

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII & XIII  
PENGURUTAN DATA**



**Disusun Oleh :**

**Muhammad Hamzah Haifan Ma'ruf**

**2311102091**

**IF-11-05**

**Dosen Pengampu :**

**Arif Amrulloh, S.Kom., M.Kom**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

Pengurutan data (sorting) adalah salah satu operasi dasar dalam pemrograman yang bertujuan untuk mengatur elemen-elemen dalam suatu struktur data berdasarkan urutan tertentu, seperti ascending (menaik) atau descending (menurun). Pengurutan mempermudah pencarian data, mengorganisasi data untuk analisis, dan mengurangi kompleksitas algoritma dalam pengolahan data selanjutnya. Dalam bahasa Golang, pengurutan data dapat dilakukan secara manual menggunakan algoritma tertentu atau dengan memanfaatkan pustaka bawaan seperti `sort`. Beberapa algoritma pengurutan manual yang umum digunakan adalah Insertion Sort, Bubble Sort, Merge Sort, dan Quick Sort.

Insertion Sort bekerja dengan menyisipkan elemen ke posisi yang sesuai dalam array yang sudah sebagian terurut. Algoritma ini memiliki kompleksitas waktu terbaik  $O(n)$  ketika data sudah terurut, namun terburuk  $O(n^2)$  ketika data terbalik. Bubble Sort membandingkan elemen yang berdekatan dan menukarnya jika urutannya salah, tetapi memiliki kompleksitas waktu yang tidak efisien  $O(n^2)$  dibandingkan algoritma lainnya. Merge Sort menggunakan pendekatan divide-and-conquer, yaitu membagi data menjadi dua bagian, mengurutkan setiap bagian, lalu menggabungkannya. Algoritma ini memiliki kompleksitas waktu  $O(n \log n)$ . Sementara itu, Quick Sort memilih pivot, membagi array menjadi elemen yang lebih kecil dan lebih besar dari pivot, lalu mengurutkannya secara rekursif. Algoritma ini memiliki rata-rata kompleksitas waktu  $O(n \log n)$ , namun dalam kasus terburuk bisa mencapai  $O(n^2)$ .

Golang juga menyediakan pustaka bawaan untuk pengurutan data melalui paket `sort`, yang lebih efisien dan mudah digunakan. Beberapa fungsi bawaan dalam pustaka ini antara lain `sort.Ints` untuk mengurutkan slice integer, `sort.Strings` untuk mengurutkan slice string, dan `sort.Float64s` untuk mengurutkan slice float. Penggunaan pustaka ini sangat disarankan dalam aplikasi praktis karena sudah dioptimalkan untuk efisiensi. Pemilihan algoritma pengurutan atau metode yang digunakan bergantung pada ukuran dataset dan kebutuhan efisiensi, dengan mempertimbangkan kompleksitas waktu dan ruang dari algoritma yang bersangkutan. Dengan memahami karakteristik dan tujuan dari berbagai metode pengurutan, programmer dapat memilih pendekatan yang paling sesuai untuk setiap kebutuhan.

## II. GUIDED

### 1. GUIDED 1

#### Studi Case :

Pengurutan nomor rumah kerabat dengan selection sort.

#### Sourcecode

```
package main

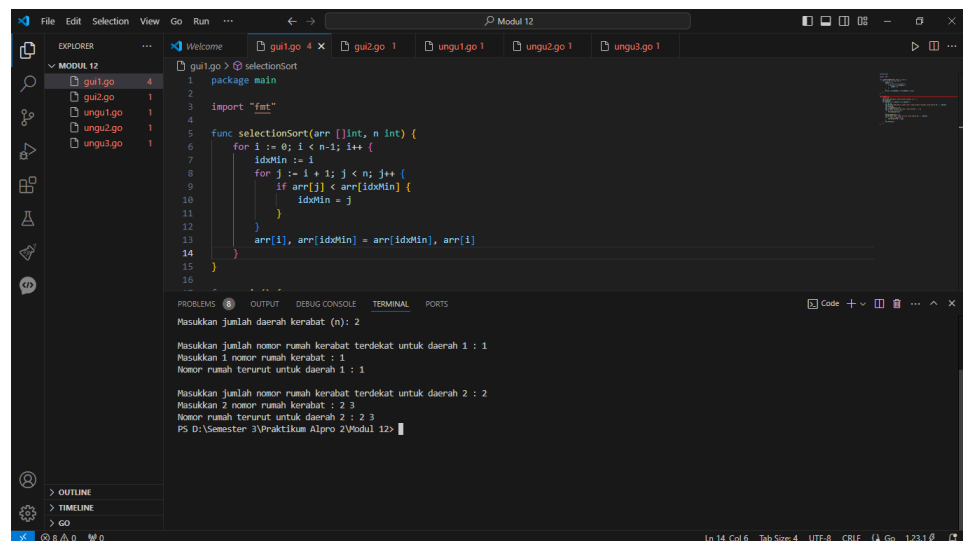
import "fmt"

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

func main() {
    var n int
    fmt.Print("Masukkan jumlah daerah kerabat (n): ")
    fmt.Scan(&n)
    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("\nMasukkan jumlah nomor rumah kerabat
terdekat untuk daerah %d : ", daerah)
        fmt.Scan(&m)
        arr := make([]int, m)
        fmt.Printf("Masukkan %d nomor rumah kerabat : ",
m)

        for i := 0; i < m; i++ {
            fmt.Scan(&arr[i])
        }
        selectionSort(arr, m)
        fmt.Printf("Nomor rumah terurut untuk daerah %d :
", daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}
```

## Screenshoot Output



The screenshot shows a Go IDE with a file explorer on the left containing files like `gui1.go`, `gui2.go`, `ungu1.go`, `ungu2.go`, and `ungu3.go`. The main editor displays a Go program implementing Selection Sort. The code defines a `selectionSort` function that iterates through an array, finds the minimum element, and swaps it with the first element of the current sub-array. The `main` function calls this function. The terminal at the bottom shows the program's execution output, which prompts the user to enter the number of regions and then the house numbers for each region, finally displaying the sorted list of house numbers.

```
1 package main
2
3 import "fmt"
4
5 func selectionSort(arr []int, n int) {
6     for i := 0; i < n-1; i++ {
7         idMin := i
8         for j := i + 1; j < n; j++ {
9             if arr[j] < arr[idMin] {
10                 idMin = j
11             }
12         }
13         arr[i], arr[idMin] = arr[idMin], arr[i]
14     }
15 }
16
17 func main() {
18     n := 0
19     fmt.Println("Masukkan jumlah daerah kerabat (n):")
20     n = 2
21     arr := make([]int, n)
22     for i := 0; i < n; i++ {
23         fmt.Println("Masukkan 1 nomor rumah kerabat terdekat untuk daerah 1 :")
24         arr[i] = 1
25     }
26     selectionSort(arr, n)
27     fmt.Println("Nomor rumah terurut untuk daerah 1 :")
28     for i := 0; i < n; i++ {
29         fmt.Println(arr[i])
30     }
31     for i := 0; i < n; i++ {
32         fmt.Println("Masukkan 2 nomor rumah kerabat :")
33         arr[i] = 2
34     }
35     selectionSort(arr, n)
36     fmt.Println("Nomor rumah terurut untuk daerah 2 :")
37     for i := 0; i < n; i++ {
38         fmt.Println(arr[i])
39     }
40 }
```

Masukkan jumlah daerah kerabat (n): 2  
Masukkan 1 nomor rumah kerabat terdekat untuk daerah 1 : 1  
Nomor rumah terurut untuk daerah 1 : 1  
Masukkan jumlah nomor rumah kerabat terdekat untuk daerah 2 : 2  
Masukkan 2 nomor rumah kerabat : 2 3  
Nomor rumah terurut untuk daerah 2 : 2 3  
PS D:\Semester 3\Praktikum Alpro 2\Modul 12>

## Deskripsi Program

Program ini merupakan sebuah aplikasi sederhana untuk mengurutkan nomor rumah kerabat menggunakan algoritma Selection Sort. Tujuan utamanya adalah membantu pengguna mengelola dan mengurutkan nomor rumah di berbagai daerah dengan cara yang mudah dan sistematis. Proses dimulai dengan pengguna memasukkan jumlah daerah kerabat yang ingin dikelola. Untuk setiap daerah, program meminta pengguna menginputkan jumlah nomor rumah dan kemudian detail nomor rumah tersebut.

Setelah data dimasukkan, program menggunakan fungsi Selection Sort untuk mengurutkan nomor rumah dari yang terkecil hingga terbesar. Algoritma Selection Sort yang diimplementasikan bekerja dengan cara mencari elemen terkecil dalam bagian array yang belum diurutkan, kemudian menukarnya dengan elemen pertama yang belum terurut. Proses ini berulang sampai seluruh array terurut dengan sempurna. Kelebihan program ini terletak pada kemudahan penggunaan dan kesederhanaan algoritma yang digunakan, memungkinkan pengguna dengan mudah mengurutkan nomor rumah untuk beberapa daerah sekaligus. Meskipun demikian, program ini memiliki keterbatasan, seperti hanya mampu menangani bilangan bulat dan tidak memiliki fitur penanganan kesalahan input yang kompleks.

## 2. GUIDED 2

### Studi Case :

Pengurutan dan analisis pola selisih elemen array.

## Sourcecode

```
package main

import (
    "fmt"
)

func insertionSort(arr []int, n int) {
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

func isConstantDifference(arr []int, n int) (bool, int) {
    if n < 2 {
        return true, 0
    }
    difference := arr[1] - arr[0]
    for i := 1; i < n-1; i++ {
        if arr[i+1]-arr[i] != difference {
            return false, 0
        }
    }
    return true, difference
}

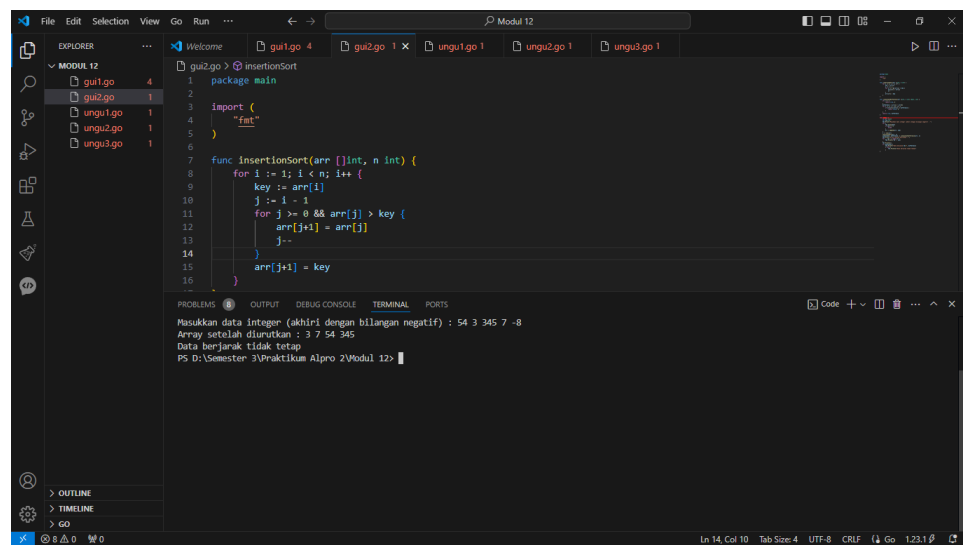
func main() {
    var arr []int
    var num int
    fmt.Print("Masukkan data integer (akhiri dengan bilangan negatif) : ")
    for {
        fmt.Scan(&num)
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }
    n := len(arr)
    insertionSort(arr, n)
    isConstant, difference := isConstantDifference(arr, n)
    fmt.Print("Array setelah diurutkan : ")
    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
}
```

```

        fmt.Println()
    if isConstant {
        fmt.Printf("Data berjarak %d\n", difference)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}

```

## Screenshoot Output



## Deskripsi Program

Program ini adalah sebuah aplikasi untuk mengelola dan menganalisis serangkaian bilangan integer dengan dua fungsi utama: pengurutan dan pengecekan perbedaan konstan. Proses dimulai dengan pengguna memasukkan sejumlah bilangan positif, dengan pemberhentian input ditandai dengan memasukkan bilangan negatif. Program kemudian akan mengurutkan bilangan-bilangan tersebut menggunakan algoritma Insertion Sort dan selanjutnya memeriksa apakah bilangan-bilangan tersebut memiliki perbedaan yang konstan atau tetap.

Algoritma Insertion Sort yang digunakan bekerja dengan cara memindahkan setiap elemen ke posisi yang tepat di antara elemen-elemen sebelumnya yang sudah terurut. Fungsi tambahan 'isConstantDifference' berperan untuk memeriksa apakah selisih antar bilangan dalam array konsisten atau tidak. Setelah pengurutan, program akan menampilkan array yang sudah terurut dan memberikan informasi apakah data memiliki perbedaan tetap beserta nilai perbedaannya. Kelebihan program ini terletak pada kemampuannya menangani input dinamis dan melakukan analisis

sederhana terhadap barisan bilangan, sementara keterbatasannya adalah hanya mampu memproses bilangan bulat positif sebelum input bilangan negatif.

### III. UNGUIDED

#### 1. UNGUIDED 1

##### Studi Case :

Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program kerabat dekat yang akan menampilkan nomor rumah mulai dari normor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil. Format Masukan masih persis sama seperti sebelumnya. Keluaran terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

No	Masukan	Keluaran
1	3 5 2 1 7 9 13 6 189 15 27 39 75 133 3 4 9 1	1 13 12 8 2 15 27 39 75 133 189 8 4 2

##### Sourcecode

```
package main

import (
    "fmt"
    "sort"
)

func main() {
    var t int
    fmt.Print("Masukkan Jumlah Daerah (n) : ")
    fmt.Scan(&t)

    for i := 1; i <= t; i++ {
        var n int
        fmt.Printf("\nMasukkan Jumlah Rumah Daerah %d : ", i)
        fmt.Scan(&n)

        rumah := make([]int, n)
        fmt.Printf("Masukkan %d Nomor Rumah Daerah %d : ", n, i)
```

```

        for j := 0; j < n; j++ {
            fmt.Scan(&rumah[j])
        }

        var ganjil, genap []int
        for _, nomor := range rumah {
            if nomor%2 == 0 {
                genap = append(genap, nomor)
            } else {
                ganjil = append(ganjil, nomor)
            }
        }

        sort.Ints(ganjil)
        sort.Sort(sort.Reverse(sort.IntSlice(genap)))

        fmt.Printf("Nomor Rumah Terurut (Ganjil - Genap)
Daerah %d : ", i)
        for _, nomor := range ganjil {
            fmt.Print(nomor, " ")
        }
        for _, nomor := range genap {
            fmt.Print(nomor, " ")
        }
        fmt.Println()
    }
}

```

## Screenshoot Output

The screenshot shows a Go IDE with a file explorer on the left containing a 'MODUL 12' directory with files 'gu1.go', 'gu2.go', 'ungu1.go', 'ungu2.go', and 'ungu3.go'. The main editor displays the code from 'ungu1.go', which includes a 'package main' declaration, imports for 'fmt' and 'sort', and a 'main' function. The 'main' function prompts the user to enter the number of regions ('Masukkan Jumlah Daerah (n) : '), reads the input, and then iterates from 1 to n, prompting for house numbers in each region and printing them sorted by odd and even numbers. The terminal at the bottom shows the execution output for three regions: Region 1 with 2 houses (1, 2), Region 2 with 4 houses (1, 2, 3, 4), and Region 3 with 3 houses.

```

1 package main
2
3 import (
4     "fmt"
5     "sort"
6 )
7
8 func main() {
9     var t int
10    fmt.Print("Masukkan Jumlah Daerah (n) : ")
11    fmt.Scan(&t)
12
13    for i := 1; i <= t; i++ {
14        var n int
15        fmt.Printf("\nMasukkan Jumlah Rumah Daerah %d : ", i)
16        fmt.Scan(&n)
17    }
18 }

```

```

Masukkan Jumlah Daerah (n) : 3
Masukkan Jumlah Rumah Daerah 1 : 2
Masukkan 2 Nomor Rumah Daerah 1 : 1 2
Nomor Rumah Terurut (Ganjil - Genap) Daerah 1 : 1 2
Masukkan Jumlah Rumah Daerah 2 : 2
Masukkan 2 Nomor Rumah Daerah 2 : 3 4
Nomor Rumah Terurut (Ganjil - Genap) Daerah 2 : 3 4
Masukkan Jumlah Rumah Daerah 3 :

```



## Deskripsi Program

Program ini merupakan sebuah aplikasi untuk mengelola dan mengurutkan nomor rumah di berbagai daerah dengan metode pengurutan khusus yang memisahkan nomor rumah ganjil dan genap. Pengguna diminta untuk memasukkan jumlah daerah yang ingin diproses, dan untuk setiap daerah, akan memasukkan jumlah dan detail nomor rumah yang ada.

Setelah nomor rumah dimasukkan, program melakukan pemisahan antara nomor rumah ganjil dan genap. Nomor rumah ganjil akan diurutkan dari yang terkecil ke terbesar menggunakan fungsi `sort.Ints()`, sementara nomor rumah genap akan diurutkan dari yang terbesar ke terkecil menggunakan `sort.Sort(sort.Reverse(sort.IntSlice(genap)))`. Hasil akhirnya adalah daftar nomor rumah yang telah diurutkan dengan menampilkan terlebih dahulu nomor rumah ganjil, baru kemudian diikuti nomor rumah genap. Kelebihan program ini terletak pada kemampuannya menangani pengurutan nomor rumah secara fleksibel untuk beberapa daerah sekaligus, dengan perlakuan khusus untuk nomor ganjil dan genap.

## 2. UNGUIDED 2

### Studi Case :

Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim bertomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

*"Median adalah nilai tengah dari suatu koleksi data yang sudah terurut. Jika jumlah data genap, maka nilai median adalah rerata dari kedua nilai tengahnya. Pada problem ini, semua data merupakan bilangan bulat positif, dan karenanya rerata nilai tengah dibulatkan ke bawah."*

Buatlah program median yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0. Masukan berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000 data, tidak termasuk bilangan 0. Data 0 merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicari mediannya. Data masukan diakhiri dengan bilangan bulat -5313. Keluaran adalah median yang diminta, satu data per baris.

No	Masukan	Keluaran
1	7 23 11 0 5 19 2 29 3 13 17 0 -5313	11 12

Keterangan : Sampai bilangan 0 yang pertama, data terbaca adalah 7 23 11, setelah tersusun: 7 11 23, maka median saat itu adalah 11. Sampai bilangan 0 yang kedua, data adalah 7 23 11 5 19 2 29 3 13 17, setelah diurutkan: 2 3 5 7 11 13 17 19 23 29, karena ada 10 data, genap, maka median adalah  $(11+13)/2=12$ .

Petunjuk : Untuk setiap data bukan 0 (dan bukan marker -5313541) simpan ke dalam array, Dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metode insertion sort dan ambil mediannya.

### Sourcecode

```
package main

import "fmt"

func urutkanArray(data []int) {
    panjang := len(data)
    for i := 0; i < panjang-1; i++ {
        indeksTerkecil := i
        for j := i + 1; j < panjang; j++ {
            if data[j] < data[indeksTerkecil] {
                indeksTerkecil = j
            }
        }
        data[i], data[indeksTerkecil] = data[indeksTerkecil], data[i]
    }
}

func hitungMedian(data []int) int {
    panjang := len(data)
    if panjang%2 == 0 {
        return (data[panjang/2-1] + data[panjang/2]) / 2
    }
    return data[panjang/2]
}

func main() {
    var angka int
    var dataSementara []int
    var kumpulanGrup [][]int

    fmt.Println("Masukkan data (akhiri dengan -5313)")
    for {
        fmt.Scan(&angka)
        if angka == -5313 {
            break
        }
        if angka == 0 {
```

```

        kumpulanGrup = append(kumpulanGrup, append([]int{},
dataSementara...))
    } else {
        dataSementara = append(dataSementara, angka)
    }
}

for indeks, grup := range kumpulanGrup {
    urutkanArray(grup)
    nilaiMedian := hitungMedian(grup)
    fmt.Printf("Median %d : %d\n", indeks+1, nilaiMedian)
}
}

```

## Screenshoot Output

```

package main

import "fmt"

func urutkanArray(data []int) {
    panjang := len(data)
    for i := 0; i < panjang-1; i++ {
        indeksTerkecil := i
        for j := i + 1; j < panjang; j++ {
            if data[j] < data[indeksTerkecil] {
                indeksTerkecil = j
            }
        }
        data[i], data[indeksTerkecil] = data[indeksTerkecil], data[i]
    }
}

func hitungMedian(data []int) int {
    urutkanArray(data)
    panjang := len(data)
    if panjang % 2 == 0 {
        return (data[panjang/2-1] + data[panjang/2]) / 2
    } else {
        return data[panjang/2]
    }
}

func main() {
    kumpulanGrup := [][]int{}
    dataSementara := []int{}
    for {
        fmt.Print("Masukkan data (akhiri dengan -5313): ")
        angka := 0
        for {
            fmt.Scan(&angka)
            if angka == -5313 {
                kumpulanGrup = append(kumpulanGrup, append([]int{}, dataSementara...))
                dataSementara = []int{}
                break
            }
            dataSementara = append(dataSementara, angka)
        }
    }

    for indeks, grup := range kumpulanGrup {
        urutkanArray(grup)
        nilaiMedian := hitungMedian(grup)
        fmt.Printf("Median %d : %d\n", indeks+1, nilaiMedian)
    }
}

```

```

Masukkan data (akhiri dengan -5313)
7 23 11 0 5 19 2 29 3 13 17 0 -5313
Median 1 : 11
Masukkan data (akhiri dengan -5313)
12 0 -5313
Median 2 : 12
PS D:\Semester 3\Praktikum Alpro 2\Modul 12>

```

## Deskripsi Program

Program ini merupakan sebuah aplikasi untuk mengelola dan menganalisis kumpulan data dengan fokus pada perhitungan median dari setiap kelompok data yang dimasukkan. Proses dimulai dengan pengguna memasukkan sejumlah bilangan, dengan aturan khusus untuk membentuk kelompok dan mengakhiri input. Setiap kali pengguna memasukkan angka 0, program akan menyimpan kumpulan bilangan yang telah dimasukkan sebelumnya sebagai satu kelompok, dan kemudian mempersiapkan kelompok baru.

Program menggunakan dua fungsi utama : `urutkanArray` untuk mengurutkan data menggunakan algoritma Selection Sort, dan `hitungMedian` untuk menghitung nilai median dari setiap kelompok data yang telah diurutkan. Input dihentikan ketika pengguna memasukkan angka

-5313. Setelah semua data dimasukkan dan dikelompokkan, program akan menampilkan median untuk setiap kelompok data. Kelebihan program ini terletak pada fleksibilitasnya dalam mengelola beberapa kelompok data sekaligus dan kemampuannya menghitung median dengan tepat, baik untuk kumpulan data dengan jumlah elemen genap maupun ganjil. Keterbatasan program mencakup penggunaan metode Selection Sort yang kurang efisien untuk kumpulan data besar dan tidak adanya penanganan kesalahan input yang kompleks.

### 3. UNGUIDED 3

#### Studi Case :

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi struct dan array seperti berikut ini :

```
const nMax : integer = 7919
type Buku = <
    id, judul, penulis, penerbit : string
    eksemplar, tahun, rating : integer >

type DaftarBuku = array [ 1..nMax ] of Buku
Pustaka : DaftarBuku
nPustaka: integer
```

Masukan terdiri dari beberapa baris. Baris pertama adalah bilangan bulat N yang menyatakan banyaknya data buku yang ada di dalam perpustakaan. N baris berikutnya, masing-masingnya adalah data buku sesuai dengan atribut atau field pada struct. Baris terakhir adalah bilangan bulat yang menyatakan rating buku yang akan dicari. Keluaran terdiri dari beberapa baris. Baris pertama adalah data buku terfavorit, baris kedua adalah lima judul buku dengan rating tertinggi, selanjutnya baris terakhir adalah data buku yang dicari sesuai rating yang diberikan pada masukan baris terakhir.

Lengkapi subprogram-subprogram dibawah ini, sesuai dengan I.S. dan F.S yang diberikan.

```

procedure DaftarkanBuku(in/out pustaka : DaftarBuku, n : integer)
{I.S. sejumlah n data buku telah siap para piranti masukan
 F.S. n berisi sebuah nilai, dan pustaka berisi sejumlah n data buku}

procedure CetakTerfavorit(in pustaka : DaftarBuku, in n : integer)
{I.S. array pustaka berisi n buah data buku dan belum terurut
 F.S. Tampilan data buku (judul, penulis, penerbit, tahun)
terfavorit, yaitu memiliki rating tertinggi}

procedure UrutBuku( in/out pustaka : DaftarBuku, n : integer )
{I.S. Array pustaka berisi n data buku
 F.S. Array pustaka terurut menurun/mengecil terhadap rating.
Catatan: Gunakan metoda Insertion sort}

procedure Cetak5Terbaru( in pustaka : DaftarBuku, n integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan 5 judul buku dengan rating tertinggi
Catatan: Isi pustaka mungkin saja kurang dari 5}

procedure CariBuku(in pustaka : DaftarBuku, n : integer, r : integer )
{I.S. pustaka berisi n data buku yang sudah terurut menurut rating
 F.S. Laporan salah satu buku (judul, penulis, penerbit, tahun,
eksemplar, rating) dengan rating yang diberikan. Jika tidak ada buku
dengan rating yang ditanyakan, cukup tuliskan "Tidak ada buku dengan
rating seperti itu". Catatan: Gunakan pencarian biner/belah dua.}

```

## Sourcecode

```

package main

import (
    "fmt"
)

type Buku struct {
    ID          string
    Judul       string
    Penulis     string
    Penerbit    string
    Eksemplar   int
    Tahun       int
    Rating      int
}

type DaftarBuku struct {
    Pustaka []Buku
    NPustaka int
}

func TambahBuku(pustaka *DaftarBuku) {
    var jumlah int
    fmt.Print("Masukkan Jumlah Buku : ")
    fmt.Scan(&jumlah)
}

```

```

pustaka.NPustaka = jumlah
pustaka.Pustaka = make([]Buku, jumlah)

for i := 0; i < jumlah; i++ {
    fmt.Printf("\nMasukkan data untuk buku ke-%d\n",
i+1)
    fmt.Print("ID : ")
    fmt.Scan(&pustaka.Pustaka[i].ID)
    fmt.Print("Judul : ")
    fmt.Scan(&pustaka.Pustaka[i].Judul)
    fmt.Print("Penulis : ")
    fmt.Scan(&pustaka.Pustaka[i].Penulis)
    fmt.Print("Penerbit : ")
    fmt.Scan(&pustaka.Pustaka[i].Penerbit)
    fmt.Print("Eksemplar : ")
    fmt.Scan(&pustaka.Pustaka[i].Eksemplar)
    fmt.Print("Tahun : ")
    fmt.Scan(&pustaka.Pustaka[i].Tahun)
    fmt.Print("Rating : ")
    fmt.Scan(&pustaka.Pustaka[i].Rating)
}
}

func CetakBuku(pustaka DaftarBuku) {
    fmt.Println("\nDaftar Buku :")
    fmt.Printf("%-5s %-25s %-20s %-20s %-10s %-10s %-5s\n", "ID", "Judul", "Penulis", "Penerbit", "Eksemplar", "Tahun", "Rating")
    for _, buku := range pustaka.Pustaka {
        fmt.Printf("%-5s %-25s %-20s %-20s %-10d %-10d %-5d\n",
            buku.ID, buku.Judul, buku.Penulis,
buku.Penerbit, buku.Eksemplar, buku.Tahun, buku.Rating)
    }
}

func UrutkanBuku(pustaka *DaftarBuku) {
    n := pustaka.NPustaka
    for i := 1; i < n; i++ {
        key := pustaka.Pustaka[i]
        j := i - 1
        for j >= 0 && pustaka.Pustaka[j].Rating <
key.Rating {
            pustaka.Pustaka[j+1] = pustaka.Pustaka[j]
            j--
        }
        pustaka.Pustaka[j+1] = key
    }
}

func CetakBukuTeratas(pustaka DaftarBuku) {
    fmt.Println("\n5 Buku dengan Rating Tertinggi :")

```

```

        fmt.Printf("%-25s %-5s\n", "Judul", "Rating")
        for i := 0; i < 5 && i < pustaka.NPustaka; i++ {
            fmt.Printf("%-25s %-5d\n",
pustaka.Pustaka[i].Judul, pustaka.Pustaka[i].Rating)
        }
    }

func CariBuku(pustaka DaftarBuku) {
    var rating int
    fmt.Print("\nMasukkan rating yang ingin dicari : ")
    fmt.Scan(&rating)

    kiri, kanan := 0, pustaka.NPustaka-1
    ditemukan := false
    for kiri <= kanan {
        tengah := (kiri + kanan) / 2
        if pustaka.Pustaka[tengah].Rating == rating {
            ditemukan = true
            buku := pustaka.Pustaka[tengah]
            fmt.Printf("\nBuku dengan Rating %d :\n",
rating)
            fmt.Printf("%-5s %-25s %-20s %-20s %-10s %-10s %-5s\n", "ID", "Judul", "Penulis", "Penerbit", "Eksemplar", "Tahun", "Rating")
            fmt.Printf("%-5s %-25s %-20s %-20s %-10d %-10d %-5d\n",
                buku.ID, buku.Judul, buku.Penulis,
buku.Penerbit, buku.Eksemplar, buku.Tahun, buku.Rating)
            break
        } else if pustaka.Pustaka[tengah].Rating < rating
{
            kanan = tengah - 1
        } else {
            kiri = tengah + 1
        }
    }
    if !ditemukan {
        fmt.Println("\nBuku dengan rating tersebut tidak
ditemukan.")
    }
}

func main() {
    var pustaka DaftarBuku

    TambahBuku(&pustaka)

    UrutkanBuku(&pustaka)

    CetakBuku(pustaka)

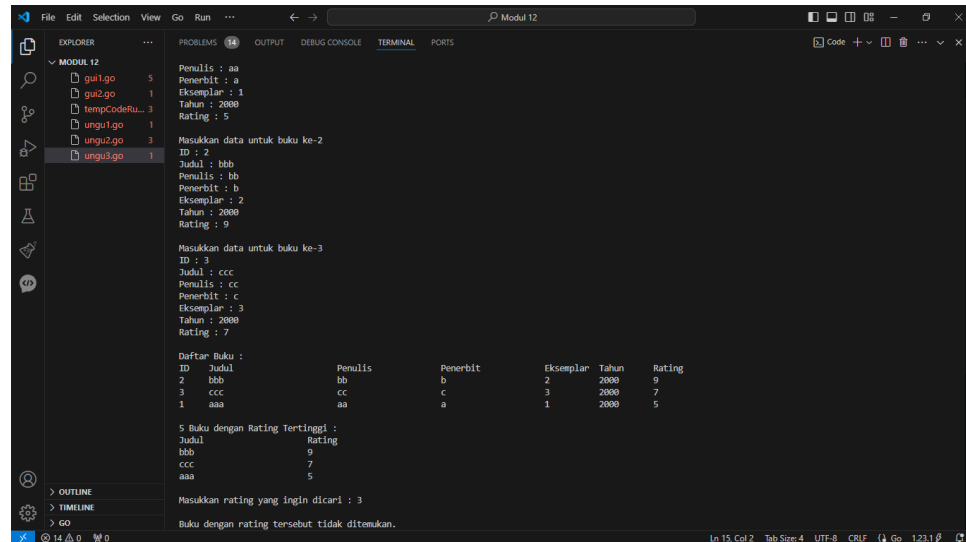
    CetakBukuTeratas(pustaka)

```

```
CariBuku(pustaka)
```

```
}
```

## Screenshoot Output



```
File Edit Selection View Go Run ... Modul 12
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
MODUL 12
  gui1.go 5
  gui2.go 1
  tempCodeRu... 3
  ungu1.go 1
  ungu2.go 3
  ungu3.go 1

Penulis : aa
Penerbit : a
Eksemplar : 1
Tahun : 2000
Rating : 5

Masukkan data untuk buku ke-2
ID : 2
Judul : bbb
Penulis : bb
Penerbit : b
Eksemplar : 2
Tahun : 2000
Rating : 9

Masukkan data untuk buku ke-3
ID : 3
Judul : ccc
Penulis : cc
Penerbit : c
Eksemplar : 3
Tahun : 2000
Rating : 7

Daftar Buku :
ID Judul Penulis Penerbit Eksemplar Tahun Rating
2 bbb bb b 2 2000 9
3 ccc cc c 3 2000 7
1 aaa aa a 1 2000 5

5 Buku dengan Rating Tertinggi :
Judul Rating
bbb 9
ccc 7
aaa 5

Masukkan rating yang ingin dicari : 3
Buku dengan rating tersebut tidak ditemukan.
```

## Deskripsi Program

Program ini adalah sebuah sistem manajemen perpustakaan sederhana yang dirancang untuk mengelola koleksi buku dengan berbagai fitur canggih. Struktur utama program menggunakan tipe data Buku untuk menyimpan informasi detail setiap buku, seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Tipe data DaftarBuku digunakan untuk menyimpan kumpulan buku dan jumlah total buku dalam koleksi.

Program menawarkan empat fungsi utama: menambah buku, menampilkan daftar buku, mengurutkan buku berdasarkan rating, dan mencari buku dengan rating tertentu. Proses dimulai dengan pengguna memasukkan detail sejumlah buku yang ingin disimpan. Setelah pengisian data, program secara otomatis mengurutkan buku dari rating tertinggi ke terendah menggunakan algoritma Insertion Sort, kemudian menampilkan seluruh daftar buku dan lima buku dengan rating tertinggi. Fitur pencarian buku menggunakan algoritma Binary Search untuk menemukan buku dengan rating spesifik yang diinginkan pengguna.

Kelebihan program ini terletak pada kemampuannya mengelola data buku secara terstruktur, menyediakan berbagai operasi pencarian dan pengurutan,



serta antarmuka pengguna yang sederhana namun fungsional. Keterbatasan program mencakup ketergantungan pada input manual dan tidak adanya fitur penyimpanan data permanen seperti basis data atau file eksternal.

#### **IV. KESIMPULAN**

##### **1. Struktur Pengolahan Data**

- Mampu mengimplementasikan berbagai algoritma pengurutan seperti Selection Sort, Insertion Sort
- Mendukung penggunaan tipe data khusus (struct) untuk representasi data kompleks
- Memiliki kemampuan untuk memanipulasi slice dan array secara dinamis

##### **2. Mekanisme Input-Output**

- Menggunakan package fmt untuk interaksi dengan pengguna
- Mendukung input berulang dengan kontrol alur yang fleksibel
- Mampu memproses input dengan berbagai kondisi pemberhentian

##### **3. Algoritma dan Logika Pemrograman**

- Dapat mengimplementasikan algoritma pencarian seperti Binary Search
- Mampu melakukan pemrosesan data dengan algoritma yang efisien
- Mendukung pengurutan dan filtering data sesuai kriteria tertentu

##### **4. Paradigma Pemrograman**

- Menerapkan pendekatan prosedural dengan fungsi-fungsi spesifik
- Mendukung manipulasi data menggunakan pointer
- Memungkinkan pengolahan data kompleks dalam satu kesatuan program

Setiap program memiliki keunikan tersendiri, mulai dari sistem pengurutan nomor rumah, penghitungan median, manajemen buku, hingga pengolahan data numerik, yang menunjukkan keserbagunaan Go dalam menyelesaikan berbagai permasalahan pemrograman.

#### **V. REFERENSI**

[1] Modul XII & XIII Praktikum Algoritma dan Pemrograman 2.

[2] *Learn Go / CodeCademy*. (2022, July 28). Codecademy.  
<https://www.codecademy.com/learn/learn-go>