LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 8

PENCARIAN NILAI ACAK PADA HIMPUNAN DATA

DISUSUN OLEH:

Keishin Naufa Alfaridzhi

103112400061

S1 IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

A. Bahasa Yang Digunakan

Pada praktikum ini bahasa pemrograman yang digunakan adalah bahasa pemrograman Go, sesuai dengan modul yang menjadi acuan praktikum. Golang (atau) Go adalah bahasa pemrograman baru, yang mulai dilirik oleh para developer karena kelebihan-kelebihan yang dimilikinya. Sudah banyak Perusahaan besar yang menggunakan bahasa ini untuk produk-produk mereka hingga di level production.

B. Komentar

Komentar biasa dimanfaatkan untuk menyisipkan catatan pada kode program, menulis penjelasan atau deskripsi mengenai suatu blok kode, atau bisa juga digunakan untuk me-remark kode (men-non-aktifkan kode yang tidak digunakan). Komentar akan diabaikan Ketika kompilasi maupun eksekusi program.

Ada 2 jenis komentar di Golang, yaitu inline dan multiline.

1. Komentar Inline

Penulisan komentar jenis ini diawali dengan tanda *double slash* (//) lalu diikuti pesan komentarnya. Komentar inline hanya berlaku untuk satu baris pesan saja. Jika pesan komentar lebih dari satu baris, maka tanda *double slash* harus ditulis lagi di baris selanjutnya.

2. Komentar Multiline

Komentar yang cukup panjang akan lebih rapi jika ditulis menggunakan teknik komentar multiline. Ciri dari komentar jenis ini adalah penulisannya diawali dengan tanda (/*) dan diakhiri (*/).

C. Variabel

Golang mengadopsi 2 jenis penulisan variabel, yang dituliskan tipe data-nya dan yang tidak. Kedua cara tersebut intinya adalah sama, pembedanya hanyalah cara penulisannya saja. Untuk penulisan variabel dengan tipe data, keyword *var* digunakan untuk deklarasi variabel kemudian diakhiri dengan tipe data misalnya *string*. Kemudian untuk penulisan variabel tanpa tipe data, variabel dideklarasikan dengan menggunakan metode type inference. Penandanya tipe data tidak dituliskan pada saat deklarasi. Pada penggunaan metode ini, operand (=) harus diganti dengan (:=) dan keyword *var* dihilangkan.

Golang memiliki aturan unik yang tidak dimiliki bahasa lain, yaitu tidak boleh ada satupun variabel yang menganggur. Artinya, semua variabel yang dideklarasikan harus digunakan. Jika terdapat variabel yang tidak digunakan tapi dideklarasikan, program akan gagal dikompilasi. Untuk mengatasi itu, golang memiliki variabel yaitu underscore. Underscore (_) adalah predefined variabel yang bisa dimanfaatkan untuk menampung nilai yang tidak dipakai.

D. Tipe Data

Golang mengenal beberapa jenis tipe data, diantaranya adalah tipe data numerik (decimal dan non-desimal), string, dan boolean.

1. Tipe Data Numerik Non-Desimal (uint, int)
2. Tipe Data Numerik Desimal (float64, float32)
3. Tipe Data Bool (true, false)
4. Tipe Data String (string, " ")

E. Operator Aritmatika

Operator aritmatika merupakan operator yang digunakan untuk operasi yang sifatnya perhitungan. Golang mendukung beberapa operator aritmatika standar, yaitu:

1. Penjumlahan (+)
2. Pengurangan (-)
3. Perkalian (*)
4. Pembagian (/)
5. Modulus atau sisa hasil pembagian (%)

F. Seleksi Kondisi

Seleksi kondisi pada program berguna untuk mengontrol sebuah blok kode yang akan dieksekusi. Yang dijadikan acuan oleh selksi kondisi adalah nilai bertipe bool, bisa berasal dari variabel, ataupun hasil operasi perbandingan. Nilai tersebut menentukan blok kode mana yang akan dieksekusi. Go memiliki 2 macam keyword untuk selesksi kondisi, yaitu if else dan switch.

1. **If Expression**

   If adalah salah satu kata kunci yang digunakan dalam percabangan. Percabangan artinya kitabisa mengeksekusi kode program tertentu ketika suatu kondisi terpenuhi. Hampir semua bahasa pemrograman mendukung if expression.

2. **Else if expression**

   Terkadang kita butuh membuat beberapa kondisi. Kasus seperti ini dapat menggunakan else if expression. If mendukung short statement sebelum kondisi.

Hal ini sangat cocok untuk membuat statement yang sederhana sebelum melakukan pengecekan terhadap kondisi.

3. **Switch-Case**

   Switch merupakan seleksi kondisi yang sifatnya fokus pada satu variabel, lalu kemudian di-cek nilainya. Contoh sederhananya seperti penentuan apakah nilai variabel x adalah: 1, 2, 3, atau lainnya. Perlu diketahui, switch pada pemrograman Go memiliki perbedaan dibanding bahasa lain. Di Go, ketika sebuah case terpenuhi, tidak akan dilanjutkan ke pengecekan case selanjutnya, meskipun tidak ada keyword "break" di situ. Konsep ini berkebalikan dengan switch pada umumnya pemrograman lain (yang ketika sebuah case terpenuhi, maka akan tetap dilanjut mengecek case selanjutnya kecuali ada keyword "break").

G. Perulangan

Perulangan merupakan proses mengulang dan mengeksekusi blok kode tanpa henti sesuai dengan kondisi yang dijadikan acuan. Biasanya disiapkan variabel untuk iterasi atau penanda kapan perulangan akan dihentikan.

a. For Loop

For loop merupakan statement perulangan dasar dan cukup sering ditemui. Format for loop yaitu sebagai berikut.

- *Init Statement*: bagian ini akan dieksekusi sebelum perulangan dimulai. Biasanya diisi dengan mendeklarasi variabel iterasi.
- *Condition Expression*: bagian ini akan dicek dan dieksekusi setiap perulangan yang dilakukan, jika true maka perulangan akan terus berjalan hingga kondisi bernilai false.
- *Post Statement*: statement ini akan dieksekusi pada akhir iterasi. Jika terdapat range, maka perulangan akan dieksekusi untuk setiap item pada range.

b. While Loop

While loop merupakan perulangan yang akan terus berjalan hingga suatu kondisi terpenuhi. Penulisan while loop adalah dengan menuliskan kondisi setelah keyword for (hanya kondisi). Deklarasi dan iterasi variabel counter tidak dituliskan setelah keyword, hanya kondisi perulangan saja. Konsepnya mirip seperti while milik bahasa pemrograman lain.

c. Repeat Until

Untuk Repeat Until ini mirip seperti for loop biasa namun hanya menggunakan inisiasi dan kondisi saja.

H. Fungsi

Dalam konteks pemrograman, fungsi adalah sekumpulan blok kode yang dibungkus dengan nama tertentu. Penerapan fungsi yang tepat akan menjadikan kode lebih modular dan juga *dry* (singkatan dari *don't repeat yourself*) yang artinya kita tidak perlu menuliskan banyak kode untuk kegunaan yang sama berulang kali. Cukup deklarasikan sekali saja blok kode sebagai suatu fungsi, lalu panggil sesuai kebutuhan.

1. Penerapan Fungsi

Sebenarnya kita sudah mengimplementasikan fungsi pada banyak praktek sebelumnya, yaitu fungsi main(). Fungsi main() sendiri merupakan fungsi utama pada program Go, yang akan dieksekusi ketika program dijalankan.

Selain fungsi main(), kita juga bisa membuat fungsi lainnya. Dan caranya cukup mudah, yaitu dengan menuliskan keyword func kemudian diikuti nama fungsi, lalu kurung () (yang bisa diisi parameter), dan diakhiri dengan kurung kurawal untuk membungkus blok kode.

Parameter merupakan variabel yang menempel di fungsi yang nilainya ditentukan saat pemanggilan fungsi tersebut. Parameter sifatnya opsional, suatu fungsi bisa tidak memiliki parameter, atau bisa saja memeliki satu atau banyak parameter (tergantung kebutuhan).

2. Fungsi dengan Nilai Balik / Return Value

Selain parameter, fungsi bisa memiliki attribute **return value** atau nilai balik. Fungsi yang memiliki return value, saat deklarasinya harus ditentukan terlebih dahulu tipe data dari nilai baliknya.

3. Fungsi tanpa Nilai Balik / Return Value

Fungsi juga dapat tidak memiliki nilai balik yang dapat disebut juga sebagai Prosedur. 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.

I. Array

Array adalah kumpulan data bertipe sama, yang disimpan dalam sebuah variabel. Array memiliki kapasitas yang nilainya ditentukan pada saat pembuatan, menjadikan elemen/data yang disimpan di array tersebut jumlahnya tidak boleh melebihi yang sudah dialokasikan.

Default nilai tiap elemen array pada awalnya tergantung dari tipe datanya. Jika `int` maka tiap element zero value-nya adalah `0`, jika `bool` maka `false`, dan seterusnya. Setiap elemen array memiliki indeks berupa angka yang merepresentasikan posisi urutan elemen tersebut. Indeks array dimulai dari 0.

II. GUIDED

1. **Source Code:**

```
package main
import (
    "fmt"
    "sort"
)

func main() {
    // Array awal
    data := []float64{2, 7, 9, 1, 5, 6, 18, 13, 25, 20}
    target := 13.0

    fmt.Println("Sequential Search (data tidak perlu urut):")
    idxSeq, iterSeq := sequentialSearch(data, target)
    if idxSeq != -1 {
        fmt.Printf("Hasil: Ditemukan di indeks %d dalam %d langkah\n\n", idxSeq,
iterSeq)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d langkah\n\n", iterSeq)
    }

    // Binary search perlu array diurutkan
    sort.Float64s(data)
    fmt.Println("Binary Search (setelah data diurutkan):")
    fmt.Println("Data terurut:", data)

    idxBin, iterBin := binarySearch(data, target)
    if idxBin != -1 {
        fmt.Printf("Hasil: Ditemukan di indeks %d dalam %d langkah\n", idxBin,
iterBin)
    } else {
        fmt.Printf("Hasil: Tidak ditemukan setelah %d langkah\n", iterBin)
    }
}

func sequentialSearch(arr []float64, target float64) (int, int) {
    iterations := 0
    for i, val := range arr {
        iterations++
        fmt.Printf("Sequential step %d: cek arr[%d] = %.1f\n", iterations, i, val)
        if val == target {
            return i, iterations
        }
    }
    return -1, iterations
}
```

```
func binarySearch(arr []float64, target float64) (int, int) {
    iterations := 0
    low := 0
    high := len(arr) - 1

    for low <= high {
        iterations++
        mid := (low + high) / 2
        fmt.Printf("Binary step %d: cek arr[%d] = %.1f\n", iterations, mid, arr[mid])

        if arr[mid] == target {
            return mid, iterations
        } else if target < arr[mid] {
            high = mid -1
        } else {
            low = mid + 1
        }
    }
    return -1, iterations
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul8\103112400061_
Sequential Search (data tidak perlu urut):
Sequential step 1: cek arr[0] = 2.0
Sequential step 2: cek arr[1] = 7.0
Sequential step 3: cek arr[2] = 9.0
Sequential step 4: cek arr[3] = 1.0
Sequential step 5: cek arr[4] = 5.0
Sequential step 6: cek arr[5] = 6.0
Sequential step 7: cek arr[6] = 18.0
Sequential step 8: cek arr[7] = 13.0
Hasil: Ditemukan di indeks 7 dalam 8 langkah

Binary Search (setelah data diurutkan):
Data terurut: [1 2 5 6 7 9 13 18 20 25]
Binary step 1: cek arr[4] = 7.0
Binary step 2: cek arr[7] = 18.0
Binary step 3: cek arr[5] = 9.0
Binary step 4: cek arr[6] = 13.0
Hasil: Ditemukan di indeks 6 dalam 4 langkah
```

Penjelasan:

Program untuk mencari nilai acak dari suatu array dengan menerapkan algoritma “Sequential Search” dan “Binary Search”. Sequential search dilakukan dengan cara cek satu persatu dari index ke-0 hingga index ke-n hingga ditemukan nilai yang ingin didapat. Sedangkan binary search dilakukan dengan cara membagi 2 index ke-0 dan index ke-n untuk mencari titik tengah, dilakukan terus menerus hingga bertemu dengan nilai yang ingin didapat.

2. **Source Code:**

```
package main
import (
    "fmt"
    "sort"
)

type mahasiswa struct {
    nama, nim, kelas, jurusan string
    ipk                       float64
}

type arrMhs [2023]mahasiswa

// Sequential Search berdasarkan nama
func SeqSearch_3(T arrMhs, n int, X string) (int, int) {
    var found int = -1
    var j int = 0
    var iterasi int = 0

    for j < n && found == -1 {
        iterasi++
        if T[j].nama == X {
            found = j
        }
        j++
    }
    return found, iterasi
}

// Binary Search berdasarkan NIM (data harus sudah terurut berdasarkan nim)
func BinarySearch_3(T arrMhs, n int, X string) (int, int) {
    var found int = -1
    var med int
    var kr int = 0
    var kn int = n - 1
    var iterasi int = 0

    for kr <= kn && found == -1 {
        iterasi++
        med = (kr + kn) / 2
        if X < T[med].nim {
            kn = med - 1
        } else if X > T[med].nim {
            kr = med + 1
        } else {
            found = med
        }
    }
```

```
    return found, iterasi
}
func main() {
    var data arrMhs
    n := 10

    // Mengisi data secara manual
    data = arrMhs{
        {nama: "Ari", nim: "2201", kelas: "A", jurusan: "Informatika", ipk: 3.4},
        {nama: "Budi", nim: "2203", kelas: "A", jurusan: "Informatika", ipk: 3.6},
        {nama: "Cici", nim: "2202", kelas: "B", jurusan: "Sistem Informasi", ipk: 3.5},
        {nama: "Dina", nim: "2205", kelas: "A", jurusan: "Informatika", ipk: 3.3},
        {nama: "Eko", nim: "2204", kelas: "B", jurusan: "Sistem Informasi", ipk: 3.7},
        {nama: "Fajar", nim: "2206", kelas: "C", jurusan: "Informatika", ipk: 3.1},
        {nama: "Gita", nim: "2209", kelas: "C", jurusan: "Informatika", ipk: 3.8},
        {nama: "Hana", nim: "2208", kelas: "B", jurusan: "Sistem Informasi", ipk: 3.2},
        {nama: "Iwan", nim: "2207", kelas: "C", jurusan: "Informatika", ipk: 3.0},
        {nama: "Joko", nim: "2210", kelas: "A", jurusan: "Informatika", ipk: 3.9},
    }

    // Pencarian Sequential Search berdasarkan nama
    namaDicari := "Fajar"
    idxSeq, iterSeq := SeqSearch_3(data, n, namaDicari)

    fmt.Printf("Sequential Search - Cari nama '%s'\n", namaDicari)
    if idxSeq != -1 {
        fmt.Printf("Ditemukan di indeks: %d, Iterasi: %d\n", idxSeq, iterSeq)
    } else {
        fmt.Printf("Tidak ditemukan, Iterasi: %d\n", iterSeq)
    }

    // Urutkan data berdasarkan NIM untuk binary search
    sort.Slice(data[:n], func(i, j int) bool {
        return data[i].nim < data[j].nim
    })

    // Pencarian Binary Search berdasarkan NIM
    nimDicari := "2206"
    idxBin, iterBin := BinarySearch_3(data, n, nimDicari)

    fmt.Printf("\nBinary Search - Cari NIM '%s'\n", nimDicari)
    if idxBin != -1 {
        fmt.Printf("Ditemukan di indeks: %d, Iterasi: %d\n", idxBin, iterBin)
    } else {
        fmt.Printf("Tidak ditemukan, Iterasi: %d\n", iterBin)
    }
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul8\103112400061_
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

Binary Search - Cari NIM '2206'
Ditemukan di indeks: 5, Iterasi: 3
```

Penjelasan:

Program untuk mencari nilai acak dari suatu array 5 kolom dengan menerapkan algoritma “Sequential Search” dan “Binary Search”. Sequential search dilakukan dengan cara cek satu persatu dari index ke-0 hingga index ke-n hingga ditemukan nilai yang ingin didapat. Sedangkan binary search dilakukan dengan cara membagi 2 index ke-0 dan index ke-n untuk mencari titik tengah, dilakukan terus menerus hingga bertemu dengan nilai yang ingin didapat.

III. UNGUIDED

1. Latihan no. 1

Source Code:

```
// KEISHIN NAUFA ALFARIDZHI
// 103112400061
package main

import (
    "fmt"
)

const N int = 1000

func main() {
    var (
        T [N]int
        x int = -1
    )

    for i := 0; x != 0; i++{
        fmt.Scan(&x)
        T[i] = x
    }

    pilkart(T)
}
```

```
func pilkart(T [N]int) {

    var (
        sahCounter int = 0
        allSuara int = 0
        bilPrinted [21]bool
    ) // 103112400061_Keishin

    for i := 0; T[i] != 0; i++ {
        allSuara++
    }

    for i := 0; i < allSuara; i++{
        val := T[i]
        if val >= 1 && val <= 20 {
            if !bilPrinted[val] {
                bilPrinted[val] = true
            }
            sahCounter++
        }
    }

    fmt.Printf("Suara masuk: %d\n", allSuara)
    fmt.Printf("Suara sah: %d\n", sahCounter)

    for i := 1; i <= 20; i++ {
        if bilPrinted[i] {
            tempCounter := 0

            for j := 0; j < allSuara; j++ {
                if T[j] == i {
                    tempCounter++
                }
            }
            fmt.Printf("%d: %d\n", i, tempCounter)
        }
    }
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul8\103112400061_
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
1: 1
2: 1
3: 2
7: 1
18: 1
19: 2
```

Deskripsi Program:

Menghitung suara pemilihan ketua RT dengan algoritma sequential search. Berikut merupakan algoritma pada function pilkart() :

1) Menghitung total suara

   Menggunakan iterasi dengan kondisi array[i] != 0.

2) Mencari suara sah dan menentukan nilai unique agar value yang sama tidak dicetak lebih dari sekali.

   Dengan kondisi val >= 1 & val <= 20 (suara sah). Lalu didalamnya cek apakah suatu value sudah unique atau belum dengan cek bilPrinted kemudian mengubahnya menjadi true jika suatu value belum unique. Setelah itu menambah sahCounter +1 tiap perulangan.

3) Print suara masuk dan suara sah

4) Hitung suara dan print

   Cek apakah value unique, jika iya maka lanjut. Lalu cek apakah value sama dengan suara sah, jika iya maka counter +1. Lalu cetak tiap value dengan total suara.

2. Latihan no. 2

Source Code:

```
// KEISHIN NAUFA ALFARIDZHI
// 103112400061
package main

import (
    "fmt"
)

const N int = 1000

func main() {
    var (
        T [N]int
        x int = -1
    )

    for i := 0; x != 0; i++{
        fmt.Scan(&x)
        T[i] = x
    }
```

```
    selectedrt(T)
}

func selectedrt(T [N]int) {

    var (
        sahCounter int = 0
        allSuara int = 0
        winnerKetua, winnerWakil int
        tempCounter1, tempCounter2 int
        bilPrinted [21]bool
        counter [21]int
    )

    for i := 0; T[i] != 0; i++ {
        allSuara++
    }

    for i := 0; i < allSuara; i++{
        val := T[i]
        if val >= 1 && val <= 20 {
            if !bilPrinted[val] {
                bilPrinted[val] = true
            }
            sahCounter++
        }
    }

    for i := 0; i < allSuara; i++ {
        val := T[i]
        if val >= 1 && val <= 20 { // 103112400061_Keishin
            counter[val]++
        }
    }

    for i := 1; i <= 20; i++ {
        if counter[i] > tempCounter1 {
            winnerWakil = winnerKetua
            tempCounter2 = tempCounter1

            winnerKetua = i
            tempCounter1 = counter[i]
        } else if counter[i] >= tempCounter2 && counter[i] <= tempCounter1 {
            winnerWakil = i
            tempCounter2 = counter[i]
        }
    }

    fmt.Printf("Suara masuk: %d\n", allSuara)
    fmt.Printf("Suara sah: %d\n", sahCounter)
```

```
    fmt.Printf("Ketua RT: %d\n", winnerKetua)
    fmt.Printf("Wakil Ketua RT: %d\n", winnerWakil)
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul8\103112400061_
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 3
Wakil Ketua RT: 19
```

Deskripsi Program:

Menghitung suara pemilihan ketua RT dengan algoritma sequential search. Untuk menghitung total suara dan mencari suara sah masih sama dengan soal sebelumnya. Lalu untuk pemilihan Ketua dan Wakil Ketua adalah, untuk setiap suara sah (1 – 20):

1) Untuk Ketua RT, cari mana suara yang paling tinggi dengan cek counter[i] > tempCounter1 lalu algoritma akan berjalan sesuai dengan code diatas untuk mengisi winnerKetua dengan suara terbanyak pertama
2) Untuk Wakil Ketua RT, cari suara paling tinggi kedua dengan cek counter[i] ≥ tempCounter2 & counter[i] ≤ tempCounter2 lalu algoritma akan berjalan sesuai dengan code diatas untuk mengisi winnerWakil dengan suara terbanyak kedua.

3. Latihan no. 3

Source Code:

```
// KEISHIN NAUFA ALFARIDZHI
// 103112400061
package main
import "fmt"

const NMAX int = 1000000
var data [NMAX]int

func main() {
    var n, k int

    fmt.Scan(&n, &k)
    isiArray(n)
    indeks := posisi(n, k)
    if indeks != -1 {
        fmt.Println(indeks)
    } else {
        fmt.Println("TIDAK ADA")
```

```
    }
}

func isiArray(n int) {
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i])
    }
}

func posisi(n, k int) int {
    left, right := 0, n

    for left <= right {
        mid := (left + right) / 2
        if data[mid] == k {
            return mid
        } else if k < data[mid] {
            right = mid -1
        } else {
            left = mid + 1
        }
    }

    return -1
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul8\103112400061_
12 534
1 3 8 16 32 123 323 323 534 543 823 999
8

D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul8\103112400061_
12 535
1 3 8 16 32 123 323 323 534 543 823 999
TIDAK ADA
```

Deskripsi Program:

Program untuk mencari index dari suatu bilangan dalam array menggunakan algoritma binary search. Mengisi nilai array dengan prosedur isiArray(). Lalu algoritma binary search untuk mencari indexnya, membagi 2 index ke-0 dan index ke-n untuk mencari titik tengah, dilakukan terus menerus hingga bertemu dengan nilai yang ingin didapat. Tapi dibanding dengan mengembalikan nilai atau value dari index-k, kita akan kembalikan index nya secara langsung.

IV. KESIMPULAN

Pada praktikum ini telah dibahas perihal cara menentukan nilai acak pada array. Mencari nilai acak dari suatu array dapat berguna dalam mendapatkan value yang diinginkan dibandingkan harus mencari value secara manual. Telah dipelajari cara mencari nilai acak dengan algoritma “Sequential Search” dan “Binary Search”. Tentunya ini akan lebih memudahkan untuk bekerja pada tipe data array.

V. DAFTAR PUSTAKA

Noval Agung Prayogo. *Dasar Pemrograman Golang.* Diakses pada 01 Oktober 2024. https://dasarpemrogramangolang.novalagung.com

Annisa Nur Isnaeni. *Golang — Seleksi Kondisi.* Diakses pada 01 Oktober 2024.

https://medium.com/@annisaisna/golang-seleksi-kondisi-f988ead004b4

Parvez Alam, *Golang for loop example | Golang Loops Tutorial – Phpflow.com* https://medium.com/@parvez1487/golang-for-loop-example-golang-loops-tutorial-phpflow-com-f4b2b0e57944