

**LAPORAN PRAKTIKUM  
ALGORITMA PEMROGRAMAN 2**

**MODUL XII  
PENGURUTAN DATA**



Oleh:

**RIZKULLOH ALPRIYANSAH**

**2311102142**

**IF-11-08**

**PROGRAM STUDI S1 INFORMATIKA  
FAKULTAS INFORMATIKA  
UNIVERSITAS TELKOM PURWOKERTO  
2024**

## I. DASAR TEORI

Pengurutan data adalah proses menyusun elemen-elemen dalam suatu kumpulan data (array, slice, atau list) berdasarkan urutan tertentu, seperti secara **ascending (menaik)** atau **descending (menurun)**. Pengurutan ini bertujuan untuk mempermudah proses pencarian, pengolahan, dan analisis data.

Pengurutan dapat dibedakan menjadi dua jenis utama:

1. Pengurutan Internal: Dilakukan sepenuhnya di dalam memori utama (RAM). Cocok untuk dataset kecil hingga sedang.
2. Pengurutan Eksternal: Digunakan untuk dataset besar yang tidak dapat ditampung sepenuhnya di memori utama.

Beberapa algoritma pengurutan yang sering digunakan meliputi:

- Bubble Sort: Algoritma sederhana yang menukar elemen berurutan jika tidak sesuai dengan urutan yang diinginkan.
- Selection Sort: Memilih elemen terkecil (atau terbesar) dari data yang belum terurut, lalu meletakkannya di posisi yang sesuai.
- Insertion Sort: Menyisipkan elemen ke posisi yang benar dalam subset data yang sudah terurut.
- Merge Sort: Membagi data menjadi bagian-bagian kecil, mengurutkannya, lalu menggabungkannya kembali.
- Quick Sort: Memilih elemen pivot untuk membagi data ke dalam dua bagian dan mengurutkannya secara rekursif.
- Heap Sort: Menggunakan struktur data heap untuk menyusun data dalam urutan.

## Implementasi Pengurutan di Go

Di Go, pengurutan data biasanya dilakukan menggunakan fungsi bawaan dari paket `sort`. Paket ini mendukung pengurutan untuk tipe data dasar (seperti `int`, `float64`, dan `string`) dan pengurutan khusus untuk tipe data yang lebih kompleks.

### a. Pengurutan untuk Slice Tipe Data Dasar

Paket `sort` menyediakan fungsi seperti:

- `sort.Ints(slice []int)` untuk mengurutkan slice bertipe `int`.
- `sort.Strings(slice []string)` untuk mengurutkan slice bertipe `string`.
- `sort.Float64s(slice []float64)` untuk mengurutkan slice bertipe `float64`.

## Kompleksitas Waktu

Kompleksitas waktu pengurutan bergantung pada algoritma yang digunakan:

- Bubble Sort:  $O(n^2)$
- Selection Sort:  $O(n^2)$
- Insertion Sort:  $O(n^2)$
- Merge Sort:  $O(n \log n)$
- Quick Sort:  $O(n \log n)$  (rata-rata),  $O(n^2)$  (kasus terburuk)
- Heap Sort:  $O(n \log n)$

Paket `sort` di Go menggunakan algoritma hybrid yang menggabungkan Quick Sort, Heap Sort, dan Insertion Sort untuk kinerja optimal.

## Kesimpulan

Pengurutan data adalah aspek penting dalam pemrograman yang berguna untuk meningkatkan efisiensi pencarian dan manipulasi data. Go menyediakan fitur bawaan yang efisien melalui paket `sort`, mendukung pengurutan data dasar maupun tipe data kompleks.

## I. GUIDED

### Guided1

#### Source Code

```
package main

import "fmt"

// Fungsi untuk melakukan Selection Sort
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ { // Iterasi dari indeks pertama sampai kedua
        terakhir
        idxMin := i // Anggap elemen pada indeks i adalah yang terkecil

        // Cari elemen terkecil di sisa array
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[idxMin] {
                idxMin = j // Update indeks nilai terkecil
            }
        }

        // Tukar elemen terkecil dengan elemen di posisi i
        if idxMin != i {
            temp := arr[i]
            arr[i] = arr[idxMin]
            arr[idxMin] = temp
        }
    }
}

func main() {
    // Data yang akan diurutkan
    data := []int{64, 25, 12, 22, 11}

    fmt.Println("Data sebelum diurutkan:", data)

    // Panggil fungsi Selection Sort
    selectionSort(data)

    fmt.Println("Data setelah diurutkan:", data)
}
```

## ScreenShot Output



```
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan11 main
> go run .\Guided\Guided1\main.go
Data sebelum diurutkan: [64 25 12 22 11]
Data setelah diurutkan: [11 12 22 25 64]
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan11 main
```

## Deskripsi

Program ini mengimplementasikan algoritma *Selection Sort* dalam bahasa Go untuk mengurutkan array integer secara *ascending*. Fungsi `selectionSort` menerima array sebagai parameter, kemudian iterasi dilakukan untuk menemukan elemen terkecil di sisa array dan menukarnya dengan elemen pada indeks saat ini. Jika elemen terkecil ditemukan di indeks lain, proses *swap* dilakukan. Pada fungsi `main`, data array diberikan, kemudian fungsi `selectionSort` dipanggil untuk mengurutkan array tersebut, dan hasilnya ditampilkan sebelum dan setelah diurutkan.

## Guided2

### Source Code

```
package main

import "fmt"

type arrInt [4321]int

func selectionSort1(T *arrInt, n int) {
    var t, i, j, idx_min int

    i = 1
    for i <= n-1 {
        idx_min = i - 1
        j = i
        for j < n {
            if T[idx_min] > T[j] {
                idx_min = j
            }
            j = j + 1
        }
        t = T[idx_min]
        T[idx_min] = T[i-1]
        T[i-1] = t
        i = i + 1
    }
}
```

```

func main() {
    var data arrInt
    var n int

    fmt.Println("Masukkan jumlah elemen array: ")
    fmt.Scan(&n)

    fmt.Println("Masukkan elemen array:")
    for i := 0; i < n; i++ {
        fmt.Scan(&data[i])
    }

    fmt.Println("\nData sebelum diurutkan:")
    for i := 0; i < n; i++ {
        fmt.Printf("%d ", data[i])
    }
    selectionSort1(&data, n)
    fmt.Println("\n\nData setelah diurutkan:")
    for i := 0; i < n; i++ {
    }
}

```

## ScreenShot Output

```

RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan11 %main
> go run .\Guided\Guided2\main.go
Masukkan jumlah elemen array: 7
Masukkan elemen array:
5
6
4
3
2
1
8

Data sebelum diurutkan:
5 6 4 3 2 1 8

Data setelah diurutkan:
1 2 3 4 5 6 8
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan11 %main

```

## Deskripsi

Program di atas mengimplementasikan algoritma *Selection Sort* dalam bahasa Go untuk mengurutkan array yang elemen dan ukurannya dimasukkan oleh pengguna. Tipe data `arrInt` digunakan untuk mendeklarasikan array dengan kapasitas maksimum 4321 elemen. Fungsi `selectionSort1` menerima array dan jumlah elemennya sebagai parameter, menggunakan pointer untuk

memungkinkan modifikasi langsung pada array asli. Algoritma *Selection Sort* bekerja dengan mencari elemen terkecil di sisa array dan menukarnya dengan elemen pada posisi saat ini. Fungsi main meminta pengguna untuk memasukkan jumlah elemen array (n) dan nilainya, kemudian menampilkan array sebelum dan sesudah diurutkan. Seluruh proses pengurutan dilakukan secara langsung pada array menggunakan fungsi selectionSort1.

### Guided3

#### Source Code

```
package main

import "fmt"

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

type arrMhs [2023]mahasiswa

func selectionSort2(T *arrMhs, n int) {

    var i, j, idx_min int
    var t mahasiswa

    i = 1
    for i <= n-1 {
        idx_min = i - 1
        j = i
        for j < n {
            if T[idx_min].ipk > T[j].ipk {
                idx_min = j
            }
            j = j + 1
        }
        t = T[idx_min]
        T[idx_min] = T[i-1]
        T[i-1] = t
        i = i + 1
    }
}

func main() {
    var data arrMhs
```

```

var n int

fmt.Print("Masukkan jumlah mahasiswa: ")
fmt.Scan(&n)

fmt.Println("Masukkan data mahasiswa (Nama, NIM, Kelas, Jurusan,
IPK):")
for i := 0; i < n; i++ {
    fmt.Printf("Data mahasiswa ke-%d:\n", i+1)
    fmt.Print("Nama: ")
    fmt.Scan(&data[i].nama)
    fmt.Print("NIM: ")
    fmt.Scan(&data[i].nim)
    fmt.Print("Kelas: ")
    fmt.Scan(&data[i].kelas)
    fmt.Print("Jurusan: ")
    fmt.Scan(&data[i].jurusan)
    fmt.Print("IPK: ")
    fmt.Scan(&data[i].ipk)
}

fmt.Println("\nData mahasiswa sebelum diurutkan:")
for i := 0; i < n; i++ {
    fmt.Printf("%s (%s) - %s - %s - IPK: %.2f\n",
data[i].nama, data[i].nim, data[i].kelas, data[i].jurusan,
data[i].ipk)
}
selectionSort2(&data, n)

fmt.Println("\nData mahasiswa setelah diurutkan berdasarkan IPK
(terkecil-terbesar):")
for i := 0; i < n; i++ {
    fmt.Printf("%s (%s) - %s - %s - IPK: %.2f\n",
data[i].nama, data[i].nim, data[i].kelas, data[i].jurusan,
data[i].ipk)
}
}

```

## ScreenShot Output



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SEARCH ERROR powershell - Pertemuan11 + v
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan11 /main
> go run .\Guided\Guided3\main.go
Masukkan jumlah mahasiswa: 3
Masukkan data mahasiswa (Nama, NIM, Kelas, Jurusan, IPK):
Data mahasiswa ke-1:
Nama: aku
NIM: 12345
Kelas: 11D
Jurusan: informatika
IPK: 4.00
Data mahasiswa ke-2:
Nama: kamu
NIM: 22345
Kelas: 11D
Jurusan: informatika
IPK: 3.90
Data mahasiswa ke-3:
Nama: dia
NIM: 33345
Kelas: 11D
Jurusan: informatika
IPK: 3.80

Data mahasiswa sebelum diurutkan:
aku (12345) - 11D - informatika - IPK: 4.00
kamu (22345) - 11D - informatika - IPK: 3.90
dia (33345) - 11D - informatika - IPK: 3.80

Data mahasiswa setelah diurutkan berdasarkan IPK (terkecil-terbesar):
dia (33345) - 11D - informatika - IPK: 3.80
kamu (22345) - 11D - informatika - IPK: 3.90
aku (12345) - 11D - informatika - IPK: 4.00
```

## Deskripsi

Program ini mengimplementasikan algoritma *Selection Sort* dalam bahasa Go untuk mengurutkan data mahasiswa berdasarkan IPK (Indeks Prestasi Kumulatif) dari yang terkecil ke yang terbesar. Tipe data mahasiswa merepresentasikan data mahasiswa yang terdiri dari atribut nama, nim, kelas, jurusan, dan ipk. Tipe `arrMhs` digunakan untuk mendeklarasikan array mahasiswa dengan kapasitas maksimum 2023 elemen. Fungsi `selectionSort2` mengurutkan array mahasiswa menggunakan algoritma *Selection Sort*, di mana elemen dengan IPK terkecil dipindahkan ke posisi awal secara iteratif. Pada fungsi `main`, pengguna diminta memasukkan jumlah mahasiswa dan datanya, yang kemudian ditampilkan sebelum dan sesudah pengurutan. Program ini menampilkan daftar mahasiswa yang telah diurutkan berdasarkan IPK dalam format yang terstruktur.

## II. UNGUIDED

### Unguided1

#### Source Code

```
package main

import (
    "fmt"
)

func selectionSort_142(arr_142 []int) {
    n_142 := len(arr_142)
    for i_142 := 0; i_142 < n_142-1; i_142++ {
        minIdx_142 := i_142
        for j_142 := i_142 + 1; j_142 < n_142; j_142++ {
            if arr_142[j_142] < arr_142[minIdx_142] {
                minIdx_142 = j_142
            }
        }
        arr_142[i_142], arr_142[minIdx_142] = arr_142[minIdx_142],
arr_142[i_142]
    }
}

func main() {
    var n_142 int
    fmt.Print("Jumlah daerah kerabat Hercules tinggal: ")
    fmt.Scan(&n_142)

    if n_142 <= 0 || n_142 > 1000 {
        return
    }

    daerah_142 := make([][]int, n_142)

    for i_142 := 0; i_142 < n_142; i_142++ {
        var m_142 int
        fmt.Printf("\nJumlah kerabat di daerah %d: ", i_142+1)
        fmt.Scan(&m_142)

        if m_142 <= 0 || m_142 > 1000000 {
            return
        }

        daerah_142[i_142] = make([]int, m_142)
```

```

        fmt.Printf("Nomor rumah untuk daerah %d: ", i_142+1)
        for j_142 := 0; j_142 < m_142; j_142++ {
            fmt.Scan(&daerah_142[i_142][j_142])
        }
        selectionSort_142(daerah_142[i_142])
    }

    fmt.Println("\nHasil pengurutan nomor rumah:")
    for i_142 := 0; i_142 < n_142; i_142++ {
        fmt.Printf("Daerah %d: ", i_142+1)
        for j_142 := 0; j_142 < len(daerah_142[i_142]); j_142++ {
            fmt.Printf("%d ", daerah_142[i_142][j_142])
        }
        fmt.Println()
    }
}

```

## Screenshot

```

RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan11 > main
> go run .\Unguided\Unguided1\main.go
Jumlah daerah kerabat Hercules tinggal: 3

Jumlah kerabat di daerah 1: 3
Nomor rumah untuk daerah 1: 1 5 3

Jumlah kerabat di daerah 2: 2
Nomor rumah untuk daerah 2: 3 7

Jumlah kerabat di daerah 3: 4
Nomor rumah untuk daerah 3: 1 3 4 2

Hasil pengurutan nomor rumah:
Daerah 1: 1 3 5
Daerah 2: 3 7
Daerah 3: 1 2 3 4

```

## Deskripsi

Program di atas merupakan implementasi algoritma *Selection Sort* dalam bahasa Go yang digunakan untuk mengurutkan nomor rumah kerabat Hercules di berbagai daerah tempat tinggalnya. Pengguna diminta untuk memasukkan jumlah daerah ( $n_{142}$ ) dan jumlah kerabat di masing-masing daerah ( $m_{142}$ ), diikuti dengan daftar nomor rumah setiap kerabat. Data untuk setiap daerah disimpan dalam array dua dimensi `daerah_142`. Fungsi `selectionSort_142` bertugas mengurutkan nomor rumah secara *ascending* menggunakan algoritma

*Selection Sort.* Setelah semua data diurutkan, program menampilkan nomor rumah kerabat di setiap daerah dalam urutan yang telah disusun. Program juga memiliki validasi untuk memastikan jumlah daerah tidak melebihi 1000 dan jumlah kerabat tidak lebih dari 1 juta per daerah.

## Unguided2

### Source Code

```
package main

import (
    "fmt"
)

func selectionSortDesc(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] > arr[maxIdx] {
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i]
    }
}

func selectionSortAsc(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[minIdx] {
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], arr[i]
    }
}

func main() {
```

```

var n int
fmt.Print("Masukkan jumlah daerah (n): ")
fmt.Scan(&n)

results := make([][]int, n)

for i := 0; i < n; i++ {
    var m int
    fmt.Printf("Masukkan jumlah rumah kerabat di daerah %d: ", i+1)
    fmt.Scan(&m)

    rumahKerabat := make([]int, m)
    fmt.Printf("Masukkan nomor rumah kerabat di daerah %d: ", i+1)
    for j := 0; j < m; j++ {
        fmt.Scan(&rumahKerabat[j])
    }

    var ganjil, genap []int
    for _, rumah := range rumahKerabat {
        if rumah%2 == 1 {
            ganjil = append(ganjil, rumah)
        } else {
            genap = append(genap, rumah)
        }
    }

    fmt.Println("Ganjil sebelum sorting:", ganjil)
    fmt.Println("Genap sebelum sorting:", genap)

    selectionSortDesc(ganjil)
    selectionSortAsc(genap)

    fmt.Println("Ganjil setelah sorting:", ganjil)
    fmt.Println("Genap setelah sorting:", genap)

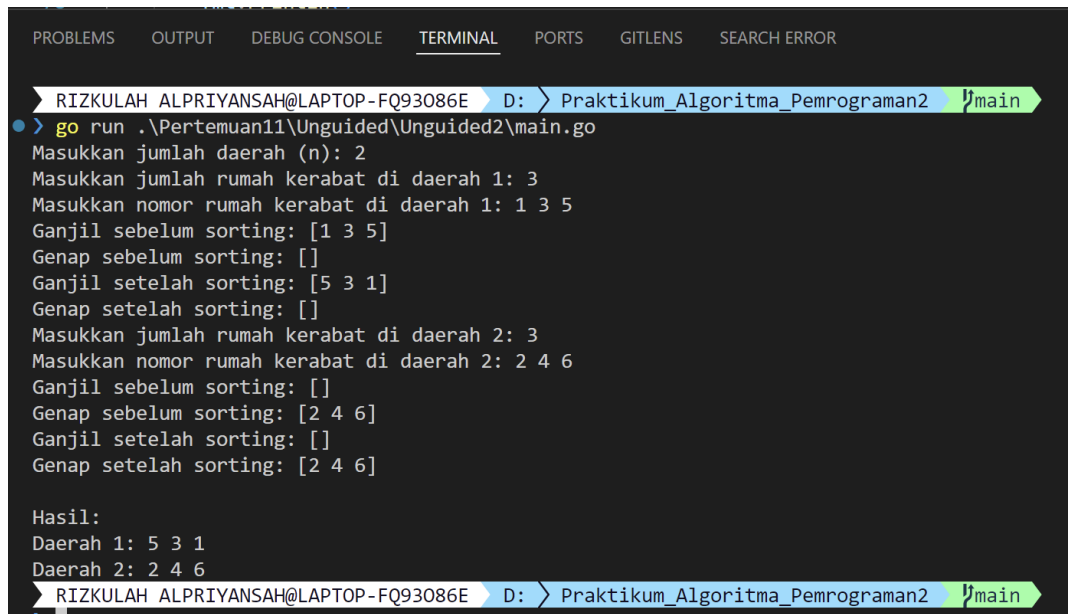
    results[i] = append(ganjil, genap...)
}

fmt.Println("\nHasil:")
for i, result := range results {
    fmt.Printf("Daerah %d: ", i+1)
    for _, rumah := range result {
        fmt.Printf("%d ", rumah)
    }
    fmt.Println()
}

```

```
}
```

## Screenshot



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS SEARCH ERROR
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 main
> go run .\Pertemuan11\Unguided\Unguided2\main.go
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat di daerah 1: 3
Masukkan nomor rumah kerabat di daerah 1: 1 3 5
Ganjil sebelum sorting: [1 3 5]
Genap sebelum sorting: []
Ganjil setelah sorting: [5 3 1]
Genap setelah sorting: []
Masukkan jumlah rumah kerabat di daerah 2: 3
Masukkan nomor rumah kerabat di daerah 2: 2 4 6
Ganjil sebelum sorting: []
Genap sebelum sorting: [2 4 6]
Ganjil setelah sorting: []
Genap setelah sorting: [2 4 6]

Hasil:
Daerah 1: 5 3 1
Daerah 2: 2 4 6
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 main
```

## Deskripsi

Program di atas mengimplementasikan algoritma selection sort untuk mengurutkan nomor rumah kerabat di beberapa daerah. Input terdiri dari jumlah daerah dan nomor rumah di tiap daerah. Setiap nomor rumah dikelompokkan menjadi ganjil dan genap: nomor ganjil diurutkan secara menurun (descending) dan nomor genap diurutkan secara menaik (ascending) menggunakan fungsi `selectionSortDesc` dan `selectionSortAsc`. Hasil pengelompokan dan pengurutan digabungkan kembali untuk setiap daerah dan ditampilkan di akhir. Program ini membantu mengorganisasi data nomor rumah secara terstruktur sesuai kategori dan urutan yang diinginkan.

## Unguided3

### Source Code

```
package main

import (
    "bufio"
    "fmt"
    "os"
```

```

"sort"
"strconv"
"strings"
)

func findMedian_142(numbers_142 []int) int {
    sort.Ints(numbers_142)
    n_142 := len(numbers_142)
    if n_142%2 == 1 {
        return numbers_142[n_142/2]
    }
    return (numbers_142[n_142/2-1] + numbers_142[n_142/2]) / 2
}

func main() {
    reader_142 := bufio.NewReader(os.Stdin)
    fmt.Println("Masukkan angka, gunakan 0 untuk menghitung median dan  
akhiri dengan -5313:")

    input_142, _ := reader_142.ReadString('\n')
    input_142 = strings.TrimSpace(input_142)

    data_142 := strings.Split(input_142, " ")
    var numbers_142 []int

    for _, value_142 := range data_142 {
        num_142, err_142 := strconv.Atoi(value_142)
        if err_142 != nil {
            fmt.Println("Input tidak valid, masukkan hanya angka.")
            return
        }

        if num_142 == -5313 {
            break
        } else if num_142 == 0 {
            if len(numbers_142) > 0 {
                median_142 := findMedian_142(numbers_142)
                fmt.Println(median_142)
            }
        } else {
            numbers_142 = append(numbers_142, num_142)
        }
    }
}

```

## Screenshot



```
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 main
> go run .\Pertemuan11\Unguided\Unguided3\main.go
Masukkan angka, gunakan 0 untuk menghitung median dan akhiri dengan -5313:
8 3 2 9 3 0 2 6 4 3 22 12 17 9 8 0 -5313
3
7
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum_Algoritma_Pemrograman2 main
```

## Deskripsi

Program di atas menerima input angka dari pengguna yang dimasukkan dalam satu baris, dipisahkan oleh spasi. Program akan mengolah angka-angka yang dimasukkan dan menghitung median ketika pengguna memasukkan angka 0. Proses dimulai dengan membaca input menggunakan `bufio.Reader`, lalu angka-angka diproses dan disimpan dalam slice `numbers_142`. Ketika angka 0 dimasukkan, program akan menghitung median dari angka yang sudah dimasukkan hingga saat itu menggunakan fungsi `findMedian_142`. Fungsi `findMedian_142` mengurutkan angka terlebih dahulu dan kemudian mencari median berdasarkan jumlah elemen (genap atau ganjil). Proses akan berhenti jika pengguna memasukkan angka -5313. Program ini juga memvalidasi input untuk memastikan hanya angka yang dimasukkan.



### **III. DAFTAR PUSTAKA**

- 1) Asisten praktikum, Akmelia Zahara dan Kyla Azzahra Kinan “Modul XII PENGURUTAN DATA” Learning Management System, 2024