

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII  
Pengurutan Data**



**Disusun Oleh :**  
**Alfin Adriansyah/ 2311102264**  
**S1IF\_11\_06**

**Dosen Pengampu :**  
**ABEDNEGO DWI SEPTIADI**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2024**

## **I. DASAR TEORI**

### **A. Insertion Sort**

Insertion sort adalah algoritma pengurutan sederhana yang bekerja dengan memasukkan setiap elemen dari daftar yang tidak diurutkan ke posisi yang benar secara berulang di bagian daftar yang diurutkan. Ini seperti mengurutkan kartu remi di tangan Anda. Anda membagi kartu menjadi dua kelompok: kartu yang diurutkan dan kartu yang tidak diurutkan. Kemudian, Anda memilih kartu dari kelompok yang tidak diurutkan dan meletakkannya di tempat yang tepat dalam kelompok yang diurutkan.

### **B. Selection Sort**

Selection Sort adalah algoritma pengurutan berbasis perbandingan. Algoritma ini mengurutkan array dengan cara berulang kali memilih elemen terkecil (atau terbesar) dari bagian yang tidak diurutkan dan menukarnya dengan elemen pertama yang tidak diurutkan. Proses ini berlanjut hingga seluruh array diurutkan.

## II. GUIDED

### 1. Guided 1

#### Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Selection Sort
func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            // Cari elemen terkecil
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
        // Tukar elemen terkecil dengan elemen di
        // posisi i
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah daerah kerabat (n): ")
    fmt.Scan(&n)

    // Proses tiap daerah
    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("\nMasukkan jumlah nomor rumah
kerabat untuk daerah %d: ", daerah)
        fmt.Scan(&m)

        // Membaca nomor rumah untuk daerah ini
        arr := make([]int, m)
        fmt.Printf("Masukkan %d nomor rumah kerabat:
", m)

        for i := 0; i < m; i++ {
            fmt.Scan(&arr[i])
        }

        // Urutkan array dari terkecil ke terbesar
        selectionSort(arr, m)

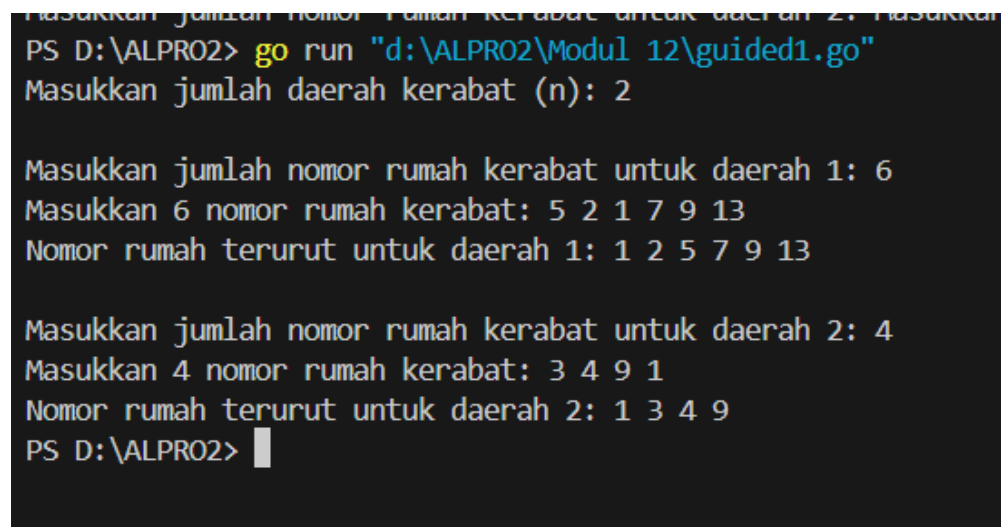
        // Tampilkan hasil
```

```

        fmt.Printf("Nomor rumah terurut untuk daerah
%d: ", daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}

```

## Screenshoot Output



```

PS D:\ALPRO2> go run "d:\ALPRO2\Modul 12\guided1.go"
Masukkan jumlah daerah kerabat (n): 2

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 6
Masukkan 6 nomor rumah kerabat: 5 2 1 7 9 13
Nomor rumah terurut untuk daerah 1: 1 2 5 7 9 13

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 4
Masukkan 4 nomor rumah kerabat: 3 4 9 1
Nomor rumah terurut untuk daerah 2: 1 3 4 9
PS D:\ALPRO2>

```

## Deskripsi Program

Program di atas adalah sebuah aplikasi sederhana yang membantu mengatur dan mengurutkan nomor rumah kerabat di berbagai daerah menggunakan algoritma Selection Sort. Pengguna diminta memasukkan jumlah daerah yang ingin diproses, kemudian untuk setiap daerah, pengguna menginputkan sejumlah nomor rumah kerabat. Program kemudian mengurutkan nomor rumah dari terkecil ke terbesar dengan metode Selection Sort dan menampilkan hasilnya. Fitur utama program ini terletak pada kemampuannya memproses beberapa daerah secara berurutan dan mengurutkan nomor rumah dengan efisien, memudahkan pengguna dalam melihat daftar nomor rumah yang terorganisir.

## 2. Guided2

### Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Insertion Sort
func insertionSort(arr []int, n int) {
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1

        // Geser elemen yang lebih besar dari key ke
        kanan
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

// Fungsi untuk memeriksa apakah selisih elemen array tetap
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

    // Input data hingga bilangan negatif ditemukan
    fmt.Println("Masukkan data integer (akhiri dengan bilangan negatif):")
    for {
```

```

        fmt.Scan(&num)
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    n := len(arr)

    // Urutkan array menggunakan Insertion Sort
    insertionSort(arr, n)

    // Periksa apakah selisih elemen tetap
    isConstant, difference := isConstantDifference(arr,
n)

    // Tampilkan hasil pengurutan
    fmt.Println("Array setelah diurutkan:")
    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
    fmt.Println()

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

```

### Screenshoot Output

```

PS D:\ALPRO2> go run "d:\ALPRO2\Modul 12\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
12 11 12 10 9 8 7 6 12 11 10 -1
Array setelah diurutkan:
6 7 8 9 10 10 11 11 12 12 12
Data berjarak tidak tetap
PS D:\ALPRO2> 

```

### Deskripsi Program

Program ini memungkinkan pengguna memasukkan bilangan bulat, kemudian mengurutkannya menggunakan Insertion Sort. Setelah pengurutan, program memeriksa apakah selisih antar elemen konstan, dan

menampilkan hasilnya, memberikan wawasan tentang pola urutan bilangan yang dimasukkan.

### III. UNGUIDED

1. 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 nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

#### Sourcecode

```
package main

import "fmt"

func main() {
    var n int
    fmt.Print("Masukkan jumlah daerah: ")
    fmt.Scanln(&n)

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

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

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

        for j := 0; j < len(ganjil)-1; j++ {
            minIndex := j
            for k := j + 1; k < len(ganjil); k++ {
                if ganjil[k] < ganjil[minIndex] {
                    minIndex = k
                }
            }
        }
    }
}
```

```

        ganjil[j], ganjil[minIndex] =
ganjil[minIndex], ganjil[j]
    }

    for j := 0; j < len(genap)-1; j++ {
        maxIndex := j
        for k := j + 1; k < len(genap); k++ {
            if genap[k] > genap[maxIndex] {
                maxIndex = k
            }
        }
        genap[j], genap[maxIndex] =
genap[maxIndex], genap[j]
    }

    fmt.Printf("Nomor rumah di daerah %d
(terurut): ", i)
    for _, r := range ganjil {
        fmt.Print(r, " ")
    }
    for _, r := range genap {
        fmt.Print(r, " ")
    }
    fmt.Println()
}
}

```

### Screenshoot Output

```

PS D:\ALPRO2> go run "d:\ALPRO2\Modul 12\unguided1.go"
Masukkan jumlah daerah: 2
Masukkan jumlah rumah di daerah 1: 2
Masukkan nomor rumah di daerah 1 (pisahkan dengan spasi): 12 10
Nomor rumah di daerah 1 (terurut): 12 10
Masukkan jumlah rumah di daerah 2: Masukkan nomor rumah di daerah 2 (pisahkan
PS D:\ALPRO2> █

```

### Deskripsi Program

Program Go ini dirancang untuk membantu Hercules, yang takut menyeberang jalan, dalam mengunjungi rumah kerabatnya secara efisien. Program menerima input berupa nomor rumah di beberapa daerah, lalu mengurutkan nomor-nomor tersebut berdasarkan aturan khusus: nomor ganjil diurutkan menaik dan nomor genap menurun. Dengan demikian, Hercules dapat mengunjungi rumah-rumah tersebut dengan meminimalkan penyeberangan jalan, dimulai dari nomor ganjil terkecil di satu sisi jalan, lalu menyeberang sekali ke sisi lain untuk mengunjungi nomor genap terbesar, dan seterusnya.



2. Kompetisi pemrograman yang baru saja berlalu diikuti oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba 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.

#### Sourcecode

```
package main

import "fmt"

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

func main() {
    var data []int

    fmt.Println("Masukkan angka satu per satu. Gunakan 0
    untuk mencetak median, dan akhiri dengan -5313 untuk
    keluar.")

    for {
        var num int
        fmt.Print("Masukkan angka: ")
        fmt.Scan(&num)

        if num == -5313 {
            fmt.Println("Program selesai.")
            break
        }
    }
}
```

```

        if num == 0 {
            if len(data) == 0 {
                fmt.Println("Tidak ada data untuk
menghitung median.")
                continue
            }

            selectionSort(data)

            length := len(data)
            if length%2 == 0 {
                mid1 := data[length/2-1]
                mid2 := data[length/2]
                fmt.Printf("Median: %d\n",
(mid1+mid2)/2)
            } else {
                fmt.Printf("Median: %d\n",
data[length/2])
            }
            data = nil
        } else {
            data = append(data, num)
        }
    }
}

```

## Screenshoot Output

```

Masukkan angka satu per satu. Gunakan 0 untuk mencetak median, dan akhiri dengan -5313 untuk keluar.
Masukkan angka: 7
Masukkan angka: 23
Masukkan angka: 11
Masukkan angka: 0
Median: 11
Masukkan angka: 5
Masukkan angka: 19
Masukkan angka: 2
Masukkan angka: 29
Masukkan angka: 3
Masukkan angka: 13
Masukkan angka: 17
Masukkan angka: 0
Median: 13
Masukkan angka: -5313
Program selesai.

```

## Deskripsi Program

Program ini memungkinkan pengguna memasukkan angka satu per satu, lalu menghitung dan menampilkan median dari angka-angka tersebut menggunakan algoritma Selection Sort. Pengguna dapat terus memasukkan angka hingga memasukkan -5313 untuk mengakhiri program.

3. Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya. Masukan terdiri dari sekumpulan bilangan bulat yang diakhiri oleh bilangan negatif. Hanya bilangan non negatif saja yang disimpan ke dalam array. Keluaran terdiri dari dua baris. Baris pertama adalah isi dari array setelah dilakukan pengurutan, sedangkan baris kedua adalah status jarak setiap bilangan yang ada di dalam array. "Data berjarak x" atau "data berjarak tidak tetap".

### Sourcecode

```
package main

import "fmt"

const nMax = 7919

type Buku struct {
    id, judul, penulis, penerbit string
    eksemplar, tahun, rating      int
}

type DaftarBuku [nMax]Buku

func main() {
    var pustaka DaftarBuku
    var n, ratingCari int

    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scanln(&n)

    if n <= 0 || n > nMax {
        fmt.Println("Jumlah buku tidak valid!")
        return
    }

    DaftarkanBuku(&pustaka, n)

    CetakTerfavorit(pustaka, n)
    UrutBuku(&pustaka, n)
    Cetak5Terbaru(pustaka, n)

    fmt.Print("Masukkan rating yang ingin dicari: ")
    fmt.Scanln(&ratingCari)
    CariBuku(pustaka, n, ratingCari)
}

func DaftarkanBuku(pustaka *DaftarBuku, n int) {
```

```

        for i := 0; i < n; i++ {
            fmt.Printf("Masukkan data buku ke-%d
(id, judul, penulis, penerbit, eksemplar, tahun,
rating):\n", i+1)
            fmt.Scanln(&pustaka[i].id,
&pustaka[i].judul, &pustaka[i].penulis,
&pustaka[i].penerbit, &pustaka[i].eksemplar,
&pustaka[i].tahun, &pustaka[i].rating)
        }
    }
func CetakTerfavorit(pustaka DaftarBuku, n int) {
    maxRating := -1
    var bukuFavorit Buku
    for i := 0; i < n; i++ {
        if pustaka[i].rating > maxRating {
            maxRating = pustaka[i].rating
            bukuFavorit = pustaka[i]
        }
    }

    fmt.Println("Buku terfavorit:")
    fmt.Println(bukuFavorit.judul,
bukuFavorit.penulis, bukuFavorit.penerbit,
bukuFavorit.tahun)
}

func UrutBuku(pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka[i]
        j := i - 1
        for j >= 0 && pustaka[j].tahun <
key.tahun {
            pustaka[j+1] = pustaka[j]
            j = j - 1
        }
        pustaka[j+1] = key
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("5 Buku terbaru:")
    for i := 0; i < 5 && i < n; i++ {
        fmt.Println(pustaka[i].judul)
    }
}

func CariBuku(pustaka DaftarBuku, n int, r int) {
    for i := 1; i < n; i++ {
        key := pustaka[i]
        j := i - 1
        for j >= 0 && pustaka[j].rating >
key.rating {
            pustaka[j+1] = pustaka[j]
            j = j - 1
        }
    }
}

```

```

        pustaka[j+1] = key
    }

    left := 0
    right := n - 1
    found := false

    for left <= right {
        mid := (left + right) / 2
        if pustaka[mid].rating == r {
            fmt.Println("Buku ditemukan:")
            fmt.Println(pustaka[mid].judul,
                pustaka[mid].penulis, pustaka[mid].penerbit,
                pustaka[mid].tahun, pustaka[mid].eksemplar,
                pustaka[mid].rating)
            found = true
            break
        } else if pustaka[mid].rating < r {
            left = mid + 1
        } else {
            right = mid - 1
        }
    }

    if !found {
        fmt.Println("Tidak ada buku dengan
rating seperti itu")
    }
}

```

## Screenshoot Output

```

Jumlah buku tidak valid!
PS D:\ALPRO2> go run "d:\ALPRO2\Modul 12\unguided3.go"
Masukkan jumlah buku: 1
Masukkan data buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
002 anaksholeh alfin erlangga 10 2020 5
Buku terfavorit:
anaksholeh alfin erlangga 2020
5 Buku terbaru:
anaksholeh
Masukkan rating yang ingin dicari: 5
Buku ditemukan:
anaksholeh alfin erlangga 2020 10 5
PS D:\ALPRO2> 

```

## **Deskripsi Program**

Program ini adalah sebuah aplikasi sederhana untuk mengelola data buku di perpustakaan. Pengguna dapat memasukkan informasi buku, termasuk judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Program dapat menampilkan buku dengan rating tertinggi, mencetak 5 buku terbaru, dan mencari buku berdasarkan rating tertentu menggunakan algoritma pencarian biner. Fitur-fitur ini memudahkan pengelolaan koleksi buku di perpustakaan serta membantu pengguna menemukan buku terfavorit dan terbaru dengan cepat.