1 = volumeTabung(r,t)                          // cara pemanggilan #1
18      v2 = volumeTabung(r,t) + volumeTabung(15,t)     // cara pemanggilan #2
19      fmt.Println(volumeTabung(14,100))               // cara pemanggilan #3
20  }
```

Pada contoh pemanggilan fungsi di atas terlihat tidak ada perbedaan pada saat pemanggilan fungsi pada pseudocode ataupun GoLang. Di sini terlihat fungsi bisa di-assign ke suatu variabel pada saat pemanggilan, bisa dioperasikan sesuai dengan tipe data yang dikembalikan, dan juga bisa langsung ditampilkan dengan perintah output ataupun print.

4. Contoh Program dengan Function

Berikut ini adalah contoh penulisan fungsi pada suatu program lengkap. Buatlah sebuah program beserta fungsi yang digunakan untuk menghitung nilai faktorial dan permutasi.

Masukan terdiri dari dua buah bilangan positif a dan b.

Keluaran berupa sebuah bilangan bulat yang menyatakan nilai a permutasi b apabila a ≥ b atau b pemutasi a untuk kemungkinan yang lain.

1. Bentuk If-Else

Berikut ini bentuk-bentuk **if-else** yang mungkin dilakukan dalam

bahasa Go. Semua bentuk di bawah merupakan satu instruksi **if-else-endif** saja (hanya satu **endif**). Bentuk **if-else** yang bersarang (dengan beberapa **endif**) dapat dibentuk dengan komposisi beberapa **if-else-endif** tersebut.

	Notasi algoritma	Penulisan dalam bahasa Go

1
2
3
	if (kondisi) then
	  .. kode untuk kondisi true
	endif
		if kondisi {
		  .. kode untuk kondisi true
		}

4
5
6
7
8
	if (kondisi) then
	  .. kode untuk kondisi true
	else
	  .. kode untuk kondisi false
	endif
		if kondisi {
		  .. kode untuk kondisi true
		} else {
		  .. kode untuk kondisi false
		}

9
10
11
12
13
14
15
16
17
	if (kondisi-1) then
	  .. kode untuk kondisi-1 true
	else if (kondisi-2) then
	  .. kode untuk kondisi-2 true
	.. dst. dst.
	else
	  .. kode jika semua kondisi
	  .. di atas false
	endif
		if kondisi_1 {
		  .. kode untuk kondisi_1 true
		} else if kondisi_2 {
		  .. kode untuk kondisi_2 true
		.. dst. dst.
		} else {
		  .. kode jika semua kondisi
		  .. di atas false
		}

Contoh konversi (nilai, tubes, kehadiran) menjadi indeks nilai.

```
1  if nilai > 75 && adaTubes {
2      indeks = 'A'
3  } else if nilai > 65 {
4      indeks = 'B'
5  } else if nilai > 50 && pctHadir > 0.7 {
6      indeks = 'C'
7  } else {
8      indeks = 'F'
9  }
10 fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes=%v, mendapat
11 indeks %c\n", nilai, pctHadir, adaTubes, indeks )
```

Contoh konversi (nilai, tubes, kehadiran) menjadi indeks nilai

```
1  if nilai > 75 && adaTubes {
2      indeks = 'A'
3  } else if nilai > 65 {
4      indeks = 'B'
5  } else if nilai > 50 && pctHadir > 0.7 {
6      indeks = 'C'
7  } else {
8      indeks = 'F'
9  }
10 fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes=%v, mendapat
11 indeks %c\n", nilai, pctHadir, adaTubes, indeks )
```

2. Bentuk Switch-Case

Dalam bahasa Go ada dua variasi bentuk switch-case. Bentuk yang biasa digunakan adalah ekspresi ditulis pada perintah **switch** dan nilai ditulis dalam setiap label **case**-nya. Bentuk yang kedua mempunyai **switch** tanpa ekspresi, tetapi setiap **case** boleh berisi ekspresi **boolean.** Tentunya bentuk yang kedua lebih bersifat umum, dan merupakan penyederhanaan bentuk (atau alias dari) susunan suatu **if-elseif-..-else-endif.**

	Notasi algoritma	Penulisan dalam bahasa Go

1
2
3
4
5
6
7
8
9
10
	depend on expresi
	  nilai_1:
	  .. kode jika ekspresi bernilai_1
	  nilai_2:
	  .. kode jika ekspresi bernilai_2
	  .. dst. dst.
	}
		switch ekspresi {
		case nilai_1:
		  .. kode jika ekspresi bernilai_1
		case nilai_2:
		  .. kode jika ekspresi bernilai_2
		.. dst. dst.
		default:
		  .. kode jika tidak ada nilai
		  .. yang cocok dengan ekspresi
		}

11
12
13
14
15
16
17
18
19
20
	depend on (daftar variabel)
	  kondisi_1:
	  .. kode jika ekspresi_1 true
	  kondisi_2:
	  .. kode jika ekspresi_2 true
	  .. dst. dst.
	}
	
		switch {
		case kondisi_1:
		  .. kode jika ekspresi_1 true
		case kondisi_2:
		  .. kode jika ekspresi_2 true
		.. dst. dst.
		default:
		  .. jika tidak ada ekspresi
		  .. yang bernilai true
		}

Contoh menentukan batas nilai untuk suatu indeks:

```
1  switch indeks {
2  case 'A':
3      batasA = 100
4      batasB = 75
5  case 'B':
6      batasA = 75
7      batasB = 65
8  case 'C':
9      batasA = 65
10     batasB = 50
11 default:
12     batasA = 50
13     batasB = 0
14 }
15 fmt.Printf( "Rentang nilai %v adalah: %v..%v\n", indeks, batasB, batasA )
```

```
16 switch {
17 case nilai > 75 && adaTubes:
18     indeks = 'A'
```

```
19 case nilai > 65:
20     indeks = 'B'
21 case nilai > 50 && pctHadir > 0.7:
22     indeks = 'C'
23 default:
24     indeks = 'F'
25 }
26 fmt.Printf( "Nilai %v dengan kehadiran %v%% dan buat tubes=%v, mendapat
27 indeks %c\n", nilai, pctHadir, adaTubes, indeks )
```

II. GUIDED

1. Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? (tidak tentunya ya :p) **Masukan** terdiri dari empat buah bilangan asli a, b, c, dan d yang dipisahkan oleh spasi, dengan syarat a ≥ c dan b ≥ d.

Keluaran terdiri dari dua baris. Baris pertama adalah hasil permutasi dan kombinasi **a** terhadap c, sedangkan baris kedua adalah hasil permutasi dan kombinasi b terhadap d. Catatan: permutasi (P) dan kombinasi (C) dari n terhadap r (n ≥ r) dapat dihitung dengan menggunakan persamaan berikut!

$$P(n,r) = \frac{n!}{(n-r)!}, \text{ sedangkan } C(n,r) = \frac{n!}{r!(n-r)!}$$

Contoh

No	Masukan	Keluaran	Penjelasan
1	5 10 3 10	60 10 3628800 1	P(5,3) = 5!/2! = 120/2 = 60 C(5,3) = 5!/(3!x2!) = 120/12 = 10 P(10,10) = 10!/0! = 3628800/1 = 3628800 C(10,10) = 10!/(10!x0!) = 10!/10! = 1
2	8 0 2 0	56 28 1 1	

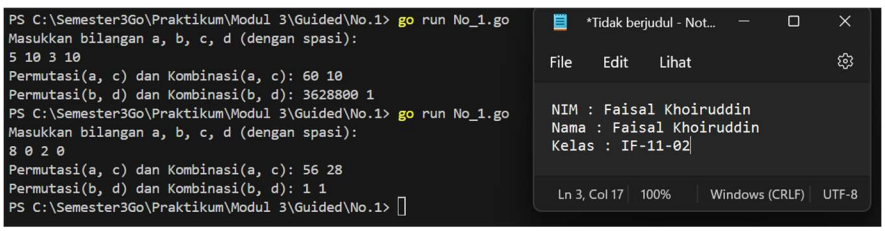
Source Code

```
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.")
    }
}
```

Screenshots Output



Deskripsi:

Program tersebut merupakan program menghitung permutasi dan kombinasi. Program tersebut meminta pengguna untuk memasukkan empat buah bilangan asli dipisahkan oleh spasi dengan syarat a ≥ c dan b ≥ d. Output dari program tersebut yaitu baris pertama hasil permutasi dan kombinasi a terhadap c dan baris kedua hasil permutasi dan kombinasi b terhadap d

- package main ➔ paket utama program golang
- import "fmt" ➔ mengimpor fmt
- func factorial(n int) int { ➔ fungsi untuk menghitung faktorial
- if n == 0 || n == 1 { ➔ jika n sama dengan 0 atau n sama dengan 1
- return 1 ➔ mengembalikan nilai 1
- result := 1 ➔ deklarasi variabel hasil
- for i := 2; i <= n; i++ { ➔ perulangan dari 2 sampai ke- n
- result *= i ➔ mengalikan hasil dengan i
- return result ➔ mengembalikan hasil faktorial
- func permutation(n, r int) int { ➔ fungsi untuk menghitung permutasi
- return factorial(n) / factorial(n-r) ➔ mengembalikan hasil permutasi
- func combination(n, r int) int { ➔ fungsi untuk menghitung kombinasi
- return factorial(n) / (factorial(r) * factorial(n-r)) ➔ mengembalikan hasil kombinasi
- func main() { ➔ merupakan fungsi utama

- var a, b, c, d int ➔ deklarasi variabel a, b, c, d dengan tipe data integer
- fmt.Println("Masukkan bilangan a, b, c, d (dengan spasi): ") ➔ meminta pengguna Masukkan bilangan a, b, c, d (dengan spasi)
- fmt.Scanf("%d %d %d %d", &a, &b, &c, &d) ➔ membaca input dan menyimpan ke variabel a, b, c, d
- if a >= c && b >= d { ➔ percabangan jika a >= c && b >= d
- permutasiAC := permutation(a, c) ➔ menghitung permutasi a c
- kombinasiAC := combination(a, c) ➔ menghitung permutasi a c
- permutasiBD := permutation(b, d) ➔ menghitung permutasi b d
- kombinasiBD := combination(b, d) ➔ menghitung permutasi b d
- fmt.Println("Permutasi(a, c) dan Kombinasi(a, c):", permutasiAC, kombinasiAC) ➔ menampilkan hasil permutasi dan kombinasi a c
- fmt.Println("Permutasi(b, d) dan Kombinasi(b, d):", permutasiBD, kombinasiBD) ➔ menampilkan hasil permutasi dan kombinasi a c
- } else { ➔ kalua tidak
- fmt.Println("Syarat a >= c dan b >= d tidak terpenuhi.") ➔ menampilkan Syarat a >= c dan b >= d tidak terpenuhi

III. Unguided

1. [**Lingkaran**] Suatu lingkaran didefinisikan dengan koordinat titik pusat (*cx ,cy*) dengan radius *r.* Apabila diberikan dua buah lingkaran, maka tentukan posisi sebuah titik sembarang (*x ,y*) berdasarkan dua lingkaran tersebut.

Masukan terdiri dari beberapa tiga baris. Baris pertama dan kedua adalah koordinat titik pusat dan radius dari lingkaran 1 dan lingkaran 2, sedangkan baris ketiga adalah koordinat titik sembarang. Asumsi sumbu x dan y dari semua titik dan juga radius direpresentasikan

dengan bilangan bulat.

Keluaran berupa string yang menyatakan posisi titik "**Titik di dalam lingkaran 1 dan 2**", "**Titik di dalam lingkaran 1**", "**Titik di dalam lingkaran 2**", atau "**Titik di luar lingkaran 1 dan 2**".

Contoh

No	Masukan	Keluaran
1	1 1 5 8 8 4 2 2	Titik di dalam lingkaran 1
2	1 2 3 4 5 6 7 8	Titik di dalam lingkaran 2
3	5 10 15 -15 4 20 0 0	Titik di dalam lingkaran 1 dan 2
4	1 1 5 8 8 4 15 20	Titik di luar lingkaran 1 dan 2

Fungsi untuk menghitung jarak titik (a, b) dan (c, d) dimana rumus jarak adalah:

$$jarak = \sqrt{(a-c)^2 + (b-d)^2}$$

dan juga fungsi untuk menentukan posisi sebuah titik sembarang berada di dalam suatu lingkaran atau tidak.

```
function jarak(a,b,c,d : real) -> real
{Mengembalikan jarak antara titik (a,b) dan titik (c,d)}
function didalam(cx,cy,r,x,y : real) -> boolean
{Mengembalikan true apabila titik (x,y) berada di dalam lingkaran yang memiliki titik pusat (cx,cy) dan radius r}
```


Catatan: Lihat paket math dalam lampiran untuk menggunakan fungsi **math.Sqrt()** untuk menghitung akar kuadrat

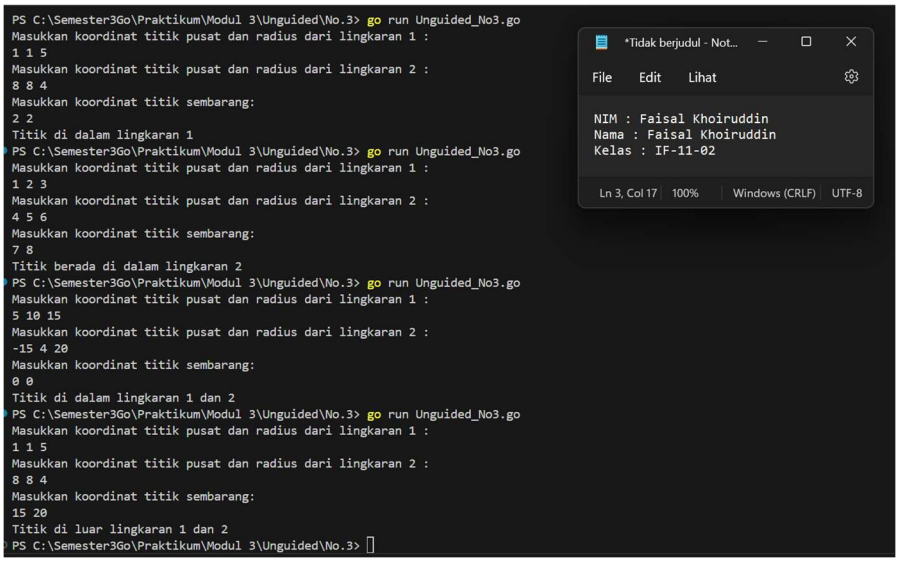
Source Code

```
package main

import (
    "fmt"
    "math"
)

func jarak(cx, cy, x, y float64) float64 {
    return math.Sqrt((x-cx)*(x-cx) + (y-
     cy)*(y-cy))
}

func beradaDalamLingkaran(cx, cy, r, x, y
     float64) bool {
    return jarak(cx, cy, x, y) <= r
}

func main() {
    var cx1, cy1, r1 float64
    var cx2, cy2, r2 float64
    var x, y float64

    fmt.Println("Masukkan koordinat titik
     pusat dan radius dari lingkaran 1 : ")
    _, err := fmt.Scanf("%f %f %f\n", &cx1,
     &cy1, &r1)
    if err != nil {
        fmt.Println("Terjadi kesalahan saat
     membaca input lingkaran 1: ", err)
        return
```

```
    }
    fmt.Println("Masukkan koordinat titik
     pusat dan radius dari lingkaran 2 : ")
    _, err = fmt.Scanf("%f %f %f\n", &cx2,
     &cy2, &r2)
    if err != nil {
        fmt.Println("Terjadi kesalahan saat
     membaca input lingkaran 2: ", err)
        return
    }
    fmt.Println("Masukkan koordinat titik
     sembarang: ")
    _, err = fmt.Scanf("%f %f\n", &x, &y)
    if err != nil {
        fmt.Println("Terjadi kesalahan saat
     membaca input titik sembarang: ", err)
        return
    }

    diDalamLingkaran1 :=
     beradaDalamLingkaran(cx1, cy1, r1, x,
     y)
    diDalamLingkaran2 :=
     beradaDalamLingkaran(cx2, cy2, r2, x,
     y)

    if diDalamLingkaran1 &&
     diDalamLingkaran2 {
        fmt.Println("Titik di dalam
     lingkaran 1 dan 2")
```

```
    } else if diDalamLingkaran1 {
        fmt.Println("Titik di dalam
     lingkaran 1")
    } else if diDalamLingkaran2 {
        fmt.Println("Titik berada di dalam
     lingkaran 2")
    } else {
        fmt.Println("Titik di luar
     lingkaran 1 dan 2")
    }
}
```

Screenshots Output



Deskripsi:

Program tersebut merupakan program mengecek sebuah titik berada di dalam satu atau dua lingkaran. Program tersebut meminta prngguna untuk menginput koordinat titik pusat dan radius dari lingkaran satu, menginput koordinat titik pusat dan radius dari lingkaran dua, dan menginput koordinat titik sembarang. Output dari program tersebut yaitu menampikan apakah titik berada di dalam

satu lingkaran, kedua lingkaran, atau di luar keduanya.

- package main ➔ paket utama program golang
- import
- "fmt" ➔ mengimpor fmt
- "math" ➔ mengimpor math
- func jarak(cx, cy, x, y float64) float64 { ➔ fungsi untuk menghitung jarak antara dua titik dengan parameter cx, cy, x, y denga tipe data float64
- return math.Sqrt((x-cx)*(x-cx) + (y-cy)*(y-cy)) ➔ mengembalikan nilai jarak antara dua titik
- func beradaDalamLingkaran(cx, cy, r, x, y float64) bool { ➔ fungsi untuk mengecek apakah titik berada di dalam lingkaran
- return jarak(cx, cy, x, y) <= r ➔ mengembalikan nilai dari jarak
- func main() { ➔ merupakan fungsi utama
- var cx1, cy1, r1 float64 ➔ deklarasi variabel untuk lingkaran 1 dengan tipe data float64
- var cx2, cy2, r2 float64 ➔ deklarasi variabel untuk lingkaran 2 dengan tipe data float64
- var x, y float64 ➔ deklarasi variabel titik sembarang
- fmt.Println("Masukkan koordinat titik pusat dan radius dari lingkaran 1 : ") ➔ meminta pengguna untuk input koordinat titik pusat dan radius dari lingkaran 1
- _, err := fmt.Scanf("%f %f %f\n", &cx1, &cy1, &r1) ➔ membaca input untuk lingkaran 1
- if err != nil { ➔ percabangan if jika ada error

- fmt.Println("Terjadi kesalahan saat membaca input lingkaran 1: ", err) ➔ menampilkan statement error jika terjadi kesalahan
- return ➔ return
- fmt.Println("Masukkan koordinat titik pusat dan radius dari lingkaran 2 : ") ➔ meminta pengguna untuk input koordinat titik pusat dan radius dari lingkaran 1
- _, err = fmt.Scanf("%f %f %f\n", &cx2, &cy2, &r2) ➔ membaca input untuk lingkaran 2
- if err != nil { ➔ percabangan if jika ada error
- fmt.Println("Terjadi kesalahan saat membaca input lingkaran 2: ", err) ➔ menampilkan statement error jika terjadi kesalahan
- return ➔ return
- fmt.Println("Masukkan koordinat titik sembarang: ") ➔ meminta pengguna untuk input koordinat titik sembarang
- _, err = fmt.Scanf("%f %f\n", &x, &y) ➔ membaca input untuk koordinat titik sembarang
- if err != nil { ➔ percabangan if jika ada error
- fmt.Println("Terjadi kesalahan saat membaca input titik sembarang: ", err) ➔ menampilkan statement error jika terjadi kesalahan
- return ➔ return
- diDalamLingkaran1 := beradaDalamLingkaran(cx1, cy1, r1, x, y) ➔ mengecek apakah titik berada di dalam lingkaran satu
- diDalamLingkaran2 := beradaDalamLingkaran(cx2, cy2, r2, x, y) ➔ mengecek apakah titik berada di dalam lingkaran dua

- if diDalamLingkaran1 && diDalamLingkaran2 { ➔ percabangan if jika diDalamLingkaran1 dan diDalamLingkaran2
- fmt.Println("Titik di dalam lingkaran 1 dan 2") ➔ menampilkan pesan Titik di dalam lingkaran 1 dan 2
- } else if diDalamLingkaran1 { ➔ kalau tidak jika diDalamLingkaran1
- fmt.Println("Titik di dalam lingkaran 1") ➔ menampilkan pesan Titik di dalam lingkaran 1
- } else if diDalamLingkaran2 { ➔ kalau tidak jika diDalamLingkaran2
- fmt.Println("Titik berada di dalam lingkaran 2") ➔ menampilkan pesan Titik di dalam lingkaran 2
- } else { ➔ kalua tidak
- fmt.Println("Titik di luar lingkaran 1 dan 2") ➔ menampilkan pesan titik di luar lingkaran 1 dan 2