

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV
PROSEDUR**



Disusun Oleh :

Haposan Felix Marcel Siregar / 2311102210

IF_11_06

Dosen Pengampu :

Abednego Dwi Septiadi, S.Kom., M.Kom

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

1. Pengertian Prosedur dalam Pemrograman

Prosedur adalah sekumpulan instruksi yang dikelompokkan dalam satu unit, yang bisa dipanggil atau dimanfaatkan berulang kali dalam program. Prosedur membantu dalam mengorganisasi kode, meningkatkan keterbacaan, serta mempermudah pemeliharaan program. Dalam konteks Golang (atau Go), prosedur sering disebut sebagai "fungsi" (function).

2. Fungsi dalam Golang

Golang menawarkan cara yang sederhana dan efisien untuk mendefinisikan fungsi. Fungsi dalam Golang dapat memiliki parameter, mengembalikan nilai, dan dapat dipanggil dari bagian lain dari program. Struktur dasar dari fungsi di Golang adalah sebagai berikut:

```
func namaFungsi(parameter1 tipeData1, parameter2 tipeData2)
tipeDataKeluaran {
    // kode yang dijalankan
    return nilai
}
```

3. Keunggulan Menggunakan Prosedur

Menggunakan prosedur dalam pemrograman Golang punya banyak keuntungannya, antara lain:

- Reusabilitas: Kode yang udah ditulis bisa dipakai berkali-kali tanpa perlu nulis ulang.
- Modularitas: Memudahkan dalam membagi kode jadi bagian-bagian kecil yang bisa ditangani sendiri.

- Keterbacaan: Membuat kode lebih mudah dipahami oleh orang lain (atau diri sendiri) ketika dilihat di masa mendatang.

4. Contoh Implementasi Prosedur

Misalnya, lo mau buat sebuah fungsi yang nambahin dua angka. Berikut contoh implementasinya:

```
package main

import "fmt"

// mendeklarasikan fungsi nambah
func nambah(a int, b int) int {
    return a + b
}

func main() {
    hasil := nambah(5, 3)
    fmt.Println("Hasil penjumlahan:", hasil)
}
```

Dalam contoh di atas, kita mendefinisikan fungsi `nambah` yang menerima dua parameter bertipe `int` dan mengembalikan hasil penjumlahan dari kedua angka tersebut.

5. Kesimpulan

Prosedur (fungsi) dalam pemrograman Golang adalah elemen kunci yang memungkinkan pengembangan perangkat lunak yang terstruktur dan mudah dikelola. Penerapan prosedur membantu pengembang dalam mengatur logika program dan memudahkan debugging serta pemeliharaan di masa depan.

II. GUIDED

1. Soal Studi Case

Membuat program sederhana yang dapat mencetak pesan dengan tipe tertentu berdasarkan nilai input. Terdapat tiga jenis pesan yang dapat ditampilkan: **error**, **warning**, atau **informasi**, bergantung pada nilai bilangan yang diberikan. Selain itu, program juga diminta untuk mencantumkan watermark dengan informasi tentang pembuat program, seperti nama dan NIM mahasiswa, di bagian atas hasil keluaran.

Sourcecode

```
package main

import (
    fmt
)

// Fungsi untuk mencetak watermark
func watermark() {
    fmt.Println("=====")
    fmt.Println("Nama: Haposan Siregar")
    fmt.Println("NIM: 2311102210")
    fmt.Println("=====")
}

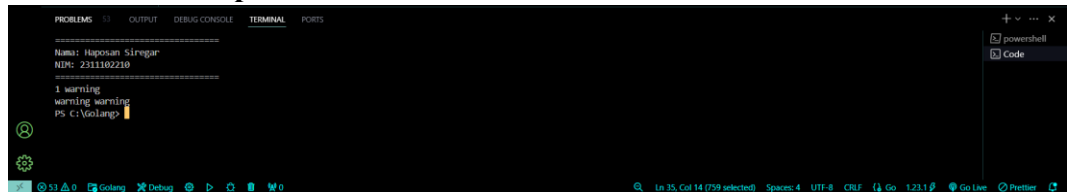
func main() {
    watermark() // Menampilkan watermark

    var bilangan int
    var pesan string
    fmt.Scan (&bilangan, &pesan)
    cetakPesan(pesan, bilangan)
}

func cetakPesan (M string, flag int){
    var jenis string = " "
    if flag == 0 {
        jenis = "error"
    } else if flag == 1 {
        jenis = "warning"
    } else if flag == 2 {
        jenis = "informasi"
    }
}
```

```
fmt.Println(M, jenis)
}
```

Screenshoot Output



Deskripsi Program

Program ini bertujuan untuk mencetak pesan berdasarkan input dari pengguna. Pengguna diminta untuk memberikan dua masukan: sebuah bilangan dan sebuah pesan. Bilangan ini akan digunakan untuk menentukan jenis pesan yang akan dicetak (error, warning, atau informasi). Selain itu, program menampilkan watermark yang berisi informasi pembuat di awal keluaran.

Alur Program:

- Program akan mencetak watermark berisi nama dan NIM dari pembuat.
- Pengguna diminta untuk memasukkan sebuah bilangan (flag) dan pesan (M).
- Program akan memanggil fungsi cetakPesan yang akan mencetak pesan yang diberikan dengan jenis pesan yang ditentukan oleh nilai bilangan (flag).

2. Soal Studi Case

Sebuah perusahaan memiliki sistem pendaftaran pengguna baru, di mana setiap kali pengguna baru terdaftar, sistem harus mengirimkan email konfirmasi secara otomatis. Perusahaan ingin memastikan bahwa setiap pengguna yang baru terdaftar menerima pesan konfirmasi melalui email. Untuk memenuhi kebutuhan ini, seorang developer diminta membuat program sederhana yang secara otomatis mengirimkan notifikasi email ke daftar pengguna baru setelah pendaftaran berhasil.

Sourcecode

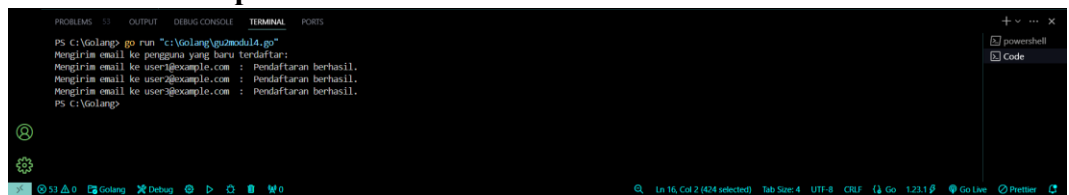
```
package main

import "fmt"
```

```
//Procedure untuk mengirim email
func sendEmailNotification (email string){
    fmt.Printf("Mengirim email ke %s : Pendaftaran
    berhasil.\n", email)
}
func main(){
    emails := []string {"user1@example.com", "user2@example.com",
    "user3@example.com"}

    fmt.Println("Mengirim email ke pengguna yang baru terdaftar: ")
    for _, email := range emails {
        sendEmailNotification(email)
    }
}
```

Screenshoot Output



```
PS c:\Golang> go run "c:\Golang\gizmodul4.go"
Mengirim email ke pengguna yang baru terdaftar:
Mengirim email ke user1@example.com : Pendaftaran berhasil.
Mengirim email ke user2@example.com : Pendaftaran berhasil.
Mengirim email ke user3@example.com : Pendaftaran berhasil.
PS c:\Golang>
```

Deskripsi Program

Program ini dirancang untuk mensimulasikan pengiriman notifikasi email kepada sekelompok pengguna yang baru saja terdaftar. Program menggunakan sebuah **prosedur** untuk mencetak pesan yang mengindikasikan bahwa email telah dikirim ke pengguna tertentu, dengan menggunakan alamat email yang diberikan.

Alur Program:

1. Program memiliki daftar email dari pengguna yang baru terdaftar.
2. Untuk setiap email dalam daftar, program memanggil fungsi `sendEmailNotification`, yang mencetak pesan bahwa email berhasil dikirim ke pengguna tersebut.
3. Program mencetak pesan untuk setiap pengguna baru yang terdaftar, menandakan bahwa notifikasi email telah dikirimkan.

3. Soal Studi Case

Seorang mahasiswa diminta untuk membuat program yang mengimplementasikan dua fungsi matematika sederhana yang menerima dua bilangan bulat sebagai input dan mengembalikan hasil perhitungan. Fungsi pertama (f1) mengembalikan hasil secara langsung, sedangkan fungsi kedua (f2) memodifikasi hasil menggunakan pointer. Program juga harus menampilkan watermark yang berisi informasi identitas pembuat program.

Sourcecode

```
package main

import "fmt"
func f1(x, y int) float64 {
    var hasil float64
    hasil = float64(2*x) - 0.5*float64(y) + 3.0
    return hasil
}
func f2 (x, y int, hasil *float64) {
    *hasil = float64(2*x) - 0.5*float64 (y) + 3.0
}

// Fungsi untuk mencetak watermark
func watermark() {
    fmt.Println("=====")
    fmt.Println("Nama: Haposan Siregar")
    fmt.Println("NIM: 2311102210")
    fmt.Println("=====")
}

func main() {
    watermark() // Menampilkan watermark
    var a, b int
    var c float64

    fmt.Print ("Enter two integers: ")
    fmt.Scan(&a, &b)

    f2 (a, b, &c)

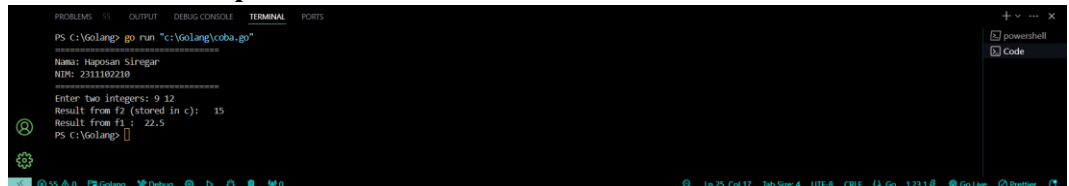
    fmt.Println("Result from f2 (stored in c): ", c)

    resultF1 := f1(b, a)
```

```
fmt.Println ("Result from f1 : ", resultF1)
```

```
}
```

Screenshoot Output

A screenshot of a terminal window showing the execution of a Go program. The prompt is 'PS C:\golang> go run "c:\golang\coba.go"'. The program prints a separator line, then asks for two integers. The user enters '9' and '12'. The program then prints 'Result from f2 (stored in c): 15' and 'Result from f1 : 22.5'. The terminal window has a dark theme and shows standard Go IDE interface elements like 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL' tabs.

```
PS C:\golang> go run "c:\golang\coba.go"
=====
Name: Hapsan Siregar
NIM: 231180219
=====
Enter two integers: 9 12
Result from f2 (stored in c): 15
Result from f1 : 22.5
PS C:\golang>
```

Deskripsi Program

Alur Program:

- Program dimulai dengan mencetak watermark menggunakan fungsi watermark.
- Pengguna diminta untuk memasukkan dua bilangan bulat (a dan b).
- Program memanggil fungsi **f2** untuk menghitung hasil operasi matematika dan menyimpannya dalam variabel c (menggunakan pointer).
- Program mencetak hasil dari fungsi f2 yang disimpan di c.
- Program juga memanggil fungsi **f1** untuk melakukan perhitungan yang sama, tetapi hasilnya langsung dikembalikan dan disimpan dalam variabel resultF1.
- Program mencetak hasil dari fungsi f1.

III. UNGUIDED

1. Soal Studi Case

Diminta untuk membuat program yang menghitung **permutasi** dan **kombinasi** dari dua pasangan nilai yang diberikan oleh pengguna. Nilai-nilai yang dimasukkan akan diproses berdasarkan rumus **$P(n, r)$** untuk permutasi dan **$C(n, r)$** untuk kombinasi, dengan syarat bahwa input $a \geq c$ dan $b \geq d$

Sourcecode

```
package main

import (
    "fmt"
)

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

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

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

// Fungsi untuk mencetak watermark
func watermark() {
    fmt.Println("=====")
    fmt.Println("Nama: Haposan Siregar")
    fmt.Println("NIM: 2311102210")
}
```

```

    fmt.Println("=====")
}

func main() {
    watermark() // Menampilkan watermark

    for {
        // Input dari user
        var a, b, c, d int
        fmt.Println("Masukkan nilai a, b, c, d (a >= c dan b >= d):")
        fmt.Scan(&a, &b, &c, &d)

        // Cek syarat a >= c dan b >= d
        if a < c || b < d {
            fmt.Println("Ada kesalahan, sampai jumpa kembali!")
            break
        }

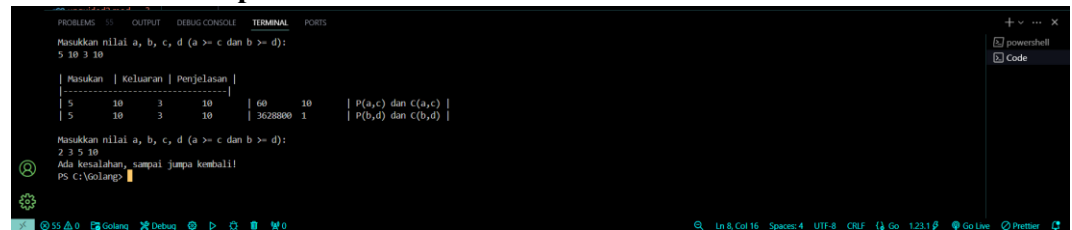
        // Hasil permutasi dan kombinasi untuk (a, c)
        permutationAC := permutation(a, c)
        combinationAC := combination(a, c)

        // Hasil permutasi dan kombinasi untuk (b, d)
        permutationBD := permutation(b, d)
        combinationBD := combination(b, d)

        // Menampilkan hasil dalam bentuk tabel
        fmt.Printf("\n| %-8s | %-8s | %-8s | \n", "Masukan", "Keluaran",
            "Penjelasan")
        fmt.Println("|-----|")
        fmt.Printf("| %-8d %-8d %-8d %-8d | %-8d %-8d | %-8s | \n",
            a, b, c, d, permutationAC, combinationAC, "P(a,c) dan C(a,c)")
        fmt.Printf("| %-8d %-8d %-8d %-8d | %-8d %-8d | %-8s | \n",
            a, b, c, d, permutationBD, combinationBD, "P(b,d) dan C(b,d)")
        fmt.Println()
    }
}

```

Screenshoot Output



```
PROBLEMS 55 OUTPUT DEBUG CONSOLE TERMINAL PORTS
Masukkan nilai a, b, c, d (a >= c dan b >= d):
5 10 3 10

| Masukan | Keluaran | Penjelasan | | | | |
|---|---|---|---|---|---|---|
| 5      | 10      | 3      | 10      | 60      | 10      | P(a,c) dan c(a,c) |
| 5      | 10      | 3      | 10      | 3628800 | 1       | P(b,d) dan c(b,d) |

Masukkan nilai a, b, c, d (a >= c dan b >= d):
2 3 5 10
Ada kesalahan, sampai jumpa kembali!
PS C:\Golang>
```

Deskripsi Program

Alur Program:

- Pengguna diminta untuk memasukkan 4 bilangan bulat positif (a, b, c, d).
 - Syarat:** $a \geq c$ dan $b \geq d$.
- Program menghitung:
 - Permutasi dan kombinasi untuk (a, c).**
 - Permutasi dan kombinasi untuk (b, d).**
- Hasil perhitungan ditampilkan dalam bentuk tabel dengan kolom "Masukan", "Keluaran", dan "Penjelasan".
- Jika input tidak memenuhi syarat, program memberikan pesan kesalahan dan keluar.

2. Soal Studi Case

Program ini dirancang untuk mengelola dan menghitung skor peserta dalam suatu kompetisi pemrograman, serta menghitung permutasi dan kombinasi berdasarkan input pengguna. Peserta akan diminta untuk mengisi waktu yang mereka habiskan untuk menyelesaikan soal-soal dalam kompetisi, dan program akan menentukan pemenang berdasarkan skor tertinggi yang dihasilkan. Selain itu, program ini juga menyediakan fitur untuk menghitung dan menampilkan permutasi dan kombinasi dari dua pasangan angka yang diinput oleh pengguna.

Source code

```
package main

import (
    "fmt"
    "strings"
)

// Prosedur menghitung skor untuk setiap peserta
func hitungSkor(input string) (int, int) {
```

```

// Pisahkan input dengan spasi untuk mengekstrak nama dan waktu
peserta untuk masalah yang diselesaikan
fields := strings.Fields(input)
totalSolved := 0
totalTime := 0

//Proses bidang yang tersisa, yang mewakili waktu untuk setiap
masalah
for i := 1; i < len(fields); i++ {
    var time int
    fmt.Sscan(fields[i], &time) // Convert the string time into an
integer
    if time <= 300 {           // Only count problems solved within 300
minutes
        totalSolved++
        totalTime += time
    }
}

//Kembalikan jumlah total masalah yang diselesaikan dan total waktu
untuk masalah yang diselesaikan
return totalSolved, totalTime
}

// Fungsi untuk mencetak watermark
func watermark() {
    fmt.Println("=====")
    fmt.Println("Nama: Haposan Siregar")
    fmt.Println("NIM: 2311102210")
    fmt.Println("=====")
}

func main() {
    watermark() // Menampilkan watermark

//Input data untuk peserta
inputs := []string{
    "Astuti 20 50 301 301 61 71 75 10",
    "Bertha 25 47 301 26 50 60 65 21",
}

var winnerName string = ""
var winnerSolved int = 0

```

```

var winnerTime int = 0

Header keluaran untuk tabel
fmt.Printf("%-10s | %-10s | %-10s\n", "Nama", "Soal Selesai",
"Total Waktu")
fmt.Println("-----")

//Memproses data setiap peserta
for _, input := range inputs {
    solved, time := hitungSkor(input)

    //Mengekstrak nama peserta
    fields := strings.Fields(input)
    name := fields[0]

    // Mencetak hasil setiap peserta dalam bentuk tabel
    fmt.Printf("%-10s | %-10d | %-10d\n", name, solved, time)

    // Tentukan apakah peserta saat ini harus menjadi pemenang baru
    if solved > winnerSolved || (solved == winnerSolved && time <
winnerTime) {
        winnerName = name
        winnerSolved = solved
        winnerTime = time
    }
}

// Keluarkan detail pemenang
fmt.Println("-----")
fmt.Printf("Pemenang: %s, Jumlah Soal: %d, Total Waktu: %d
menit\n", winnerName, winnerSolved, winnerTime)
}

```

Screenshoot Output

```

PS C:\Golang> go run "c:\Golang\tempcodeRunnerFile.go"
Nama: Hapusan Siregar
NIK: 211102210
-----
Nama | Soal Selesai | Total Waktu
-----
Astuti | 6 | 287
Bertha | 7 | 294
-----
Pemenang: Bertha, Jumlah Soal: 7, Total Waktu: 294 menit
PS C:\Golang>

```

Deskripsi Program

a) Inisialisasi Program:

- Program dimulai dengan menampilkan informasi pengembang (nama dan NIM).
- b) Input Data Peserta:
- Data peserta didefinisikan dalam bentuk array string, berisi nama dan waktu penyelesaian untuk setiap masalah.
- c) Menampilkan Header Tabel:
- Program mencetak header tabel untuk hasil penilaian.
- d) Proses Penilaian:
- Untuk setiap peserta:
 - i. Program menghitung jumlah masalah yang diselesaikan dan total waktu menggunakan fungsi `hitungSkor()`.
 - ii. Hasil dicetak dalam format tabel.
- e) Menentukan Pemenang:
- Pemenang ditentukan berdasarkan jumlah masalah yang diselesaikan dan total waktu (semakin banyak masalah dan semakin sedikit waktu).
- f) Akhir Program:
- Program mencetak informasi pemenang dan selesai.

3. Soal Studi Case

Program ini merupakan implementasi sederhana dari **Collatz Conjecture** atau yang sering disebut dengan **$3n + 1$ problem**. Program meminta input berupa angka positif dan mencetak deret berdasarkan aturan tertentu. Selain itu, program juga mencetak watermark yang berisi informasi nama dan nomor mahasiswa.

Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mencetak urutan berdasarkan angka awal yang
// diberikan dalam format tabel
func cetakDeret(n int) {
    counter := 0
    fmt.Printf("%-10s", "No")
    fmt.Printf("%-10s\n", "Suku")

    // Putar sampai n menjadi 1
```

```

for n != 1 {
    counter++
    // Cetak penghitung dan angka dalam kolom
    fmt.Printf("%0-10d", counter)
    fmt.Printf("%0-10d\n", n)
    if n%2 == 0 {
        n = n / 2 // If even, divide by 2
    } else {
        n = 3*n + 1 // If odd, calculate 3n + 1
    }
}
counter++
// Cetak angka terakhir (yang akan menjadi 1)
fmt.Printf("%0-10d", counter)
fmt.Printf("%0-10d\n", n)
}

// Fungsi untuk mencetak watermark
func watermark() {
    fmt.Println("=====")
    fmt.Println("Nama: Haposan Siregar")
    fmt.Println("NIM: 2311102210")
    fmt.Println("=====")
}

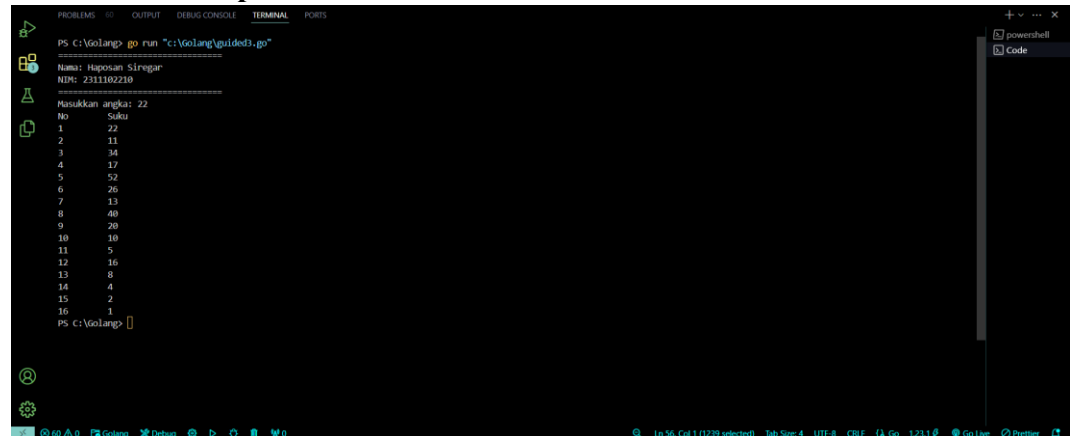
func main() {
    watermark() // Menampilkan watermark

    // Contoh input
    var n int
    fmt.Print("Masukkan angka: ")
    fmt.Scanln(&n)

    // Pastikan input valid
    if n > 0 && n < 1000000 {
        cetakDeret(n) // Call the function to print the sequence in table
        format
    } else {
        fmt.Println("Masukan harus bilangan positif dan kurang dari
1000000.")
    }
}

```

Screenshoot Output



```
PS C:\Golang> go run "c:\Golang\guides3.go"
=====
Nama: Haposan Siregar
NIM: 2311102210
=====
Masukkan angka: 22
=====
No    Suku
1     22
2     11
3     34
4     17
5     52
6     26
7     13
8     40
9     20
10    10
11    5
12    16
13    8
14    4
15    2
16    1
PS C:\Golang>
```

Deskripsi Program

- Input:** Pengguna diminta untuk memasukkan sebuah angka positif kurang dari 1 juta.
- Validasi Input:** Program akan memeriksa apakah input yang dimasukkan valid (bilangan positif dan kurang dari 1 juta). Jika tidak valid, pesan error akan ditampilkan.
- Proses Deret Collatz:** Jika input valid, program akan mencetak deret Collatz (jika angka genap, dibagi 2; jika ganjil, dihitung $3n + 1$) dalam bentuk tabel dengan dua kolom (No dan Suku) hingga mencapai angka 1.
- Output:** Hasil dari perhitungan deret tersebut ditampilkan dalam bentuk tabel di mana kolom pertama berisi nomor urut dan kolom kedua berisi angka hasil perhitungan.