

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2**

**MODUL 3&4
FUNGSI DAN PROSEDUR**



Oleh:

CINTA HERTANTIARA BINTANG

NIM

IF 11 07

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

1. DASAR TEORI

FUNGSI

A. Definisi Fungsi

Fungsi merupakan satu kesatuan rangkaian instruksi yang memberikan atau menghasilkan suatu nilai dan biasanya memetakan 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 variable
- Bagian dari ekspresi
- Bagian dari argumen suatu subprogram, dsb.

Karena itu selalu pilih nama fungsi yang menggambarkan nilai, seperti kata benda dan kata sifat.

B. Deklarasi Fungsi

	Notasi Algoritma
1	function <nama function> (<params>) -> <type>
2	kamus
3	{deklarasi variabel lokal dari fungsi}
4	...
5	algoritma
6	{badan algoritma fungsi}
7	...
8	return <value/variabel>
9	endfunction
	Notasi dalam bahasa Go
10	func <nama function> (<params>) <type> {
11	/* deklarasi variabel lokal dari fungsi */
12	...
13	/* badan algoritma fungsi*/
14	...
15	return <value/variabel>
16	
17	}

Pada bagian deklarasi terlihat setelah parameter, terdapat tipe data nilai yang dikembalikan, sedangkan pada bagian badan fungsi terdapat return dari nilai yang dikembalikan..

C. Cara Pemanggilan Fungsi

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

PROSEDUR

A. Definisi Procedure

Prosedur dapat dianggap sebagai potongan beberapa instruksi program menjadi suatu Instruksi baru yang dibuat untuk mengurangi kerumitan dari kode program yang kompleks pada suatu program yang besar. Prosedur akan menghasilkan suatu akibat atau efek langsung pada program ketika dipanggil pada program utama. Suatu subprogram dikatakan prosedur apabila:

1. Tidak ada deklarasi tipe nilai yang dikembalikan, dan
2. Tidak terdapat kata kunci return dalam badan subprogram.

Kedudukannya prosedur sama seperti instruksi dasar yang sudah ada sebelumnya (assignment) dan/atau instruksi yang berasal dari paket (fmt), seperti fmt.Scan dan fmt.Print. Karena itu selalu pilih nama prosedur yang berbentuk kata kerja atau sesuatu yang merepresentasikan proses sebagai nama dari prosedur. Contoh: cetak, hitungRerata, cariNilai, belok, mulal,.

B. Deklarasi Prosedur

Notasi Algoritma	
1	procedure <nama procedure> (<params>)
2	kamus
3	{deklarasi variabel lokal dari procedure}
4	...
5	algoritma
6	{badan algoritma procedure}
7	...
8	endprocedure
Notasi dalam bahasa Go	
9	func <nama procedure> (<params>) {
10	/* deklarasi variabel lokal dari procedure */
11	...
12	/* badan algoritma procedure */
13	...
14	}

Penulisan deklarasi ini berada di luar blok yang dari program utama atau **func main()** pada suatu program Go, dan bisa ditulis sebelum atau setelah dari blok program utama tersebut

C. Cara pemanggilan prosedur

Seperti yang sudah dijelaskan sebelumnya, suatu prosedur hanya akan dieksekusi apabila dipanggil baik secara langsung atau tidak langsung oleh program utama. Tidak langsung di sini maksudnya adalah prosedur dipanggil oleh program utama melalui perantara subprogram yang lain.

Pemanggilan suatu prosedur cukup mudah, yaitu dengan hanya menuliskan nama beserta parameter atau argumen yang diminta dari suatu prosedur.

D. Parameter

Suatu subprogram yang dipanggil dapat berkomunikasi dengan pemanggilnya melalui argumen yang diberikan melalui parameter yang dideklarasikan pada subprogramnya. Berikut ini jenis atau pembagian dari parameter.

Berdasarkan letak penulisannya pada program, maka parameter dapat dikelompokkan menjadi dua, yaitu parameter formal dan parameter aktual.

Parameter Formal

Parameter formal adalah parameter yang ditulis pada saat deklarasi suatu subprogram, parameter ini berfungsi sebagai petunjuk bahwa argumen apa saja yang diperlukan pada saat pemanggilan subprogram. Sebagai contoh parameter `Jarl jarl`, `tinggi` pada deklarasi fungsi `volumeTabung` adalah parameter formal (teks berwarna merah). Artinya ketika memanggil `volumeTabung` maka kita harus mempersiapkan dua integer (berapapun nilainya) sebagai jari jari dan tinggi.

Parameter Aktual

Sedangkan parameter aktual adalah argumen yang digunakan pada bagian parameter saat pemanggilan suatu subprogram. Banyaknya argumen dan tipe data yang terdapat pada parameter aktual harus mengikuti parameter formal.

Sebagai contoh argumen `r`, `t`, `15`, `14` dan `100` pada contoh kode di atas (teks berwarna biru) adalah parameter aktual, yang menyatakan nilai yang kita berikan sebagai jari-jari dan tinggi.

parameter juga dikelompokkan berdasarkan alokasi memorinya, yaitu `pass by value` dan `pass by reference`.

Pass by Value:

- Nilai parameter aktual disalin ke variabel lokal di subprogram.
- Parameter aktual dan formal memiliki alamat memori yang berbeda, sehingga perubahan di subprogram tidak mempengaruhi nilai parameter asli.
- Digunakan pada fungsi dan prosedur, tetapi hanya fungsi yang mengembalikan nilai ke pemanggil.
- Pada pseudocode, semua parameter fungsi adalah `pass by value`, dan pada prosedur digunakan kata kunci `in`. Di Go, tidak ada kata kunci khusus.

Pass by Reference:

- Parameter formal menyimpan alamat memori dari parameter aktual (pointer).
- Perubahan pada parameter formal akan memengaruhi parameter aktual.
- Biasanya digunakan pada prosedur untuk memungkinkan subprogram mengirimkan nilai kembali ke pemanggil.
- Pada pseudocode, digunakan kata kunci in/out, sedangkan pada Go, digunakan tanda asterisk (*) sebelum tipe data.

Fungsi sebaiknya menggunakan **pass by value** karena dapat mengembalikan nilai tanpa memengaruhi program utama. **Prosedur** lebih cocok menggunakan **pass by reference** agar bisa mengirimkan nilai ke pemanggil karena prosedur tidak mengembalikan nilai.

2. GUIDED

GUIDED MODUL 3-FUNGSI

Source Code

```
package main

import (
    "fmt"
)

func volumeTabung(jari_jari, tinggi int) float64 {
    var luasAlas, volume float64

    luasAlas = 3.14 * float64(jari_jari*jari_jari)
    volume = luasAlas * float64(tinggi)

    return volume
}

func main() {
    var r, t int

    var v1, v2 float64

    r = 5
    t = 10

    v1 = volumeTabung(r, t)

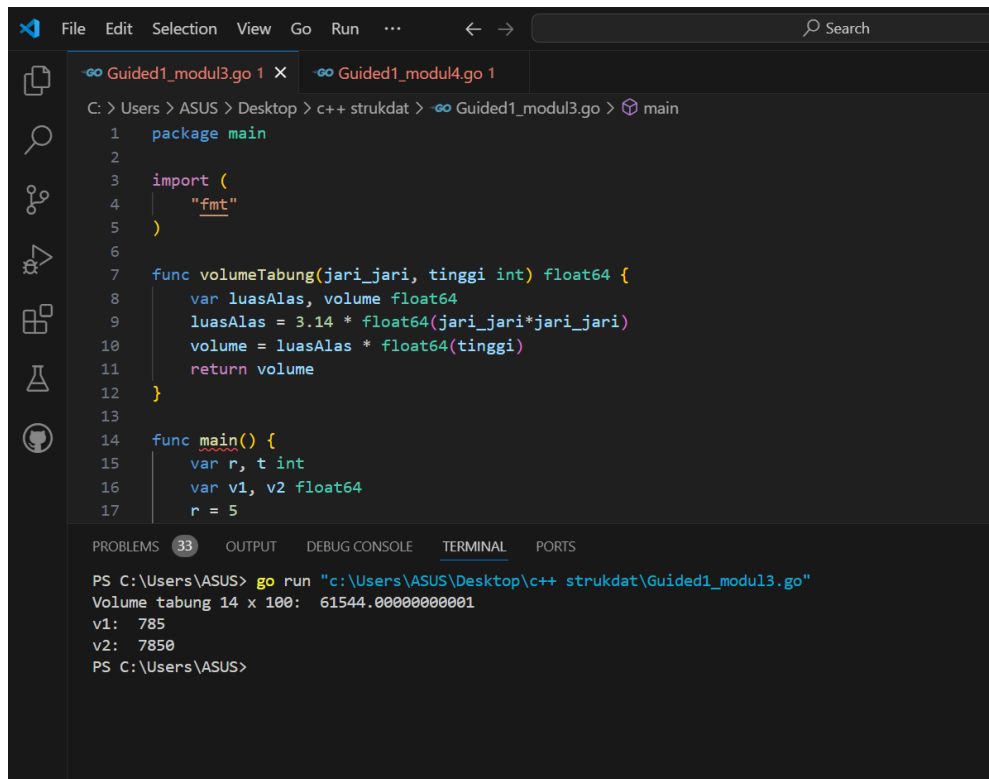
    v2 = volumeTabung(r, t) + volumeTabung(15, t)

    fmt.Println("Volume tabung 14 x 100: ", volumeTabung(14,
100))

    fmt.Println("v1: ", v1)

    fmt.Println("v2: ", v2)
}
```

Screenshoot Program



```
File Edit Selection View Go Run ... Search
Guided1_modul3.go 1 X Guided1_modul4.go 1
C: > Users > ASUS > Desktop > c++ strukdat > -e Guided1_modul3.go > main
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func volumeTabung(jari_jari, tinggi int) float64 {
8     var luasAlas, volume float64
9     luasAlas = 3.14 * float64(jari_jari*jari_jari)
10    volume = luasAlas * float64(tinggi)
11    return volume
12 }
13
14 func main() {
15     var r, t int
16     var v1, v2 float64
17     r = 5
18
19     v1 = volumeTabung(r, 10)
20     v2 = volumeTabung(r, 10) + volumeTabung(r, 10)
21     fmt.Println("Volume tabung 14 x 100: ", volumeTabung(14, 100))
22     fmt.Println("v1: ", v1)
23     fmt.Println("v2: ", v2)
24 }
PROBLEMS 33 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\Guided1_modul3.go"
Volume tabung 14 x 100: 61544.00000000001
v1: 785
v2: 7850
PS C:\Users\ASUS>
```

Deskripsi Program

Program ini menghitung volume tabung menggunakan fungsi `volumeTabung` yang menerima dua parameter: jari-jari dan tinggi, lalu mengembalikan hasil perhitungan volume berdasarkan rumus $\pi \times r^2 \times t$ dengan π mendekati 3.14. Dalam fungsi `main`, beberapa variabel diinisialisasi: jari-jari `r` dan tinggi `t` diatur menjadi 5 dan 10. Program kemudian menghitung volume tabung dengan nilai `r = 5` dan `t = 10` dan menyimpan hasilnya dalam variabel `v1`. Selain itu, volume tabung dengan jari-jari `r = 5` dan `r = 15` untuk tinggi yang sama dijumlahkan dan disimpan di variabel `v2`. Program juga menampilkan volume tabung dengan jari-jari 14 dan tinggi 100 serta akan mengoutputkan nilai `v1` dan `v2`.

GUIDED MODUL 4-PROSEDUR

Source Code

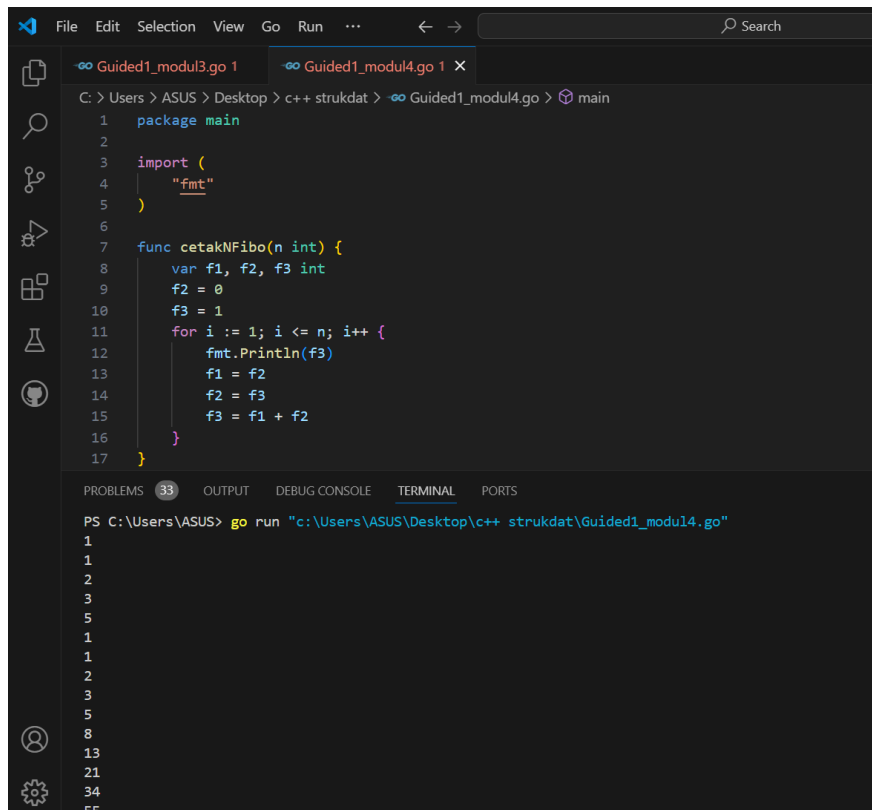
```
package main

import (
    "fmt"
)

func cetakNFibo(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
    cetakNFibo(x)
    cetakNFibo(100)
}
```


Screenshoot Program



```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func cetakNFibo(n int) {
8     var f1, f2, f3 int
9     f2 = 0
10    f3 = 1
11    for i := 1; i <= n; i++ {
12        fmt.Println(f3)
13        f1 = f2
14        f2 = f3
15        f3 = f1 + f2
16    }
17 }
```

```
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\Guided1_modul4.go"
1
1
2
3
5
1
1
2
3
5
8
13
21
34
55
```

Deskripsi Program

Program ini mencetak deret Fibonacci sebanyak n angka, dengan n ditentukan oleh input. Fungsi cetakNFibo menerima satu parameter n, yang menunjukkan jumlah angka Fibonacci yang ingin dicetak. Variabel f1, f2, dan f3 digunakan untuk menyimpan elemen Fibonacci saat dihitung, dimulai dengan f2 = 0 dan f3 = 1, yang mewakili dua angka pertama dalam deret. Dalam setiap iterasi loop, angka Fibonacci berikutnya dihitung sebagai penjumlahan dari dua angka sebelumnya, dan hasilnya dicetak. Dalam fungsi main, program pertama mencetak 5 angka Fibonacci menggunakan nilai x = 5, kemudian mencetak 100 angka Fibonacci dengan langsung memanggil fungsi cetakNFibo(100).

3. UNGUIDED

UNGUIDED MODUL 3-FUNGSI

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? **Masukan** terdiri dari empat buah bilangan asli a , b , c , dan d yang dipisahkan oleh spasi, di mana $a \geq c$ dan $b \geq d$.

Keluaran terdiri dari dua baris, di mana 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 di mana ($n \geq r$) 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	

Selesaikan program tersebut dengan memanfaatkan subprogram yang diberikan berikut ini !

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

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

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

Source Code

```
package main

import (
    "fmt"
)
```

```

func factorial(n int) int {
    if n == 0 {
        return 1
    }

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

    return result
}

func permutation(n, r int) int {
    return factorial(n) / factorial(n-r)
}

func combination(n, r int) int {
    return factorial(n) / (factorial(r) * factorial(n-r))
}

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

    fmt.Print("Masukkan 4 bilangan: ")

    fmt.Scan(&a, &b, &c, &d)

    p1 := permutation(a, c)
    c1 := combination(a, c)

    fmt.Printf("Hasil permutasi a terhadap c: %d\n", p1)
    fmt.Printf("Hasil kombinasi a terhadap c: %d\n", c1)
}

```

```

p2 := permutation(b, d)

c2 := combination(b, d)

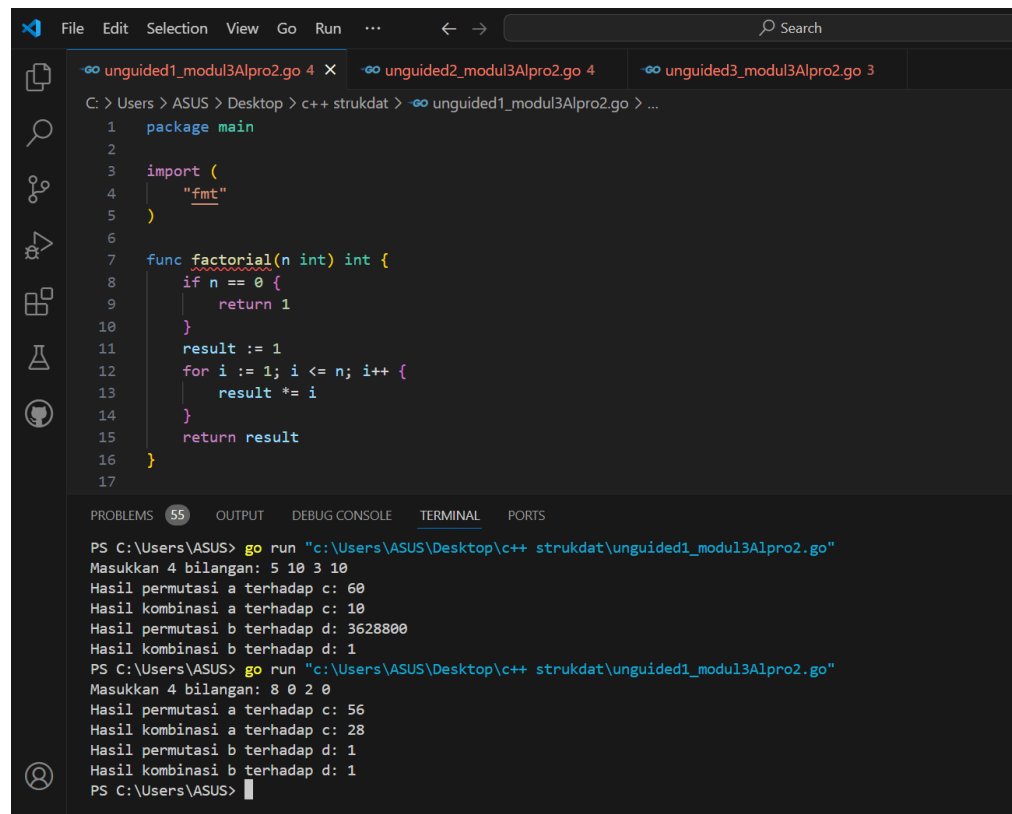
fmt.Printf("Hasil permutasi b terhadap d: %d\n", p2)

fmt.Printf("Hasil kombinasi b terhadap d: %d\n", c2)

}

```

Screenshoot Program



The screenshot shows a Go IDE with the following code in the editor:

```

1 package main
2
3 import (
4     "fmt"
5 )
6
7 func factorial(n int) int {
8     if n == 0 {
9         return 1
10    }
11    result := 1
12    for i := 1; i <= n; i++ {
13        result *= i
14    }
15    return result
16 }
17

```

The terminal output shows the program being run twice with different inputs:

```

PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided1_modul3Alpro2.go"
Masukkan 4 bilangan: 5 10 3 10
Hasil permutasi a terhadap c: 60
Hasil kombinasi a terhadap c: 10
Hasil permutasi b terhadap d: 3628800
Hasil kombinasi b terhadap d: 1
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided1_modul3Alpro2.go"
Masukkan 4 bilangan: 8 0 2 0
Hasil permutasi a terhadap c: 56
Hasil kombinasi a terhadap c: 28
Hasil permutasi b terhadap d: 1
Hasil kombinasi b terhadap d: 1
PS C:\Users\ASUS>

```

Deskripsi Program

Program ini menghitung nilai permutasi dan kombinasi dari dua pasang bilangan yang dimasukkan oleh pengguna. Fungsi factorial digunakan untuk menghitung faktorial dari suatu bilangan dengan metode iteratif. Fungsi permutation menghitung permutasi $P(n,r)$ menggunakan rumus $n!/(n-r)!$, sedangkan fungsi combination menghitung kombinasi $C(n,r)$ dengan rumus $n!/(r! \times (n-r)!)$. Dalam fungsi main, program meminta pengguna untuk memasukkan empat bilangan yang disimpan dalam variabel a, b, c, dan d. Program kemudian menghitung dan menampilkan hasil permutasi dan kombinasi dari bilangan a terhadap c dan b terhadap d.

2. Diberikan tiga buah fungsi matematika yaitu $f(x) = x^2$, $g(x) = x - 2$ dan $h(x) = x + 1$. Fungsi komposisi $(fogoh)(x)$ artinya adalah $f(g(h(x)))$. Tuliskan $f(x)$, $g(x)$ dan $h(x)$ dalam bentuk function.

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

Keluaran terdiri dari tiga baris, di mana baris pertama adalah $(fogoh)(a)$, baris kedua $(gohof)(b)$, dan baris ketiga adalah $(hofog)(c)$!

Contoh masukan dan keluaran

No	Masukan	Keluaran	Penjelasan
1	7 2 10	36 3 65	$(fogog)(7) = 36$ $(gohof)(2) = 3$ $(hofog)(10) = 65$
2	5 5 5	16 24 10	$(fogog)(5) = 16$ $(gohof)(5) = 24$ $(hofog)(5) = 10$
3	3 8 4	4 63 5	$(fogog)(5) = 4$ $(gohof)(5) = 63$ $(hofog)(5) = 5$

Source Code

```
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 main() {
```

```

var a, b, c int

fmt.Print("Masukkan 3 bilangan bulat a, b, dan c: ")

fmt.Scan(&a, &b, &c)

fogoh := f(g(h(a)))

gohof := g(h(f(b)))

hofog := h(f(g(c)))

fmt.Println("Hasil (fogoh) (a):", fogoh)

fmt.Println("Hasil (gohof) (b):", gohof)

fmt.Println("Hasuk (hofog) (c):", hofog)

}

```

Screenshoot Program

The screenshot shows a Go program in VS Code. The source code defines three functions: `f(x) = x * x`, `g(x) = x - 2`, and `h(x) = x + 1`. It then uses these functions to calculate `fogoh(a)`, `gohof(b)`, and `hofog(c)` based on user input. The terminal shows three runs of the program with different inputs and their corresponding outputs.

```

1 package main
2
3 import (
4     "fmt"
5 )
6
7 func f(x int) int {
8     return x * x
9 }
10
11 func g(x int) int {
12     return x - 2
13 }
14
15 func h(x int) int {
16     return x + 1
17 }

```

Terminal Output:

```

PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided2_modul3Alpro2.go"
Masukkan 3 bilangan bulat a, b, dan c: 5 5 5
Hasil (fogoh)(a): 16
Hasil (gohof)(b): 24
Hasuk (hofog)(c): 10
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided2_modul3Alpro2.go"
Masukkan 3 bilangan bulat a, b, dan c: 3 8 4
Hasil (fogoh)(a): 4
Hasil (gohof)(b): 63
Hasuk (hofog)(c): 5
PS C:\Users\ASUS>

```

Deskripsi Program

Program ini menghitung komposisi fungsi matematika menggunakan tiga fungsi: $f(x) = x * x$, $g(x) = x - 2$, dan $h(x) = x + 1$. Dalam fungsi `main`, program meminta pengguna untuk memasukkan tiga bilangan bulat `a`, `b`, dan `c`. Setelah menerima input, program menghitung tiga komposisi fungsi: `(fogoh)(a)`, yaitu komposisi dari fungsi $f(g(h(a)))$, kemudian `(gohof)(b)`, yang merupakan hasil dari $g(h(f(b)))$, dan terakhir `(hofog)(c), yaitu $h(f(g(c)))$. Hasil dari masing-masing komposisi fungsi tersebut ditampilkan

3. 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 diluar lingkaran 1 dan 2".

Contoh masukan dan keluaran

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}
```

Source Code

```
package main

import (
    "fmt"
    "math"
)

func jarak(a, b, c, d float64) float64 {
    return math.Sqrt((a-c)*(a-c) + (b-d)*(b-d))
}

func didalam(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 pusat lingkaran 1\n(cx1 cy1) dan radius r1:")
    fmt.Scan(&cx1, &cy1, &r1)

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

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

    dalamLingkaran1 := didalam(cx1, cy1, r1, x, y)
    dalamLingkaran2 := didalam(cx2, cy2, r2, x, y)

    if dalamLingkaran1 && dalamLingkaran2 {
        fmt.Println("Titik di dalam lingkaran 1 dan 2")
    } else if dalamLingkaran1 {
        fmt.Println("Titik di dalam lingkaran 1")
    } else if dalamLingkaran2 {
```



```

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

```

Screenshoot Program

```

C: > Users > ASUS > Desktop > c++ strukdat > -go unguided3_modul3A1pro2.go > ...
1 package main

PROBLEMS 55 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided3_modul3A1pro2.go"
Masukkan koordinat pusat lingkaran 1 (cx1 cy1) dan radius r1:
1 1 5
Masukkan koordinat pusat lingkaran 2 (cx2 cy2) dan radius r2:
8 8 4
Masukkan koordinat titik sembarang (x y):
2 2
Titik di dalam lingkaran 1
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\tempCodeRunnerFile.go"
Masukkan koordinat pusat lingkaran 1 (cx1 cy1) dan radius r1:
1 2 3
Masukkan koordinat pusat lingkaran 2 (cx2 cy2) dan radius r2:
4 5 6
Masukkan koordinat titik sembarang (x y):
7 8
Titik di dalam lingkaran 2
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\tempCodeRunnerFile.go"
5 10 15
Masukkan koordinat pusat lingkaran 2 (cx2 cy2) dan radius r2:
-15 4 20
Masukkan koordinat titik sembarang (x y):
0 0
Titik di dalam lingkaran 1 dan 2
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\tempCodeRunnerFile.go"
Masukkan koordinat pusat lingkaran 1 (cx1 cy1) dan radius r1:
1 1 5
Masukkan koordinat pusat lingkaran 2 (cx2 cy2) dan radius r2:
8 8 4
Masukkan koordinat titik sembarang (x y):
15 20
Titik di luar lingkaran 1 dan 2
PS C:\Users\ASUS>

```

Deskripsi Program

Program ini menentukan apakah sebuah titik berada di dalam atau di luar dua lingkaran yang berbeda. Pertama, fungsi `jarak` menghitung jarak antara dua titik menggunakan rumus jarak Euclidean. Fungsi `didalam` mengecek apakah suatu titik berada di dalam lingkaran dengan membandingkan jarak antara pusat lingkaran dan titik tersebut dengan radius lingkaran. Di dalam fungsi `main`, program meminta pengguna untuk memasukkan koordinat pusat dan radius untuk dua lingkaran, serta koordinat titik sembarang. Program kemudian memeriksa apakah titik tersebut berada di dalam salah satu atau kedua lingkaran menggunakan

fungsi `didalam`. Berdasarkan hasilnya, program menampilkan salah satu dari empat kemungkinan: titik berada di dalam lingkaran 1 dan 2, di dalam lingkaran 1 saja, di dalam lingkaran 2 saja, atau di luar kedua lingkaran

UNGUIDED MODUL 4-PROSEDUR

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? **Masukan** terdiri dari empat buah bilangan asli a , b , c , dan d yang dipisahkan oleh spasi, di mana $a \geq c$ dan $b \geq d$.

Keluaran terdiri dari dua baris, di mana 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 di mana ($n \geq r$) 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	

Selesaikan program tersebut dengan memanfaatkan subprogram 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

```
package main

import (
    "fmt"
)

func factorial(n int) int {
    if n == 0 {
        return 1
    }

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

    return result
}

func permutation(n, r int) int {
    return factorial(n) / factorial(n-r)
}

func combination(n, r int) int {
    return factorial(n) / (factorial(r) * factorial(n-r))
}

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

    fmt.Print("Masukkan 4 bilangan: ")
}
```

```

fmt.Scan(&a, &b, &c, &d)

p1 := permutation(a, c)

c1 := combination(a, c)

fmt.Printf("Hasil permutasi a terhadap c: %d\n", p1)
fmt.Printf("Hasil kombinasi a terhadap c: %d\n", c1)

p2 := permutation(b, d)

c2 := combination(b, d)

fmt.Printf("Hasil permutasi b terhadap d: %d\n", p2)
fmt.Printf("Hasil kombinasi b terhadap d: %d\n", c2)

}

```

Screenshoot Program

The screenshot shows a Go program in VS Code. The code defines a `factorial` function and uses it to calculate permutations and combinations for two sets of input numbers. The terminal output shows the results for two different input sets: (5, 10, 3, 10) and (8, 0, 2, 0).

```

1  package main
2
3  import (
4      "fmt"
5  )
6
7  func factorial(n int) int {
8      if n == 0 {
9          return 1
10     }
11     result := 1
12     for i := 1; i <= n; i++ {
13         result *= i
14     }
15     return result
16 }
17

```

Terminal Output:

```

PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided1_modul3Alpro2.go"
Masukkan 4 bilangan: 5 10 3 10
Hasil permutasi a terhadap c: 60
Hasil kombinasi a terhadap c: 10
Hasil permutasi b terhadap d: 3628800
Hasil kombinasi b terhadap d: 1
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided1_modul3Alpro2.go"
Masukkan 4 bilangan: 8 0 2 0
Hasil permutasi a terhadap c: 56
Hasil kombinasi a terhadap c: 28
Hasil permutasi b terhadap d: 1
Hasil kombinasi b terhadap d: 1
PS C:\Users\ASUS>

```

Deskripsi Program

Program ini menghitung nilai permutasi dan kombinasi dari dua pasang bilangan yang dimasukkan oleh pengguna. Fungsi factorial digunakan untuk menghitung faktorial dari suatu bilangan dengan metode iteratif. Fungsi permutation menghitung permutasi $P(n,r)$ menggunakan rumus $n!/(n-r)!$, sedangkan fungsi combination menghitung kombinasi $C(n,r)$ dengan rumus $n!/(r! \times (n-r)!)$. Dalam fungsi main, program meminta pengguna untuk memasukkan empat bilangan yang disimpan dalam variabel a, b, c, dan d. Program kemudian menghitung dan menampilkan hasil permutasi dan kombinasi dari bilangan a terhadap c dan b terhadap d

2. 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.

prosedure `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

```
package main

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

func hitungSkor(input string) (soal int, skor int) {
    parts := strings.Fields(input)
    soal = 0
    skor = 0

    for i := 1; i < len(parts); i++ {
        var waktu int
        fmt.Sscan(parts[i], &waktu)

        if waktu <= 300 {
            soal++
            skor += waktu
        }
    }

    return soal, skor
}

func main() {
    scanner := bufio.NewScanner(os.Stdin)

    var inputData []string
```

```
        fmt.Println("Masukkan data peserta. Akhiri dengan  
mengetik 'Selesai':")

    for {

        scanner.Scan()

        line := scanner.Text()

        if strings.ToLower(line) == "selesai" {

            break

        }

        inputData = append(inputData, line)

    }

    var pemenang string

    var maxSoal, minSkor int

    maxSoal = 0

    minSkor = 1000000

    for _, data := range inputData {

        soal, skor := hitungSkor(data)

        nama := strings.Fields(data)[0]

        if soal > maxSoal || (soal == maxSoal && skor < minSkor) {

            pemenang = nama

            maxSoal = soal

            minSkor = skor

        }

    }

    return pemenang
```

```

    }

    }

    fmt.Println("\nHasil:")

    fmt.Printf("%s %d %d\n", pemenang, maxSoal, minSkor)

}

```

Screenshoot Program

```

1 package main
2
3 import (
4     "bufio"
5     "fmt"
6     "os"
7     "strings"
8 )
9
10 func hitungSkor(input string) (soal int, skor int) {
11     parts := strings.Fields(input)
12     soal = 0
13     skor = 0
14
15     for i := 1; i < len(parts); i++ {
16         var waktu int
17         fmt.Sscan(parts[i], &waktu)
18     }
19 }
20
21 func main() {
22     reader := bufio.NewReader(os.Stdin)
23     fmt.Println("Masukkan data peserta. Akhiri dengan mengetik 'Selesai':")
24     input := ""
25     for {
26         input, _ = reader.ReadString('\n')
27         if input == "Selesai" {
28             break
29         }
30         soal, skor = hitungSkor(input)
31         fmt.Println("Astuti 20 50 301 301 61 71 75 10")
32         fmt.Println("Bertha 25 47 301 26 50 60 65 21")
33     }
34     fmt.Println("Selesai")
35
36     pemenang := "Bertha"
37     maxSoal := 301
38     minSkor := 25
39
40     fmt.Println("\nHasil:")
41     fmt.Printf("%s %d %d\n", pemenang, maxSoal, minSkor)
42 }

```

```

PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided2_modul4.go"
Masukkan data peserta. Akhiri dengan mengetik 'Selesai':
Astuti 20 50 301 301 61 71 75 10
Bertha 25 47 301 26 50 60 65 21
Selesai

Hasil:
Bertha 7 294
PS C:\Users\ASUS>

```

Deskripsi Program

Program ini digunakan untuk menghitung skor dan menentukan pemenang dalam sebuah kompetisi pemrograman berdasarkan input data peserta. Fungsi `hitungSkor` menerima string input yang berisi nama peserta diikuti oleh waktu penyelesaian setiap soal. Fungsi ini menghitung jumlah soal yang berhasil diselesaikan dalam waktu 300 menit atau kurang, serta total waktu yang digunakan untuk menyelesaikan soal-soal tersebut. Dalam fungsi `main`, program meminta pengguna untuk memasukkan data peserta hingga pengguna mengetik "Selesai". Data yang dimasukkan disimpan dalam slice `inputData`, yang kemudian dianalisis untuk menemukan

pemenang berdasarkan jumlah soal yang diselesaikan dan waktu terpendek. Jika ada peserta yang menyelesaikan jumlah soal yang sama, pemenang ditentukan oleh skor terendah. Setelah pemrosesan, program mencetak nama pemenang, jumlah soal yang diselesaikan, dan total waktu yang dihabiskan.

3. 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 $1/2n$, 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 **sklena** 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 cetak Deret (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

```
package main

import (
    "fmt"
)

func cetakDeret(n int) {
    for n != 1 {
        fmt.Printf("%d ", n)
```

```
        if n%2 == 0 {
            n = n / 2
        } else {
            n = 3*n + 1
        }
    }

    fmt.Println(n)
}

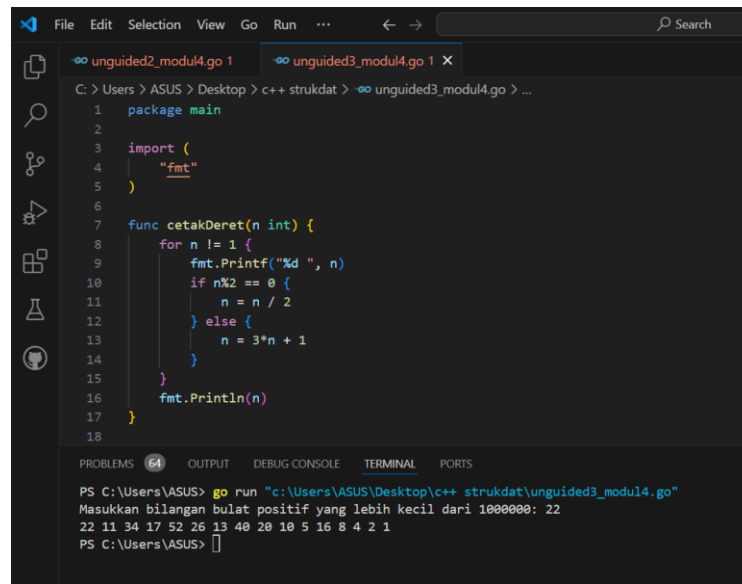
func main() {
    var n int

    fmt.Print("Masukkan bilangan bulat positif yang lebih
kecil dari 1000000: ")

    fmt.Scan(&n)

    if n > 0 && n < 1000000 {
        cetakDeret(n)
    } else {
        fmt.Println("Bilangan harus positif dan kurang dari
1000000")
    }
}
```

Screenshoot Program



```
1 package main
2
3 import (
4     "fmt"
5 )
6
7 func cetakDeret(n int) {
8     for n != 1 {
9         fmt.Printf("%d ", n)
10        if n%2 == 0 {
11            n = n / 2
12        } else {
13            n = 3*n + 1
14        }
15    }
16    fmt.Println(n)
17 }
18
```

PROBLEMS 64 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\ASUS> go run "c:\Users\ASUS\Desktop\c++ strukdat\unguided3_modul4.go"
Masukkan bilangan bulat positif yang lebih kecil dari 1000000: 22
22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
PS C:\Users\ASUS>
```

Deskripsi Program

Program ini mencetak deret angka berdasarkan algoritma Collatz untuk bilangan bulat positif yang dimasukkan oleh pengguna, di mana bilangan tersebut harus lebih kecil dari 1.000.000. Fungsi `cetakDeret` mengimplementasikan logika algoritma Collatz, yang menyatakan bahwa jika bilangan genap, maka dibagi dua; jika bilangan ganjil, maka dikalikan tiga dan ditambahkan satu. Proses ini diulang hingga mencapai angka 1. Dalam fungsi `main`, program meminta pengguna untuk memasukkan bilangan bulat positif, dan jika bilangan tersebut valid (positif dan kurang dari 1000000), maka fungsi `cetakDeret` dipanggil untuk mencetak deret tersebut. Jika bilangan tidak valid, program akan menampilkan pesan kesalahan yang menyatakan bahwa bilangan harus positif dan kurang dari 1000000.