

**LAPORAN PRAKTIKUM  
ALGORITMA PEMROGRAMAN 2**

**MODUL XIII  
PENGURUTAN DATA**



Oleh:

**RIZKULLOH ALPRIYANSAH**

**2311102142**

**IF-11-08**

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

## I. DASAR TEORI

Dasar teori dari program ini berlandaskan pada prinsip-prinsip **struktur data**, **algoritma pengurutan**, dan **algoritma pencarian**, yang digunakan untuk mengelola informasi daftar buku secara terstruktur dan efisien. Pada dasarnya, data buku direpresentasikan dalam bentuk **struct**, yang memungkinkan penyimpanan beberapa atribut (judul, penulis, penerbit, tahun, dan rating) dalam satu entitas. Struct ini menjadi elemen fundamental dalam pengelolaan data terstruktur karena mendukung modularitas dan keterbacaan program. Daftar buku dikelola menggunakan **slice**, yaitu struktur data dinamis dalam Go yang mendukung penambahan data secara fleksibel tanpa perlu menentukan ukuran awal, sehingga sesuai untuk aplikasi yang melibatkan jumlah data yang tidak pasti.

Untuk mengurutkan buku berdasarkan rating dalam urutan **descending**, digunakan fungsi bawaan Go, `sort.Slice`, yang merupakan implementasi **algoritma pengurutan berbasis komparator**. Algoritma ini memungkinkan pengurutan dilakukan dengan membandingkan elemen-elemen dalam slice berdasarkan aturan tertentu, yaitu rating. Proses pengurutan ini memiliki kompleksitas waktu rata-rata  $O(n \log n)$ , sehingga cukup efisien untuk pengelolaan data dalam jumlah besar.

Pencarian buku berdasarkan rating dilakukan menggunakan **Linear Search**, yaitu algoritma pencarian sederhana dengan kompleksitas waktu  $O(n)$ . Dalam algoritma ini, setiap elemen dalam slice diperiksa satu per satu hingga ditemukan elemen yang sesuai dengan kriteria pencarian, atau hingga seluruh elemen diperiksa. Meskipun linear search kurang efisien dibandingkan algoritma pencarian seperti binary search, algoritma ini tetap relevan dalam program karena datanya tidak dijamin terurut sebelum pencarian.

Selain itu, program juga menerapkan konsep **iterasi dan seleksi** untuk menampilkan informasi, seperti mencari buku dengan rating tertinggi dan mencetak daftar buku. Kombinasi dari struktur data slice, algoritma pengurutan, pencarian, dan pemrosesan data berbasis iterasi mencerminkan pendekatan berbasis prinsip dasar pemrograman terstruktur, yang menjadikan program ini fleksibel, modular, dan mudah digunakan.

## I. GUIDED

### Guided1

#### Source Code

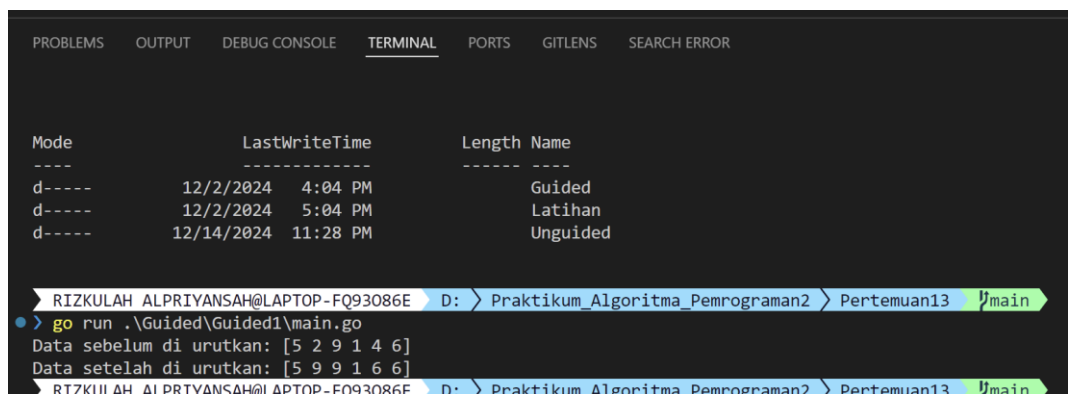
```
package main

import "fmt"

//Insertionsort untuk mengurutkan array secara discanding
func InsertionSort(arr []int){
    n := len(arr)
    for i := 1; i < n; i++ {
        temp := arr[i]
        j := i - 1
        //Geser elemen yang lebih kecil ke kanan
        for j > 0 && temp > arr[j-1] {
            arr[j] = arr[j-1]
            j--
        }
        arr[j+1] = temp
    }
}

func main(){
    data := []int{5, 2, 9, 1, 4, 6}
    fmt.Println("Data sebelum di urutkan:", data)
    InsertionSort(data)
    fmt.Println("Data setelah di urutkan:", data)
}
```

#### ScreenShot Output



The screenshot shows a code editor interface with a file explorer on the left and a terminal on the right. The file explorer displays a list of files with columns for Mode, LastWriteTime, Length, and Name. The terminal shows the execution of a Go program that sorts an array using Insertion Sort.

Mode	LastWriteTime	Length	Name
d-----	12/2/2024 4:04 PM		Guided
d-----	12/2/2024 5:04 PM		Latihan
d-----	12/14/2024 11:28 PM		Unguided

```
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93O86E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan13 > main
> go run .\Guided\Guided1\main.go
Data sebelum di urutkan: [5 2 9 1 4 6]
Data setelah di urutkan: [5 9 9 1 6 6]
RIZKULAH ALPRIYANSAH@LAPTOP-FQ93O86E D: > Praktikum_Algoritma_Pemrograman2 > Pertemuan13 > main
```

#### Deskripsi

Program di atas mengimplementasikan algoritma **Insertion Sort** untuk mengurutkan elemen dalam sebuah array secara **descending** (dari besar ke kecil). Fungsi InsertionSort mengambil array sebagai input, kemudian memproses elemen dari indeks kedua hingga akhir, membandingkannya dengan elemen sebelumnya, dan menggeser elemen yang lebih kecil ke kanan untuk memberikan ruang bagi elemen yang lebih besar. Pada fungsi main, sebuah array contoh diurutkan menggunakan fungsi ini, dan hasilnya ditampilkan sebelum dan setelah pengurutan.

## Guided2

### Source Code

```
package main

import "fmt"

// Struct mahasiswa

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

// InsertionSort untuk mengurutkan array mahasiswa berdasarkan nama
(descending)

func insertionSort(arr []mahasiswa) {
    n := len(arr)
    for i := 1; i < n; i++ {
        temp := arr[i]
        j := i
        // Geser elemen yang lebih kecil (berdasarkan nama) ke kanan
        for j > 0 && temp.nama > arr[j-1].nama {
            arr[j] = arr[j-1]
            j--
        }
        arr[j] = temp
    }
}

func main() {

    data := []mahasiswa{
        {"Kyla", "101", "IF1", "Teknik Informatika", 4.0},
```

```

        {"Azzahra", "102", "IF2", "Teknik Informatika", 3.8},
        {"Kinan", "103", "IF1", "Teknik Informatika", 3.9},
        {"Ara", "104", "IF2", "Teknik Informatika", 3.4},
    }

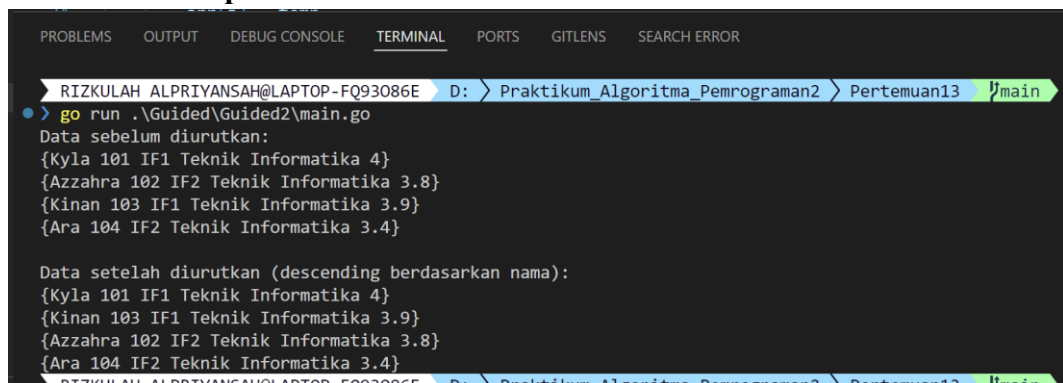
    fmt.Println("Data sebelum diurutkan:")
    for _, m := range data {
        fmt.Println(m)
    }

    insertionSort(data)

    fmt.Println("\nData setelah diurutkan (descending berdasarkan nama):")
    for _, m := range data {
        fmt.Println(m)
    }
}

```

## ScreenShot Output



```

RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D:\Praktikum_Algoritma_Pemrograman2\Pertemuan13 main
> go run .\Guided\Guided2\main.go
Data sebelum diurutkan:
{Kyla 101 IF1 Teknik Informatika 4}
{Azzahra 102 IF2 Teknik Informatika 3.8}
{Kinan 103 IF1 Teknik Informatika 3.9}
{Ara 104 IF2 Teknik Informatika 3.4}

Data setelah diurutkan (descending berdasarkan nama):
{Kyla 101 IF1 Teknik Informatika 4}
{Kinan 103 IF1 Teknik Informatika 3.9}
{Azzahra 102 IF2 Teknik Informatika 3.8}
{Ara 104 IF2 Teknik Informatika 3.4}

```

## Deskripsi

Program di atas menggunakan algoritma **Insertion Sort** untuk mengurutkan data mahasiswa berdasarkan nama secara **descending** (urut dari Z ke A). Data mahasiswa direpresentasikan sebagai struktur (struct) yang menyimpan informasi seperti nama, NIM, kelas, jurusan, dan IPK. Fungsi `insertionSort` bekerja dengan membandingkan nama mahasiswa dan menggeser elemen-elemen ke posisi yang sesuai. Pada fungsi `main`, data mahasiswa diinisialisasi, ditampilkan sebelum dan sesudah diurutkan, sehingga hasil urutannya dapat terlihat dengan jelas.

## II. UNGUIDED

### Unguided1

#### Source Code

```
package main

import (
    "fmt"
    "sort"
)

func main() {
    var data []int

    fmt.Println("Masukkan")
    for {
        var num int
        fmt.Scan(&num)
        if num < 0 {
            break
        }
        data = append(data, num)
    }

    if len(data) == 0 {
        fmt.Println("Tidak ada data yang dimasukkan.")
        return
    }

    sort.Ints(data)

    fmt.Println("Keluaran")
    for _, val := range data {
        fmt.Printf("%d ", val)
    }
    fmt.Println()

    status := checkDistance(data)
    fmt.Println(status)
}

func checkDistance(arr []int) string {
    if len(arr) < 2 {
        return "Data berjarak tidak tetap"
    }
}
```

```

distance := arr[1] - arr[0]
for i := 1; i < len(arr)-1; i++ {
    if arr[i+1]-arr[i] != distance {
        return "Data berjarak tidak tetap"
    }
}

return fmt.Sprintf("Data berjarak %d", distance)
}

```

## Screenshot

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GIT LENS  SEARCH ERROR

RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum Algoritma Pemrograman2 > Pertemuan13 > main
> go run .\Unguided\Unguided1\main.go
Masukkan
1 12 36 6 24 30 18 -1
Keluaran
1 6 12 18 24 30 36
Data berjarak tidak tetap

RIZKULAH ALPRIYANSAH@LAPTOP-FQ93086E D: > Praktikum Algoritma Pemrograman2 > Pertemuan13 > main
> go run .\Unguided\Unguided1\main.go
Masukkan
5 10 15 20 25 30 35 40 45 50 -11
Keluaran
5 10 15 20 25 30 35 40 45 50
Data berjarak 5

```

## Deskripsi

Program di atas memungkinkan pengguna untuk memasukkan sejumlah bilangan bulat yang akan diurutkan secara **ascending** (menaik) menggunakan fungsi bawaan `sort.Ints()`. Pengguna dapat memasukkan bilangan positif satu per satu, dan masukan akan dihentikan saat angka negatif dimasukkan. Setelah data diurutkan, program menampilkan data yang telah diurutkan dan memeriksa apakah jarak antar elemen dalam data konsisten (tetap). Fungsi `checkDistance` memeriksa jarak antar elemen berturut-turut dalam array, dan memberikan keluaran berupa pesan apakah jaraknya tetap atau tidak tetap. Jika jaraknya tetap, program juga mencantumkan nilai jarak tersebut.

## Unguided2

### Source Code

```
package main

import (
    "fmt"
    "sort"
)

const mMax = 7919

type Buku struct {
    judul   string
    penulis string
    penerbit string
    tahun   int
    rating  float64
}

type DaftarBuku struct {
    data []Buku
    nBuku int
}

func main() {
    var pustaka DaftarBuku
    var cariRating float64

    DaftarkanBuku(&pustaka)

    CetakTerfavorit(&pustaka)

    UrutBuku(&pustaka)

    fmt.Println("\nDaftar buku setelah diurutkan berdasarkan rating:")
    CetakSemuaBuku(&pustaka)

    fmt.Println("\nMasukkan rating buku yang ingin dicari:")
    fmt.Scan(&cariRating)
    CariBuku(&pustaka, cariRating)
}

func DaftarkanBuku(pustaka *DaftarBuku) {
    var n int
    fmt.Print("Masukkan jumlah buku: ")
```



```

fmt.Scan(&n)

for i := 0; i < n; i++ {
    var buku Buku
    fmt.Printf("\nMasukkan data buku ke-%d:\n", i+1)
    fmt.Print("Judul: ")
    fmt.Scan(&buku.judul)
    fmt.Print("Penulis: ")
    fmt.Scan(&buku.penulis)
    fmt.Print("Penerbit: ")
    fmt.Scan(&buku.penerbit)
    fmt.Print("Tahun: ")
    fmt.Scan(&buku.tahun)
    fmt.Print("Rating: ")
    fmt.Scan(&buku.rating)

    pustaka.data = append(pustaka.data, buku)
}
pustaka.nBuku = n
}

func CetakTerfavorit(pustaka *DaftarBuku) {
    if pustaka.nBuku == 0 {
        fmt.Println("Tidak ada data buku.")
        return
    }

    tertinggi := pustaka.data[0]
    for _, buku := range pustaka.data {
        if buku.rating > tertinggi.rating {
            tertinggi = buku
        }
    }

    fmt.Println("\nBuku dengan rating tertinggi:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %.2f\n",
        tertinggi.judul, tertinggi.penulis,
        tertinggi.penerbit, tertinggi.tahun, tertinggi.rating)
}

func UrutBuku(pustaka *DaftarBuku) {
    sort.Slice(pustaka.data, func(i, j int) bool {
        return pustaka.data[i].rating >
pustaka.data[j].rating
    })
}

```

```

}

func CetakSemuaBuku(pustaka *DaftarBuku) {
    if pustaka.nBuku == 0 {
        fmt.Println("Tidak ada data buku.")
        return
    }

    for _, buku := range pustaka.data {
        fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %.2f\n",
            buku.judul, buku.penulis, buku.penerbit,
            buku.tahun, buku.rating)
    }
}

func CariBuku(pustaka *DaftarBuku, rating float64) {
    for _, buku := range pustaka.data {
        if buku.rating == rating {
            fmt.Println("\nBuku yang ditemukan:")
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %.2f\n",
                buku.judul, buku.penulis, buku.penerbit,
                buku.tahun, buku.rating)
            return
        }
    }

    fmt.Println("\nBuku dengan rating tersebut tidak ditemukan.")
}

```

## Screenshot

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS  SEARCH ERROR

Masukkan data buku ke-1:
Judul: bunga
Penulis: NotFirst
Penerbit: NotFirst
Tahun: 2020
Rating: 3.5

Masukkan data buku ke-2:
Judul: cinta
Penulis: NotFirst
Penerbit: NotFirst
Tahun: 2021
Rating: 3.7

Masukkan data buku ke-3:
Judul: kan
Penulis: NotFirst
Penerbit: NotFirst
Tahun: 2023
Rating: 3.9

Masukkan data buku ke-4:
Judul: abadi
Penulis: NotFirst
Penerbit: NotFirst
Tahun: 2024
Rating: 4.0

Buku dengan rating tertinggi:
Judul: abadi, Penulis: NotFirst, Penerbit: NotFirst, Tahun: 2024, Rating: 4.00

Daftar buku setelah diurutkan berdasarkan rating:
Judul: abadi, Penulis: NotFirst, Penerbit: NotFirst, Tahun: 2024, Rating: 4.00
Judul: kan, Penulis: NotFirst, Penerbit: NotFirst, Tahun: 2023, Rating: 3.90
Judul: cinta, Penulis: NotFirst, Penerbit: NotFirst, Tahun: 2021, Rating: 3.70
Judul: bunga, Penulis: NotFirst, Penerbit: NotFirst, Tahun: 2020, Rating: 3.50

Masukkan rating buku yang ingin dicari:
4.0

Buku yang ditemukan:
Judul: abadi, Penulis: NotFirst, Penerbit: NotFirst, Tahun: 2024, Rating: 4.00
```

## Deskripsi

Program ini adalah sistem manajemen daftar buku yang memungkinkan pengguna untuk mendata buku, menampilkan buku dengan rating tertinggi, mengurutkan daftar buku berdasarkan rating secara **descending**, mencetak semua data buku, dan mencari buku berdasarkan rating tertentu. Setiap buku direpresentasikan sebagai struct yang mencakup informasi seperti judul, penulis, penerbit, tahun, dan rating. Pengguna dapat memasukkan data buku melalui fungsi

DaftarkanBuku, melihat buku terbaik menggunakan  
CetakTerfavorit, mengurutkan buku dengan UrutBuku,  
mencetak daftar menggunakan CetakSemuaBuku, dan mencari  
buku tertentu dengan CariBuku. Program ini mempermudah  
pengelolaan dan pencarian buku dalam sebuah koleksi.

### **III. DAFTAR PUSTAKA**

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