

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 8
PENCARIAN NILAI ACAK PADA HIMPUNAN DATA**



Oleh:

SAVILA NUR FADILLA

103112400031

IF-12-01

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO
2025**

I. DASAR TEORI

8.1 Sequential Search

Pencarian secara sekuensial ini adalah pencarian yang dilakukan dari data pertama, kedua hingga terakhir secara satu persatu dan berurutan. Ciri khas dari pencarian ini adalah proses pencarian akan berhenti ketika data yang dicari ditemukan, walaupun masih terdapat data yang belum dicek nilainya.

Algoritma ini dikenal dengan nama Sequential Search, karena prosesnya melakukan pengecekan setiap elemen array secara satu persatu dan sekuensial dari data pertama hingga ditemukan atau data terakhir.

- 1.) Asumsi terdapat suatu array of integer T dengan indeks dari 0 hingga N-1, dan suatu nilai yang dicari pada array T, yaitu X.
- 2.) Status pencarian digunakan untuk menandakan data yang dicari ditemukan atau tidak, misalnya found dengan tipe boolean.
- 3.) Pencarian dilakukan dari T[0] sampai ke T[N-1], setiap kali perbandingan dengan X, update nilai found.
- 4.) Perulangan harus dihentikan apabila status pencarian found bernilai true (data ditemukan) atau T[N-1] telah dicek.

	Notasi Algoritma	Notasi dalam bahasa Go
1	found \leftarrow false	found = false
2	i \leftarrow 0	i = 0
3	while i < n and not found do	for i < n && !found {
4	found \leftarrow T[i] == X	found = T[i] == X
5	i \leftarrow i + 1	i = i + 1
6	endwhile	}

8.2 Binary Search

Ide algoritma adalah: (dengan asumsi data terurut dari kecil membesar (ascending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan")

- 1.) Ambil salah satu data dalam rentang data yang ada, algoritma di bawah menggunakan rentang dari kiri **kr** s.d. kanan **kn**. Untuk kemudahan dan alasan lainnya, biasanya yang diambil adalah paling tengah dalam rentang tersebut.
- 2.) Jika data terambil tersebut terlalu kecil, maka ubah/geser rentang data ke sebelah kanan posisi data tersebut. Ini karena jika data terambil terlalu kecil, maka semua data sebelah kirinya juga akan terlalu kecil dari yang ingin dicari.
- 3.) Begitu juga sebaliknya jika data terambil terlalu besar.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$kr \leftarrow 0$	$kr = 0$
2	$kn \leftarrow n-1$	$kn = n-1$
3	$found \leftarrow false$	$found = false$
4	while $kr \leq kn$ and not found do	for $kr \leq kn \ \&\& \ !found$ {
5	$med \leftarrow (kr+kn) \text{ div } 2$	$med = (kr+kn) / 2$
6	if $a[med] < X$ then	if $a[med] < X$ {
7	$kr \leftarrow med + 1$	$kr = med + 1$
8	else if $a[med] > X$ then	} else if $a[med] > X$ {
9	$kn \leftarrow med - 1$	$kn = med$
10	else	} else {
11	$found \leftarrow true$	$found = true$
12	endif	}
13	endwhile	}

8.3 Pencarian pada Array Bertipe Data Struct

Algoritma pencarian pada array bertipe data struct tidak berbeda jauh dengan tipe data dasar, kita hanya perlu menambahkan field kategori dari pencarian yang akan dilakukan. Khusus untuk algoritma binary search, keterurutan array harus sama dengan field kategori dari pencariannya. Misalnya apabila pencarian berdasarkan nim mahasiswa, maka algoritma binary search hanya akan bisa digunakan apabila array terurut berdasarkan nim. Algoritma binary search tidak akan berjalan apabila array terurut berdasarkan nim mahasiswa, tetapi pencarian yang dilakukan berdasarkan filed nama atau kategori yang lain selain nim. Solusinya adalah array harus diurutkan berdasarkan nim terlebih dahulu, atau gunakan algoritma sequential search.

II. GUIDED

1.) Source Code

```
package main

import (
    "fmt"
    "sort"
)

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
}
```

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

    fmt.Println("Sequential Search (data tidak perluurut)")
    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)
    }

    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)
    }
}
```

Output :

```
PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\103112400031_MODUL8> go run "c:\103112400031_MODUL8\103112400031_guided1.go"
Sequential Search (data tidak perluurut)
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 ini bertujuan untuk mencari posisi sebuah nilai dalam array (indeks ke berapa) dan jumlah langkah yang diperlukan menggunakan dua metode, sequential search dan binary search. Program meminta input array data (slice) dan target yang dicari. Kemudian program akan membandingkan dua metode dalam pencarian data, menggunakan sequential search dan binary search. Pertama program menjalankan sequential search, metode pencarian elemen pertama hingga akhir secara urut. Setiap elemen dicek satu satu sampai ditemukan yang sama atau cocok dengan target. Program akan mengembalikan indeks jika ditemukan, dan -1 jika tidak. Kemudian program menjalankan binary search, metode pencarian dengan mengurutkan data dahulu menggunakan sort.Float64s. Program membandingkan nilai tengah array dengan target, kalau nilai tengah lebih besar pencarian dilanjutkan ke separuh kiri dan kalau lebih kecil dilanjut ke separuh kanan. Proses berulang sampai ditemukan yang sama atau cocok dengan target. Program akan mengembalikan indeks jika ditemukan, dan -1 jika tidak. Outputnya adalah langkah pencarian kedua metode, informasi apakah target ditemukan dan indeks di mana target ditemukan pada masing masing metode, serta jumlah langkah yang diperlukan masing masing metode.

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)
    }

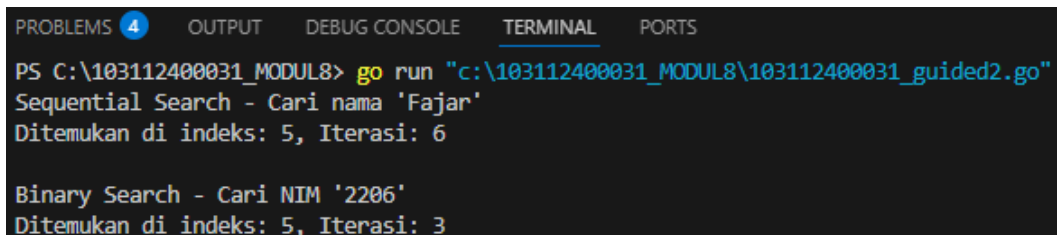
    // Urutkan data berdasarkan NIM untuk binary search
    sort.Slice(data[:n], func(i, j int) bool {
        return data[i].nim < data[j].nim
    })

    // 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 :



```

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\103112400031_MODUL8> go run "c:\103112400031_MODUL8\103112400031_guided2.go"
Sequential Search - Cari nama 'Fajar'
Ditemukan di indeks: 5, Iterasi: 6

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

```

Penjelasan : Program ini bertujuan untuk mencari posisi data mahasiswa dalam array (slice) dan menghitung jumlah iterasi menggunakan dua metode, sequential search dan binary search. Data mahasiswa diinput langsung dalam program, dan program meminta input dua pencarian, satu nama mahasiswa dan satu NIM yang akan dicari. Pertama program menjalankan sequential search, metode pencarian elemen pertama hingga akhir secara urut. Proses berulang sampai ditemukan nama mahasiswa yang dicari atau hingga seluruh data diperiksa. Kemudian program menjalankan binary search, metode pencarian dengan mengurutkan data dahulu menggunakan sort.Slice. Lalu program akan membagi data menjadi dua bagian dan melanjutkan pencarian ke bagian yang relevan (kiri atau kanan). Proses berulang sampai ditemukan NIM yang dicari. Outputnya adalah informasi apakah nama dan NIM ditemukan dan ditemukan di indeks ke berapa serta jumlah iterasi yang diperlukan masing masing metode.

III. UNGUIDED

1.) Source code

```
// Savila Nur Fadilla
// 103112400031
package main

import (
    "fmt"
    "sort"
)

func main() {
    var input int
    suaraMasuk := 0
    suaraSah := 0
    suara := make(map[int]int)

    for {
        fmt.Scan(&input)
        if input == 0 {
            break
        }

        suaraMasuk++

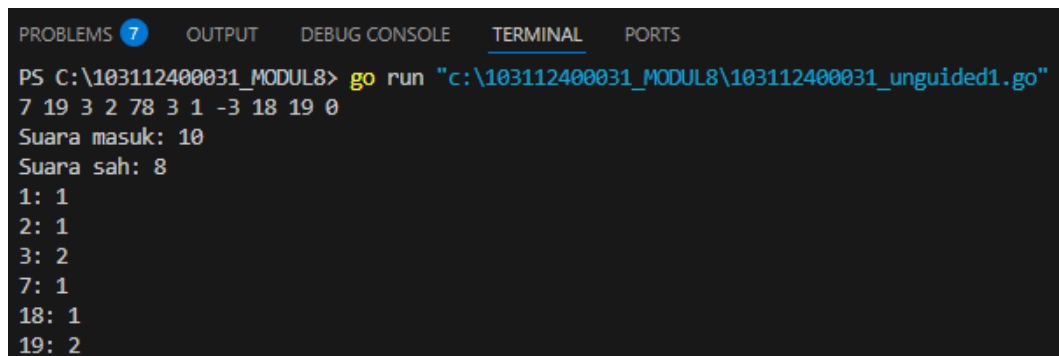
        if input >= 1 && input <= 20 {
            suara[input]++
            suaraSah++
        }
    }

    fmt.Println("Suara masuk:", suaraMasuk)
    fmt.Println("Suara sah:", suaraSah)

    var calon []int
    for k := range suara {
        calon = append(calon, k)
    }
    sort.Ints(calon)

    for _, c := range calon {
        fmt.Printf("%v: %v\n", c, suara[c])
    }
}
```

Output :

A screenshot of a Go IDE terminal window. The terminal has tabs for PROBLEMS (7), OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The command prompt shows the execution of a Go program with the command: `PS C:\103112400031_MODUL8> go run "c:\103112400031_MODUL8\103112400031_unguided1.go"`. The program outputs a list of numbers: `7 19 3 2 78 3 1 -3 18 19 0`. It then displays the total number of valid votes: `Suara masuk: 10` and the total number of invalid votes: `Suara sah: 8`. Finally, it lists the votes for each candidate: `1: 1`, `2: 1`, `3: 2`, `7: 1`, `18: 1`, and `19: 2`.

```
PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\103112400031_MODUL8> go run "c:\103112400031_MODUL8\103112400031_unguided1.go"
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
```

Penjelasan : Program ini bertujuan untuk membaca, memvalidasi, dan menghitung suara yang diberikan dalam pemilihan ketua RT (pilkart). Program meminta input data yang berisi bilangan bulat yang kadang disisipi dengan data yang tidak valid (satu baris saja). Data valid adalah integer dengan nilai antara 1 sampai 20, dan data akan berakhir jika kita menginput angka 0. Jika angka yang diinput 1 sampai 20 dianggap suara sah, tapi jika tidak tetap dianggap suara masuk. Output berupa total suara masuk, total suara sah, dan sejumlah baris yang mencetak data para calon mana saja yang mendapatkan suara dan jumlah suara yang didapatkan masing masing calon.

2.) Source code

```
// Savila Nur Fadilla
// 103112400031

package main

import (
    "fmt"
    "sort"
)

func main() {
    var input int
    suaraMasuk := 0
    suaraSah := 0
    suara := make(map[int]int)

    for {
        fmt.Scan(&input)
        if input == 0 {
            break
        }
        suaraMasuk++
        if input >= 1 && input <= 20 {
            suara[input]++
            suaraSah++
        }
    }
    fmt.Println("Suara masuk:", suaraMasuk)
    fmt.Println("Suara sah:", suaraSah)

    type calon struct {
        nomor, jumlah int
    }
    var data []calon
    for k, v := range suara {
        data = append(data, calon{k, v})
    }

    sort.Slice(data, func(a, b int) bool {
        if data[a].jumlah == data[b].jumlah {
            return data[a].nomor < data[b].nomor
        }
        return data[a].jumlah > data[b].jumlah
    })
}
```

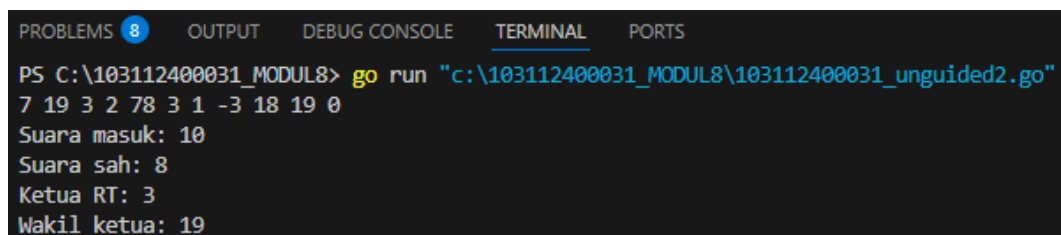
```

    })

    if len(data) > 0 {
        fmt.Println("Ketua RT:", data[0].nomor)
    }
    if len(data) > 1 {
        fmt.Println("Wakil ketua:", data[1].nomor)
    }
}

```

Output :



The screenshot shows a terminal window with the following output:

```

PROBLEMS 8 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\103112400031_MODUL8> go run "c:\103112400031_MODUL8\103112400031_unguided2.go"
7 19 3 2 78 3 1 -3 18 19 0
Suara masuk: 10
Suara sah: 8
Ketua RT: 3
Wakil ketua: 19

```

Penjelasan : Program ini bertujuan untuk membaca, memvalidasi, dan menghitung suara yang diberikan dalam pemilihan ketua RT (pilkart), mencari siapa pemenang pemilihan ketua RT. Program meminta input data yang berisi bilangan bulat yang kadang disisipi dengan data yang tidak valid (satu baris saja). Data valid adalah integer dengan nilai antara 1 sampai 20, dan data akan berakhir jika kita menginput angka 0. Jika angka yang diinput 1 sampai 20 dianggap suara sah, tapi jika tidak tetap dianggap suara masuk. Lalu program akan mencari pemenang pilkart, program juga akan menentukan bahwa wakil ketua RT adalah calon yang mendapatkan suara terbanyak kedua. Jika beberapa calon mendapatkan suara terbanyak yang sama, ketua terpilih adalah dengan nomor peserta yang paling kecil dan wakilnya dengan nomor peserta terkecil berikutnya. Output berupa total suara masuk, total suara sah, pasangan ketua RT dan wakil ketua.

3.) Source code

```
// Savila Nur Fadilla
// 103112400031

package main

import "fmt"

const NMAX = 1000000

var data [NMAX]int

func main() {
    var n, k int
    fmt.Scan(&n, &k)
    isiArray(n)

    pos := posisi(n, k)
    if pos == -1 {
        fmt.Println("TIDAK ADA")
    } else {
        fmt.Println(pos)
    }
}

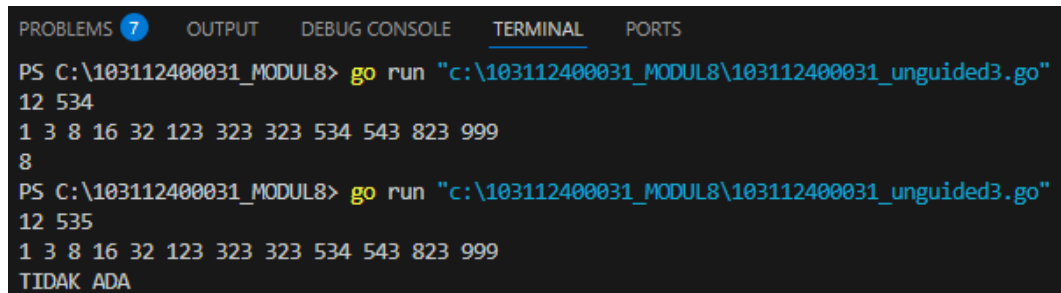
func isiArray(n int) {
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i])
    }
}

func posisi(n, k int) int {
    kiri := 0
    kanan := n - 1

    for kiri <= kanan {
        tengah := (kiri + kanan) / 2
        if data[tengah] == k {
            return tengah
        } else if data[tengah] < k {
            kiri = tengah + 1
        } else {
            kanan = tengah - 1
        }
    }
}
```

```
}  
    return -1  
}
```

Output :



The screenshot shows a Go IDE interface with tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. The TERMINAL tab is active, displaying the following text:

```
PS C:\103112400031_MODUL8> go run "c:\103112400031_MODUL8\103112400031_unguided3.go"  
12 534  
1 3 8 16 32 123 323 323 534 543 823 999  
8  
PS C:\103112400031_MODUL8> go run "c:\103112400031_MODUL8\103112400031_unguided3.go"  
12 535  
1 3 8 16 32 123 323 323 534 543 823 999  
TIDAK ADA
```

Penjelasan : Program ini bertujuan untuk menentukan apakah bilangan k ada di dalam array dan menentukan ada di indeks ke berapa bilangan k tersebut (indeks dimulai dari 0), serta mencetak pesan “TIDAK ADA” jika k tidak ditemukan. Program meminta input yang terdiri dari dua baris. Baris pertama berisi dua buah integer positif, yaitu n dan k menyatakan banyaknya data, dimana $1 < n \leq 1000000$. k adalah bilangan yang ingin dicari. Baris kedua berisi n buah data integer positif yang sudah terurut membesar. Lalu fungsi array digunakan untuk mengisi array dengan data, fungsi posisi digunakan untuk mencari posisi k di array menggunakan binary search. Program akan mengembalikan indeks jika ditemukan, dan -1 jika tidak. Output berupa posisi k pada indeks ke berapa (indeksnya dari 0) dan pesan “TIDAK ADA” jika k tidak ditemukan.

IV. KESIMPULAN

Berdasarkan laporan ini, dapat disimpulkan jika soal soal tersebut menggunakan array, slice, map, struct, prosedur dan fungsi, serta perulangan for dan percabangan if else. Hal ini dilakukan untuk mempermudah dalam membaca dan memahami program, serta membuat program lebih rapi, lebih efisien, terstruktur, dan mudah untuk dikembangkan.

REFERENSI

Telkom University. (2025). *Modul Praktikum Algoritma dan Pemrograman 2*.

<https://dasarpemrogramangolang.novalagung.com/>