

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2**

**MODUL 3 & 4
FUNGSI DAN PROSEDUR**



Oleh:

IQBAL BAWANI

2311102130

S1IF-11-07

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

1. FUNGSI

Fungsi merupakan satu kesatuan indtruksi yang memberikan atau menghasilkan suatu nilai dan biasanya memetakan input ke suatu nilai yang lain. Oleh karena itu, fungsi selalu menghasilkan/mengembalikan nilai.

Syarat fungsi yaitu : Ada deklarasi tipe nilai yang di kembalikan , dan tada 'return' dalam badfan program bagian. Maka fungsi dapat di pakai jika nilai yang di perlukan :

- Assigment nilai ke suatu variable
- Bagian dari ekspresi
- Bagian dari argument dustu dub progrsm

Fungsi dalam Go dapat dibedakan menjadi beberapa jenis berdasarkan penggunaannya. Pertama, fungsi tanpa parameter dan nilai balik, yang digunakan untuk melakukan aksi sederhana seperti mencetak pesan. Contoh fungsi ini adalah `func sapaan()`, yang mencetak sapaan tanpa menerima input dari pengguna. Kedua, fungsi dengan parameter, yang memungkinkan pengguna untuk memberikan input yang diperlukan untuk proses lebih lanjut. Misalnya, `func sapaanDenganNama(nama string)` menerima nama sebagai parameter dan mencetak sapaan yang sesuai. Terakhir, fungsi dengan nilai balik, yang mengembalikan hasil dari suatu proses untuk digunakan di bagian lain dari program. Sebagai contoh, fungsi `func tambahAngka(a int, b int) int` mengembalikan hasil penjumlahan dua angka.

2. PROSEDUR

Prosedur, dalam konteks pemrograman, merujuk pada blok kode yang dapat dieksekusi secara berulang untuk menjalankan tugas tertentu. Dalam bahasa Go, prosedur diimplementasikan sebagai fungsi menggunakan kata kunci `func`. Fungsi ini dapat menerima parameter untuk memproses data dan mengembalikan nilai setelah eksekusi. Konsep ini sangat penting dalam pengembangan perangkat lunak karena memungkinkan pengorganisasian kode yang lebih baik, memudahkan pemeliharaan, dan meningkatkan keterbacaan. Dengan membagi kode menjadi fungsi-fungsi yang lebih kecil, programmer dapat fokus pada satu aspek pada suatu waktu, yang meningkatkan efisiensi pengembangan.

Penggunaan prosedur dalam bahasa Go tidak hanya meningkatkan modularitas kode, tetapi juga memungkinkan pengujian yang lebih mudah. Fungsi dapat diuji secara terpisah dari bagian lain dalam program, yang membantu dalam mendeteksi dan memperbaiki bug. Dengan menyediakan cara untuk mengelompokkan operasi yang terkait, fungsi membantu menjaga

kode tetap bersih dan terorganisir. Selain itu, Go mendukung fitur-fitur canggih seperti fungsi sebagai nilai, yang memungkinkan programmer untuk mengoper fungsi sebagai argumen ke fungsi lain, memberikan fleksibilitas dan kekuatan lebih dalam pengembangan aplikasi.

II. GUIDED

Guided1 (fungsi)

Source code:

```
package main

import (
    "fmt"
)

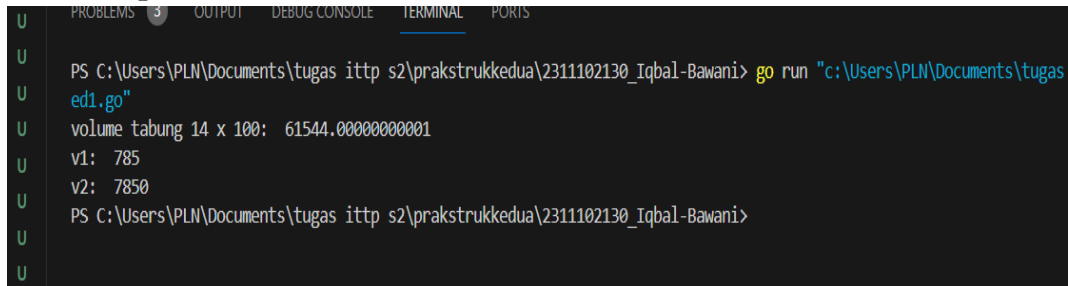
func volumetabung(jar_jari, tinggi int) float64 {
    var luas_alas, volume float64
    luas_alas = 3.14 * float64(jar_jari*jar_jari)
    volume = luas_alas * float64(tinggi)
    return volume
}

func main() {
    var r, t int
    var v1, v2 float64
    r = 5
    t = 10

    v1 = volumetabung(r, t)
    v2 = volumetabung(r, t) + volumetabung(15, t)

    fmt.Println("volume tabung 14 x 100: ", volumetabung(14,
100))
    fmt.Println("v1: ", v1)
    fmt.Println("v2: ", v2)
}
```

Output =

A screenshot of a terminal window showing the output of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, and TERMINAL. The command executed is 'go run "c:\Users\PLN\Documents\tugas ed1.go"'. The output shows the volume of a cylinder with radius 14 and height 100 as 61544.00000000001, followed by two additional calculations for different radii (5 and 15) with the same height (10), resulting in 785 and 7850 respectively.

```
PS C:\Users\PLN\Documents\tugas ed1> go run "c:\Users\PLN\Documents\tugas ed1.go"
volume tabung 14 x 100: 61544.00000000001
v1: 785
v2: 7850
PS C:\Users\PLN\Documents\tugas ed1>
```

Penjelasan =

Program di atas mendemonstrasikan penggunaan fungsi dalam bahasa Go untuk menghitung volume tabung. Fungsi `volumetabung` didefinisikan untuk menerima dua parameter: `jar_jari` dan `tinggi`, keduanya bertipe integer. Di dalam fungsi, luas alas tabung dihitung menggunakan rumus $\pi \times r^2$ (di mana π didekati dengan 3.14) dan hasilnya dikonversi menjadi tipe float64 untuk mendapatkan hasil yang lebih presisi. Kemudian, volume tabung dihitung dengan mengalikan luas alas dengan tinggi dan mengembalikan hasilnya sebagai nilai float64.

Di dalam fungsi `main`, variabel `r` dan `t` diinisialisasi dengan nilai 5 dan 10, masing-masing, dan fungsi `volumetabung` dipanggil untuk menghitung volume tabung dengan jarak jari 5 dan tinggi 10. Hasil volume ini disimpan dalam variabel `v1`. Selain itu, `v2` juga dihitung dengan menambahkan volume dari tabung dengan jari-jari 15 dan tinggi 10. Program ini mencetak volume dari tabung dengan jarijari 14 dan tinggi 100, serta nilai dari `v1` dan `v2`. Dengan demikian, fungsi `volumetabung` menyediakan cara yang efisien dan terorganisir untuk menghitung volume tabung berdasarkan parameter yang diberikan, mendemonstrasikan kekuatan dan fleksibilitas fungsi dalam pemrograman Go.

Guided 2 (Procedure)

Source code :

```
package main

import (
    "fmt"
)

func cetaknifbon(n int) {
    var f1, f2, f3 int
```

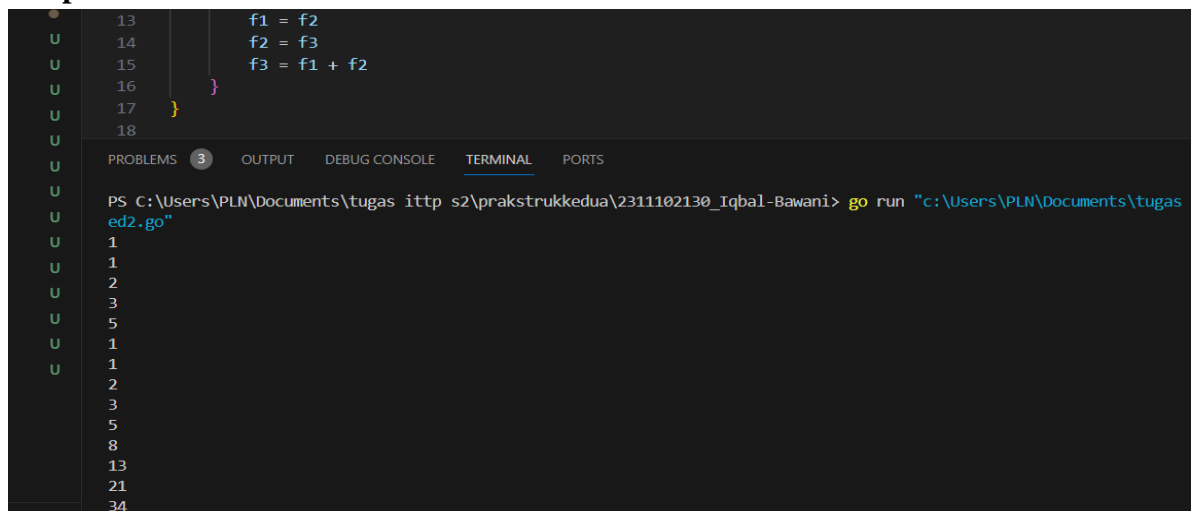
```

    f2 = 0
    f3 = 1
    for i := 1; i <= n; i++ {
        fmt.Println(f3)
        f1 = f2
        f2 = f3
        f3 = f1 + f2
    }
}

func main() {
    var x int
    x = 5
    cetaknifbon(x)
    cetaknifbon(100)
}

```

Output =



```

13     f1 = f2
14     f2 = f3
15     f3 = f1 + f2
16 }
17 }
18
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> go run "c:\Users\PLN\Documents\tugas
ed2.go"
1
1
2
3
5
1
1
2
3
5
8
13
21
34

```

Penjelasan =

Program di atas menggunakan prosedur yang diimplementasikan dalam bentuk fungsi untuk mencetak deret Fibonacci. Fungsi `cetaknifbon` berperan sebagai prosedur utama yang bertanggung jawab untuk menghasilkan dan mencetak suku-suku dalam deret Fibonacci. Prosedur ini menerima parameter `n`, yang menunjukkan jumlah suku Fibonacci yang ingin dicetak. Di dalamnya, tiga variabel `f1`, `f2`, dan `f3` digunakan untuk menyimpan nilai dari dua suku terakhir dan suku saat ini. Dengan menggunakan loop `for`, prosedur ini mencetak nilai `f3` dan memperbarui nilai-nilai variabel untuk menghasilkan suku berikutnya dalam deret.

Penggunaan prosedur dalam kode ini meningkatkan modularitas dan keterbacaan. Fungsi ``cetaknifbon`` dapat dipanggil berkali-kali dengan parameter yang berbeda untuk mencetak deret Fibonacci dengan panjang yang diinginkan tanpa perlu menulis ulang logika perhitungan. Pada bagian ``main``, prosedur ini dipanggil dua kali, pertama untuk mencetak lima suku Fibonacci dan kedua untuk mencetak seratus suku. Pendekatan ini memungkinkan programmer untuk dengan mudah memperluas atau memodifikasi perilaku pencetakan deret Fibonacci tanpa mengubah struktur dasar dari kode lainnya, mendemonstrasikan kekuatan dan efisiensi penggunaan prosedur dalam pemrograman Go.

III. UNGUIDED

Uguided ke1 modul 3 (Fungsi)

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 $a \geq c$ dan $b \geq d$.

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 ($n \geq r$) 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	$P(5, 3) = 5! / 2! = 120 / 2 = 60$ $C(5, 3) = 5! / (3! \times 2!) = 120 / 12 = 10$ $P(10, 10) = 10! / 0! = 3628800 / 1 = 3628800$ $C(10, 10) = 10! / (10! \times 0!) = 10! / 10! = 1$
2	8 8 2 8	56 28 1 1	

Selesaikan program tersebut dengan memanfaatkan subprogram yang diberikan berikut ini!

```
function faktorial(n: integer) → integer
{mengembalikan nilai faktorial dari n}

function permutasi(n, r : integer) → integer
{Mengembalikan hasil n permutasi r, dan n >= r}

function kombinasi(n, r : integer) → integer
```

Source code

```
//iqbal bawani
```

```
package main
```

```
import (
    "fmt"
)
```

```
func faktorial(n int) int {
    if n == 0 || n == 1 {
        return 1
    }
}
```

```

    result := 1
    for i := 2; i <= n; i++ {
        result *= i
    }
    return result
}

func permutasi(n, r int) int {
    return faktorial(n) / faktorial(n-r)
}

func kombinasi(n, r int) int {
    return faktorial(n) / (faktorial(r) * faktorial(n-r))
}

func main() {
    var a, b, c, d int

    fmt.Println("Masukkan nilai a, b, c, dan d (a >= c, b >= d):")
    fmt.Scan(&a, &b, &c, &d)

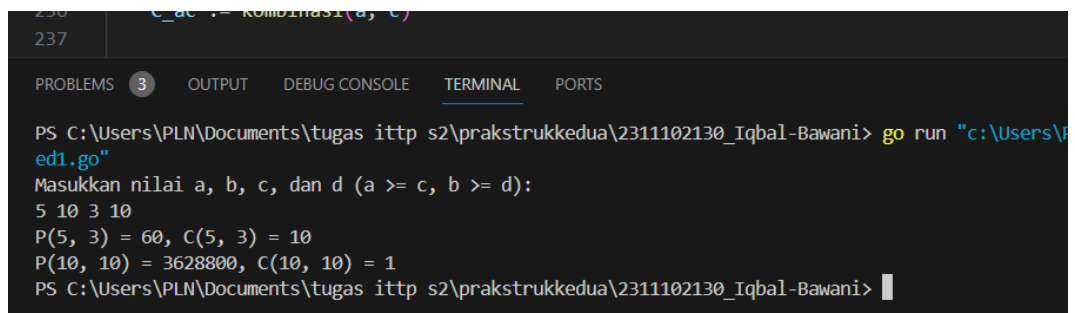
    P_ac := permutasi(a, c)
    C_ac := kombinasi(a, c)

    P_bd := permutasi(b, d)
    C_bd := kombinasi(b, d)

    fmt.Printf("P(%d, %d) = %d, C(%d, %d) = %d\n", a, c, P_ac, a, c,
C_ac)
    fmt.Printf("P(%d, %d) = %d, C(%d, %d) = %d\n", b, d, P_bd, b, d,
C_bd)
}

```

Output =



```

236      C_ac := kombinasi(a, c)
237
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> go run "c:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani\ed1.go"
Masukkan nilai a, b, c, dan d (a >= c, b >= d):
5 10 3 10
P(5, 3) = 60, C(5, 3) = 10
P(10, 10) = 3628800, C(10, 10) = 1
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani>

```

Penjelasan =

Program di atas ditulis dalam bahasa Go dan berfungsi untuk menghitung nilai permutasi dan kombinasi dari dua set nilai yang dimasukkan pengguna. Fungsi ``faktorial(n int) int`` bertanggung jawab untuk menghitung nilai faktorial dari suatu bilangan bulat non-negatif n . Faktorial didefinisikan sebagai hasil kali dari semua bilangan bulat positif hingga n . Untuk menghitung faktorial, fungsi ini menggunakan pendekatan iteratif, di mana nilai awal ``result`` diatur ke 1 dan dikalikan dengan setiap bilangan bulat dari 2 hingga n . Jika n adalah 0 atau 1, fungsi ini mengembalikan 1, sesuai dengan definisi faktorial.

Fungsi ``permutasi(n, r int) int`` dan ``kombinasi(n, r int) int`` menggunakan fungsi ``faktorial`` untuk menghitung permutasi dan kombinasi, masing-masing. Permutasi dihitung menggunakan rumus $P(n, r) = n! / (n-r)!$, di mana n adalah jumlah total item, dan r adalah jumlah item yang diambil. Kombinasi dihitung dengan rumus $C(n, r) = n! / (r!(n-r)!)$, yang menggambarkan cara memilih r item dari n item tanpa memperhatikan urutan. Dalam fungsi ``main``, pengguna diminta untuk memasukkan nilai a , b , c , dan d yang akan digunakan untuk menghitung permutasi dan kombinasi. Hasil perhitungan kemudian ditampilkan ke layar.

Unguided Ke 2 Modul 3(Fungsi)

Diberikan tiga buah fungsi matematika yaitu $f(x) = x^2$, $g(x) = x - 2$ dan $h(x) = x + 1$.

Fungsi komposisi $(f \circ g \circ h)(x)$ artinya adalah $f(g(h(x)))$. Tuliskan $f(x)$, $g(x)$ dan $h(x)$ dalam bentuk function.

Masukan terdiri dari sebuah bilangan bulat a , b dan c yang dipisahkan oleh spasi.

Keluaran terdiri dari tiga baris. Baris pertama adalah $(f \circ g \circ h)(a)$, baris kedua $(g \circ h \circ f)(b)$, dan baris ketiga adalah $(h \circ f \circ g)(c)$!

Contoh

No	Masukan	Keluaran	Penjelasan
1	7 2 10	36 3 65	$(f \circ g \circ h)(7) = 36$ $(g \circ h \circ f)(2) = 3$ $(h \circ f \circ g)(10) = 65$
2	5 5 5	16 24 10	$(f \circ g \circ h)(5) = 16$ $(g \circ h \circ f)(5) = 24$ $(h \circ f \circ g)(5) = 10$
3	3 8 4	4 63 5	$(f \circ g \circ h)(5) = 4$ $(g \circ h \circ f)(5) = 63$ $(h \circ f \circ g)(5) = 5$

Source code

```
//iqbalbawani
```

```
package main
```

```
import (  
    "fmt"  
)
```

```
func f(x int) int {  
    return x * x  
}
```

```
func g(x int) int {  
    return x - 2  
}
```

```
func h(x int) int {  
    return x + 1  
}
```

```
func fogoh(x int) int {
```

```

    return f(g(h(x)))
}

func gohof(x int) int {
    return g(h(f(x)))
}

func hofog(x int) int {
    return h(f(g(x)))
}

func main() {
    var a, b, c int

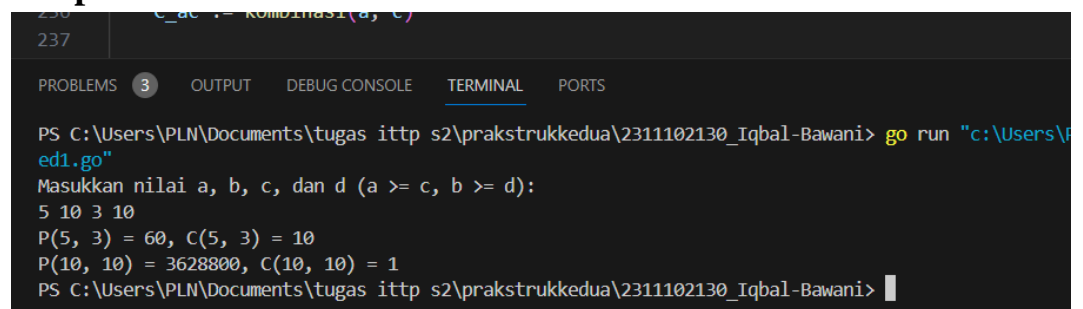
    fmt.Println("Masukkan tiga bilangan bulat a, b, dan c:")
    fmt.Scan(&a, &b, &c)

    result1 := fogoh(a)
    result2 := gohof(b)
    result3 := hofog(c)

    fmt.Printf("(fogoh)(%d) = %d\n", a, result1)
    fmt.Printf("(gohof)(%d) = %d\n", b, result2)
    fmt.Printf("(hofog)(%d) = %d\n", c, result3)
}

```

Output =



The screenshot shows a terminal window with the following output:

```

PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> go run "c:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani\ed1.go"
Masukkan nilai a, b, c, dan d (a >= c, b >= d):
5 10 3 10
P(5, 3) = 60, C(5, 3) = 10
P(10, 10) = 3628800, C(10, 10) = 1
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani>

```

Penjelasan =

Program ini ditulis dalam bahasa Go dan terdiri dari beberapa fungsi yang digunakan untuk mengimplementasikan komposisi fungsi. Tiga fungsi utama, yaitu `f`, `g`, dan `h`, masing-masing melakukan operasi matematis sederhana. Fungsi `f(x int) int` mengembalikan kuadrat dari bilangan `x`, fungsi `g(x int) int` mengurangi `x` dengan 2, dan fungsi `h(x int) int` menambahkan 1 ke `x`. Ketiga fungsi ini membentuk dasar untuk komposisi fungsi yang lebih kompleks.

Komposisi fungsi dilakukan melalui tiga fungsi baru: `fogoh`, `gohof`, dan `hofog`. Fungsi `fogoh(x int) int` melakukan komposisi $f(g(h(x)))$, yang berarti bahwa input pertama-tama diproses oleh `h`, lalu hasilnya diproses oleh `g`, dan akhirnya diproses oleh `f`. Fungsi `gohof(x int) int` menjalankan komposisi $g(h(f(x)))$, sedangkan `hofog(x int) int` menjalankan $h(f(g(x)))$. Di dalam fungsi `main`, pengguna diminta untuk memasukkan tiga bilangan bulat `a`, `b`, dan `c`, yang kemudian diproses melalui ketiga komposisi fungsi tersebut. Hasil dari masing-masing komposisi ditampilkan ke layar, menunjukkan bagaimana input awal diproses melalui rangkaian fungsi.

Unguided Ke 3 Modul 3 (Fungsi_

i. **[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	Titik di dalam lingkaran 2
	4 5 6 7 8	
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

```

// iqbalbawani

package main

import (
    "fmt"
    "math"
)

func jarak(x1, y1, x2, y2 int) float64 {
    return math.Sqrt(float64((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2)))
}

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

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

    fmt.Println("Masukkan koordinat pusat dan radius lingkaran 1:")
    fmt.Scan(&cx1, &cy1, &r1)

    fmt.Println("Masukkan koordinat pusat dan radius lingkaran 2:")
    fmt.Scan(&cx2, &cy2, &r2)

    fmt.Println("Masukkan koordinat titik sembarang:")
    fmt.Scan(&x, &y)

    diDalamL1 := diDalamLingkaran(cx1, cy1, r1, x, y)
    diDalamL2 := diDalamLingkaran(cx2, cy2, r2, x, y)

    if diDalamL1 && diDalamL2 {
        fmt.Println("Titik di dalam lingkaran 1 dan 2")
    } else if diDalamL1 {
        fmt.Println("Titik di dalam lingkaran 1")
    } else if diDalamL2 {
        fmt.Println("Titik di dalam lingkaran 2")
    } else {
        fmt.Println("Titik di luar lingkaran 1 dan 2")
    }
}

```

Output =

```
308     var cx2, cy2, r2 int
309     var x, y int
310
PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> go r
ed1.go"
Masukkan koordinat pusat dan radius lingkaran 1:
1 1 5
Masukkan koordinat pusat dan radius lingkaran 2:
8 8 4
Masukkan koordinat titik sembarang:
2 2
Titik di dalam lingkaran 1
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> |
```

Penjelasan =

Program ini ditulis dalam bahasa Go dan dirancang untuk menentukan apakah sebuah titik berada di dalam dua lingkaran yang ditentukan oleh pengguna. Fungsi utama dalam program ini adalah `jarak`, yang menghitung jarak antara dua titik dalam bidang dua dimensi menggunakan rumus jarak Euclidean. Fungsi ini menerima empat parameter, yaitu koordinat x1, y1 untuk titik pertama dan x2, y2 untuk titik kedua. Hasil jarak dihitung dengan mengkuadratkan selisih antara koordinat x dan y, kemudian dijumlahkan dan diambil akar kuadratnya, mengembalikan nilai dalam bentuk float64.

Fungsi `diDalamLingkaran` digunakan untuk menentukan apakah suatu titik (x, y) berada di dalam lingkaran dengan pusat (cx, cy) dan radius r. Fungsi ini memanggil fungsi `jarak` untuk menghitung jarak antara titik pusat lingkaran dan titik yang diberikan, kemudian membandingkan jarak tersebut dengan radius lingkaran. Di dalam fungsi `main`, pengguna diminta untuk memasukkan koordinat pusat dan radius dari dua lingkaran, serta koordinat titik yang ingin diperiksa. Program kemudian memanggil fungsi `diDalamLingkaran` untuk kedua lingkaran dan menampilkan hasil apakah titik tersebut berada di dalam lingkaran 1, lingkaran 2, kedua lingkaran, atau di luar keduanya.

Unguided Ke 1 Modul 4 (Prosedur)

1) Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika

Halaman 44 | Modul Praktikum Algoritma dan Pemrograman 2

diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersedia 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 $a \geq c$ dan $b \geq d$.

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 ($n \geq r$) 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	$P(5, 3) = 5! / 2! = 120 / 2 = 60$ $C(5, 3) = 5! / (3! \times 2!) = 120 / 12 = 10$ $P(10, 10) = 10! / 0! = 3628800 / 1 = 3628800$ $C(10, 10) = 10! / (10! \times 0!) = 10! / 10! = 1$
2	8 0 2 0	56 28 1 1	

Seselaikan program tersebut dengan memanfaatkan prosedur yang diberikan berikut ini!

```
procedure factorial(in n: integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n
 F.S. hasil berisi nilai faktorial dari n}

procedure permutation(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n permutasi r}

procedure combination(in n,r : integer, in/out hasil:integer)
{I.S. terdefinisi bilangan bulat positif n dan r, dan n >= r
 F.S. hasil berisi nilai dari n kombinasi r}
```

Source code

//iqbalbawani


```

package main

import "fmt"

func cetakderetbilangan(n int) {
    fmt.Print(n)
    for n != 1 {
        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
        fmt.Print(" ", n)
    }
    fmt.Println()
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakderetbilangan(n)
    } else {
        fmt.Println("Input harus bilangan bulat positif dan kurang dari
1000000.")
    }
}

// unguided1modul4
package main

import (
    "fmt"
)

func faktorial(n int, result *int) {
    if n == 0 || n == 1 {
        *result = 1
        return
    }
    *result = 1
    for i := 2; i <= n; i++ {
        *result *= i
    }
}

```

```

    }
}

func permutasi(n, r int, result *int) {
    var fn, fn_r int
    faktorial(n, &fn)
    faktorial(n-r, &fn_r)
    *result = fn / fn_r
}

func kombinasi(n, r int, result *int) {
    var fn, fr, fn_r int
    faktorial(n, &fn)
    faktorial(r, &fr)
    faktorial(n-r, &fn_r)
    *result = fn / (fr * fn_r)
}

func main() {
    var a, b, c, d int

    fmt.Println("Masukkan nilai a, b, c, dan d (a >= c, b >= d):")
    fmt.Scan(&a, &b, &c, &d)

    var P_ac, C_ac, P_bd, C_bd int

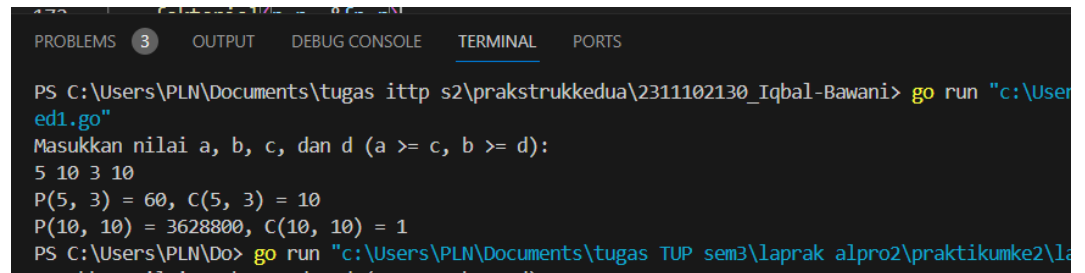
    permutasi(a, c, &P_ac)
    kombinasi(a, c, &C_ac)

    permutasi(b, d, &P_bd)
    kombinasi(b, d, &C_bd)

    fmt.Printf("P(%d, %d) = %d, C(%d, %d) = %d\n", a, c, P_ac, a, c,
C_ac)
    fmt.Printf("P(%d, %d) = %d, C(%d, %d) = %d\n", b, d, P_bd, b, d,
C_bd)
}

```

Output =

A screenshot of a terminal window showing the execution of a Go program. The terminal has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The command executed is 'go run "c:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani\ed1.go"'. The program prompts the user to 'Masukkan nilai a, b, c, dan d (a >= c, b >= d):'. The user inputs '5 10 3 10'. The program then outputs the results of the Collatz conjecture for n=5 and n=10, and the values of P(5, 3), C(5, 3), P(10, 10), and C(10, 10).

```
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> go run "c:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani\ed1.go"
Masukkan nilai a, b, c, dan d (a >= c, b >= d):
5 10 3 10
P(5, 3) = 60, C(5, 3) = 10
P(10, 10) = 3628800, C(10, 10) = 1
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani>
```

Penjelasan =

Program ini terdiri dari dua modul dalam bahasa Go. Modul pertama mencetak deret bilangan berdasarkan algoritma Collatz. Fungsi `cetakderetbilangan(n int)` menerima bilangan bulat positif `n` dan mencetak deret hingga `n` mencapai 1. Proses iteratif dilakukan dengan membagi `n` dengan 2 jika genap, atau menghitung $3n + 1$ jika ganjil, dan hasilnya dicetak setiap langkahnya.

Modul kedua menghitung permutasi dan kombinasi menggunakan prosedur dan pointer. Fungsi `faktorial(n int, result *int)` menghitung faktorial `n` dan menyimpan hasilnya dalam variabel yang direferensikan oleh `result`. Fungsi `permutasi(n, r int, result *int)` dan `kombinasi(n, r int, result *int)` menghitung $P(n, r)$ dan $C(n, r)$ dengan memanggil fungsi faktorial. Keduanya menggunakan pointer untuk menyimpan hasil perhitungan, menunjukkan penggunaan prosedur dan pointer dalam pemrograman Go.

Unguided Ke 2 Modul 3 (Prosedur)

!) Kompetisi pemrograman tingkat nasional berlangsung ketat. Setiap peserta diberikan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam saja. Peserta yang berhasil menyelesaikan soal paling banyak dalam waktu paling singkat adalah pemenangnya.

Buat program **gema** yang mencari pemenang dari daftar peserta yang diberikan. Program harus dibuat modular, yaitu dengan membuat prosedur `hitungSkor` yang mengembalikan total soal dan total skor yang dikerjakan oleh seorang peserta, melalui parameter formal. Pembacaan nama peserta dilakukan di program utama, sedangkan waktu pengerjaan dibaca di dalam prosedur.

procedure `hitungSkor(in/out soal, skor : integer)`

Setiap baris **masukan** dimulai dengan satu string nama peserta tersebut diikuti dengan adalah 8 integer yang menyatakan berapa lama (dalam menit) peserta tersebut menyelesaikan soal. Jika tidak berhasil atau tidak mengirimkan jawaban maka otomatis dianggap menyelesaikan dalam waktu 5 jam 1 menit (301 menit).

Satu baris **keluaran** berisi nama pemenang, jumlah soal yang diselesaikan, dan nilai yang diperoleh. Nilai adalah total waktu yang dibutuhkan untuk menyelesaikan soal yang berhasil diselesaikan.

No	Masukan	Keluaran
1	Astuti 20 50 301 301 61 71 75 10 Bertha 25 47 301 26 50 60 65 21 Selesai	Bertha 7 294

Keterangan:

Astuti menyelesaikan 6 soal dalam waktu 287 menit, sedangkan Bertha 7 soal dalam waktu 294 menit. Karena Bertha menyelesaikan lebih banyak, maka Bertha menang. Jika keduanya menyelesaikan sama banyak, maka pemenang adalah yang menyelesaikan dengan total waktu paling kecil.

Source code

```
//iqbalbawani

package main

import (
    "bufio"
    "fmt"
    "os"
    "strings"
)

func hitungSkor(soal []int, skor *int) {
```

```

        *skor = 0
        for _, waktu := range soal {
            if waktu <= 300 {
                *skor += waktu
            }
        }
    }

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Println("Masukkan data peserta dan soal (akhiri dengan 'Selesai'):")
    var masukan []string
    for scanner.Scan() {
        b := scanner.Text()
        if b == "Selesai" {
            break
        }
        masukan = append(masukan, b)
    }

    var pemenang string
    skorMaks := 1000000
    totalSelesai := 0

    for _, b := range masukan {
        kolom := strings.Fields(b)

        nama := kolom[0]

        soal := make([]int, 8)
        for i := 0; i < 8; i++ {
            fmt.Sscanf(kolom[i+1], "%d", &soal[i])
        }

        var skor int
        hitungSkor(soal, &skor)

        selesai := 0
        for _, waktu := range soal {
            if waktu <= 300 {
                selesai++
            }
        }
        if selesai > totalSelesai || (selesai == totalSelesai && skor < skorMaks) {

```

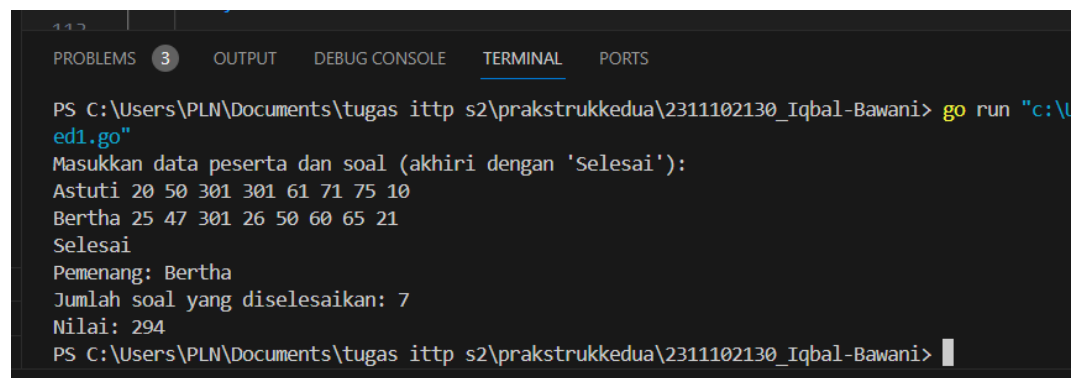
```

        pemenang = nama
        skorMaks = skor
        totalSelesai = selesai
    }
}

fmt.Printf("Pemenang: %s\nJumlah soal yang diselesaikan:
%d\nNilai: %d\n", pemenang, totalSelesai, skorMaks)
}

```

Output =



```

PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> go run "c:\U
ed1.go"
Masukkan data peserta dan soal (akhiri dengan 'Selesai'):
Astuti 20 50 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Selesai
Pemenang: Bertha
Jumlah soal yang diselesaikan: 7
Nilai: 294
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani>

```

Penjelasan =

Program ini ditulis dalam bahasa Go dan berfungsi untuk menentukan pemenang dari sebuah kompetisi berdasarkan waktu yang dibutuhkan peserta untuk menyelesaikan serangkaian soal. Di dalam program, terdapat fungsi ``hitungSkor(soal []int, skor *int)`` yang menghitung total skor peserta. Fungsi ini menerima dua parameter: slice bilangan bulat ``soal``, yang berisi waktu penyelesaian soal, dan pointer ``skor`` untuk menyimpan hasil skor. Di dalam fungsi, skor diinisialisasi menjadi 0, dan waktu setiap soal yang diselesaikan dalam waktu 300 detik atau kurang akan dijumlahkan untuk mendapatkan skor total peserta.

Dalam fungsi ``main``, pengguna diminta untuk memasukkan nama peserta beserta waktu yang diperlukan untuk menyelesaikan delapan soal, yang diakhiri dengan kata "Selesai". Data peserta disimpan dalam slice ``masukan``, dan setelah semua data dimasukkan, program melakukan iterasi untuk menghitung skor setiap peserta menggunakan fungsi ``hitungSkor``. Program juga menghitung jumlah soal yang diselesaikan dalam waktu yang sesuai. Dengan membandingkan jumlah soal yang diselesaikan dan skor, program menentukan pemenang dengan jumlah soal terbanyak dan, jika ada kesamaan, skor terendah. Hasil akhir menampilkan nama pemenang, jumlah soal yang diselesaikan, dan nilai total skor.

Unguided 3 Modul 4(Prosedur)

- I Skiena dan Revilla dalam *Programming Challenges* mendefinisikan sebuah deret bilangan. Deret dimulai dengan sebuah bilangan bulat n . Jika bilangan n saat itu genap, maka suku berikutnya adalah $\frac{1}{2}n$, tetapi jika ganjil maka suku berikutnya bernilai $3n+1$. Rumus yang sama digunakan terus menerus untuk mencari suku berikutnya. Deret berakhir ketika suku terakhir bernilai 1. Sebagai contoh jika dimulai dengan $n=22$, maka deret bilangan yang diperoleh adalah:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Untuk suku awal sampai dengan 1000000, diketahui deret selalu mencapai suku dengan nilai 1.

Buat program **skiena** yang akan mencetak setiap suku dari deret yang dijelaskan di atas untuk nilai suku awal yang diberikan. Pencetakan deret harus dibuat dalam prosedur cetakDeret yang mempunyai 1 parameter formal, yaitu nilai dari suku awal.

prosedure cetakDeret(**in** n : **integer**)

Masukan berupa satu bilangan integer positif yang lebih kecil dari 1000000.

Keluaran terdiri dari satu baris saja. Setiap suku dari deret tersebut dicetak dalam baris yang dan dipisahkan oleh sebuah spasi.

No	Masukan	Keluaran
1	22	22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Source code (soal)

```
//iqbalbawani
package main

import "fmt"

func cetakderetbilangan(n int) {
    fmt.Print(n)
    for n != 1 {
```

```

        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
        fmt.Print(" ", n)
    }
    fmt.Println()
}

func main() {
    var n int
    fmt.Print("Masukkan bilangan bulat positif: ")
    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakderetbilangan(n)
    } else {
        fmt.Println("Input harus bilangan bulat positif dan kurang dari
1000000.")
    }
}

```

Output =

```

PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani> go run "c:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani\ed1.go"
Masukkan bilangan bulat positif: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Users\PLN\Documents\tugas ittp s2\prakstrukkedua\2311102130_Iqbal-Bawani>

```

Penjelasan =

Program ini ditulis dalam bahasa Go dan berfungsi untuk mencetak deret bilangan berdasarkan algoritma Collatz. Fungsi utama dalam program ini adalah `cetakderetbilangan(n int)`, yang menerima satu parameter berupa bilangan bulat positif n . Di dalam fungsi, n dicetak terlebih dahulu, lalu program menjalankan proses iteratif berdasarkan aturan yang ditetapkan. Jika n genap, n akan dibagi 2, dan jika n ganjil, n dihitung dengan rumus $3n + 1$. Hasil n setelah setiap langkah akan dicetak hingga n mencapai 1, sehingga membentuk deret bilangan yang menunjukkan perubahan nilai n .

Dalam fungsi `main`, pengguna diminta untuk memasukkan bilangan bulat positif. Program akan memeriksa apakah nilai n berada dalam rentang yang valid, yaitu lebih besar dari 0 dan kurang dari 1.000.000. Jika nilai n valid, fungsi `cetakderetbilangan` akan dipanggil untuk menghasilkan deret bilangan. Namun, jika nilai n tidak memenuhi kriteria tersebut, program akan menampilkan pesan kesalahan, yang menyatakan bahwa input harus berupa bilangan bulat positif dan kurang dari 1.000.000. Program ini secara efektif mengilustrasikan penggunaan fungsi dan kontrol alur untuk menghasilkan hasil yang diinginkan.

DAFTAR PUSTAKA

Modul praktikum 3 & 4 fungtion dan procedure