

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12 & 13  
PENGURUTAN DATA**



**Disusun Oleh :**

**Afif Rijal Azzami / 2311102235**

**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**

### **A. Selection Sort**

Pengurutan secara seleksi ini idenya adalah mencari nilai ekstrim pada sekumpulan data, kemudian meletakkan pada posisi yang seharusnya. Pada penjelasan berikut ini data akan diurut membesar (ascending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan", cara kerjanya yaitu;

1. Cari nilai terkecil di dalam rentang data tersisa
2. Pindahkan/tukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
3. Ulangi proses ini sampai tersisa hanya satu data saja.

Algoritma ini dikenal juga dengan nama Selection Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau swap.

### **B. Insertion Sort**

Pengurutan secara insertion ini idenya adalah menyisipkan suatu nilai pada posisi yang seharusnya. Berbeda dengan pengurutan seleksi, yang mana pada pengurutan ini tidak dilakukan pencarian nilai ekstrim terlebih dahulu, cukup memilih suatu nilai tertentu kemudian mencari posisinya secara sequential search. Pada penjelasan berikut ini data akan diurut mengecil (descending), dan data dengan indeks kecil ada di "kiri" dan indeks besar ada di "kanan", cara kerjanya yaitu;

1. Geser data yang sudah terurut tersebut (ke kanan), sehingga ada satu ruang kosong untuk memasukkan data data yang yang belum belum terurut ke k dalam data yang sudah terurut dan tetap menjaga keterurutan.
2. Ulangi proses tersebut untuk setiap data yang belum terurut terhadap rangkaian data yang sudah terurut.

Algoritma ini dikenal juga dengan nama Insertion Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan

## II. UNGUIDED 1

### Sourcecode

```
package main

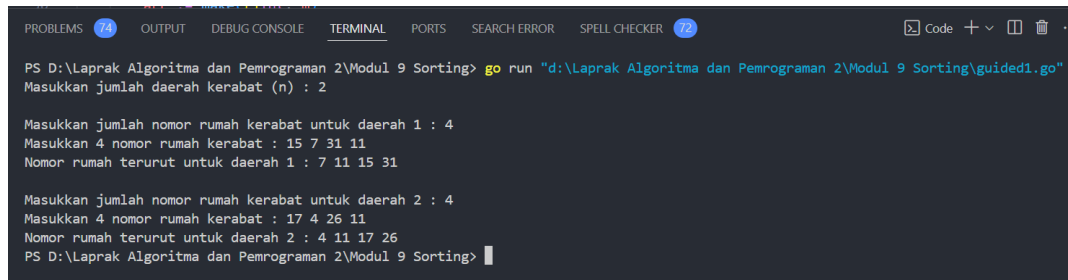
import (
    "fmt"
)

// 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++ {
            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 i := 0; i < n; i++ {  
  
    var m int  
  
    fmt.Printf("\nMasukkan jumlah nomor rumah kerabat untuk  
daerah %d : ", i+1)  
  
    fmt.Scan(&m)  
  
    arr := make([]int, m)  
  
    fmt.Printf("Masukkan %d nomor rumah kerabat : ", m)  
    for j := 0; j < m; j++ {  
        fmt.Scan(&arr[j])  
    }  
  
    selectionSort(arr, m)  
  
    // Tampilkan nomor rumah terurut  
    fmt.Printf("Nomor rumah terurut untuk daerah %d : ", i+1)  
    for _, num := range arr {  
        fmt.Printf("%d ", num)  
    }  
  
    fmt.Println()  
}  
}
```

## Screenshoot Output



```
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting\guided1.go"
Masukkan jumlah daerah kerabat (n) : 2

Masukkan jumlah nomor rumah kerabat untuk daerah 1 : 4
Masukkan 4 nomor rumah kerabat : 15 7 31 11
Nomor rumah terurut untuk daerah 1 : 7 11 15 31

Masukkan jumlah nomor rumah kerabat untuk daerah 2 : 4
Masukkan 4 nomor rumah kerabat : 17 4 26 11
Nomor rumah terurut untuk daerah 2 : 4 11 17 26
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting>
```

## Soal Studi Case

Program untuk mengurutkan nomer rumah dengan algoritma selection sort.

### Deskripsi Program

Sintaks tersebut adalah program untuk mengurutkan nomer rumah menggunakan algoritma selection sort. Untuk pengurutan array kita membuat fungsi selectionSort, cara kerja fungsi tersebut adalah looping yang diinisiasi dengan var i untuk menentukan posisi elemen saat ini (i) yang akan diisi dengan nilai minimum. Selanjutnya looping yang diinisiasi dengan var j untuk mencari elemen terkecil di sub-array yang dimulai dari j, selanjutnya jika ditemukan elemen yang lebih kecil dari arr[idxMin], maka indeks idxMin diperbarui, dan selanjutnya menukar elemen terkecil yang ditemukan dengan elemen di indeks i. Pada func main untuk menginput nomer rumah dan menampilkan pengurutan dengan memanggil func selectionSort.

### III. GUIDED 2

```
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)

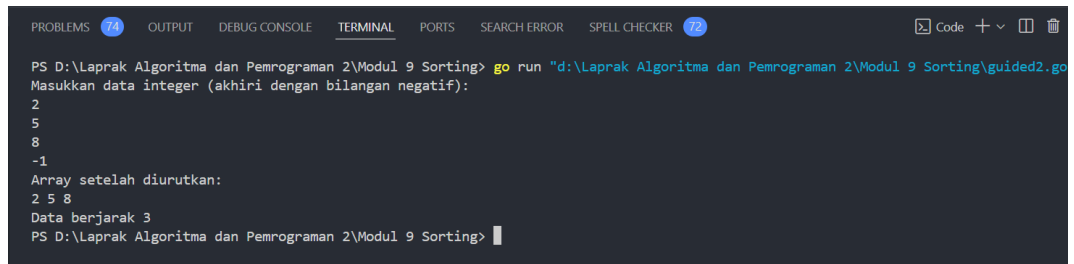
// Tampilkan hasil pengurutan
fmt.Println("Array setelah diurutkan:")
for _, val := range arr {
    fmt.Printf("%d ", val)
}

fmt.Println()

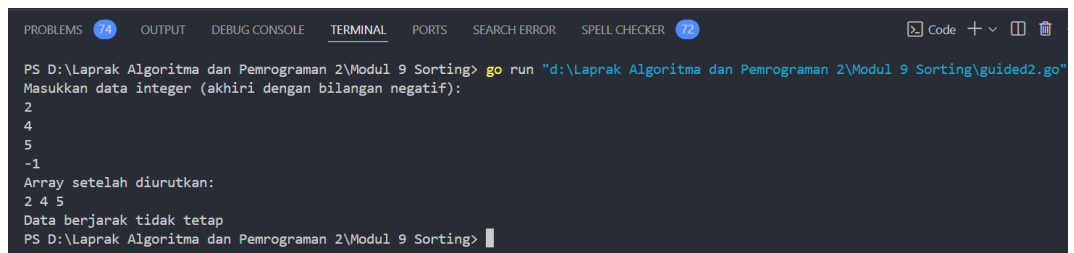
// Tampilkan status jarak
if isConstant {
    fmt.Printf("Data berjarak %d\n", difference)
} else {
    fmt.Println("Data berjarak tidak tetap")
}
}
```



## Screenshoot Output



```
PROBLEMS 74 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR SPELL CHECKER 72
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
2
5
8
-1
Array setelah diurutkan:
2 5 8
Data berjarak 3
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting>
```



```
PROBLEMS 74 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR SPELL CHECKER 72
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting\guided2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
2
4
5
-1
Array setelah diurutkan:
2 4 5
Data berjarak tidak tetap
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting>
```

## Soal Studi Case

Mengecek suatu deret angka apakah berjarak tetap atau gak.

### Deskripsi Program

Sintaks tersebut adalah program untuk melakukan pengecekan apakah sebuah deret angka berjarak tetap atau tidak menggunakan algoritma insertion sort. Pertama kita membuat fungsi yang dinamai `insertionSort` untuk mengurutkan deret angka yang diinput oleh user. Kemudian kita membuat fungsi yang dinamai `isConstantDifference` yang berguna untuk mengecek apakah deret angka berjarak tetap apa gak. Cara kerjanya yaitu jika array memiliki kurang dari 2 maka dianggap memiliki selisih tetap dengan nilai selisih 0, kemudian untuk menghitung selisih menggunakan var `difference` dengan rumus `arr[1] - arr[0]`, Jika ada selisih yang berbeda, fungsi mengembalikan `false`, dan jika semua selisih sama, fungsi mengembalikan `true`, `difference`. Pada func `main` terdapat sintaks agar user dapat menginputkan deret angka, loop akan berhenti jika user menginput angka negatif, dan fungsi yang sudah dibuat, dipanggil di func `main` untuk mengurutkan data, dan melakukan pengecekan selisih pada deret.

#### IV. UNGUIDED 1

```
package main

import "fmt"

// Fungsi untuk mengurutkan array dengan selection sort
func selectionSort235(arr235 []int, asc235 bool) {

    n235 := len(arr235)

    for i235 := 0; i235 < n235-1; i235++ {

        idx235 := i235

        for j235 := i235 + 1; j235 < n235; j235++ {

            if (asc235 && arr235[j235] < arr235[idx235]) ||
(!asc235 && arr235[j235] > arr235[idx235]) {

                idx235 = j235

            }

        }

        arr235[i235], arr235[idx235] = arr235[idx235],
arr235[i235]

    }

}

func main() {

    var n235 int

    // Input jumlah daerah kerabat

    fmt.Print("Masukkan jumlah daerah kerabat (n) : ")

    fmt.Scan(&n235)

    for i235 := 0; i235 < n235; i235++ {

        var m235 int
```

```

// Input jumlah rumah kerabat untuk setiap daerah

    fmt.Printf("\nMasukkan jumlah rumah kerabat di daerah ke-
%d (m): ", i235+1)

    fmt.Scan(&m235)

    arr235 := make([]int, m235)

    // Input nomor rumah kerabat

    fmt.Printf("Masukkan %d nomor rumah kerabat: ", m235)

    for j235 := 0; j235 < m235; j235++ {

        fmt.Scan(&arr235[j235])

    }

    // untuk memisahkan nomor ganjil dan genap

    var odd235, even235 []int

    for _, num235 := range arr235 {

        if num235%2 == 0 {

            even235 = append(even235, num235)

        } else {

            odd235 = append(odd235, num235)

        }

    }

    // untuk mengurutkan ganjil dengan menaik

    selectionSort235(odd235, true)

    // untuk mengurutkan genap dengan menurun

    selectionSort235(even235, false)

    // Output nomor rumah yang terurut

    fmt.Printf("\nNomor rumah terurut untuk daerah ke-%d:\n",
i235+1)

```

```
// Tampilkan nomor ganjil

    for _, num235 := range odd235 {

        fmt.Printf("%d ", num235)

    }

// Tampilkan nomor genap

    for _, num235 := range even235 {

        fmt.Printf("%d ", num235)

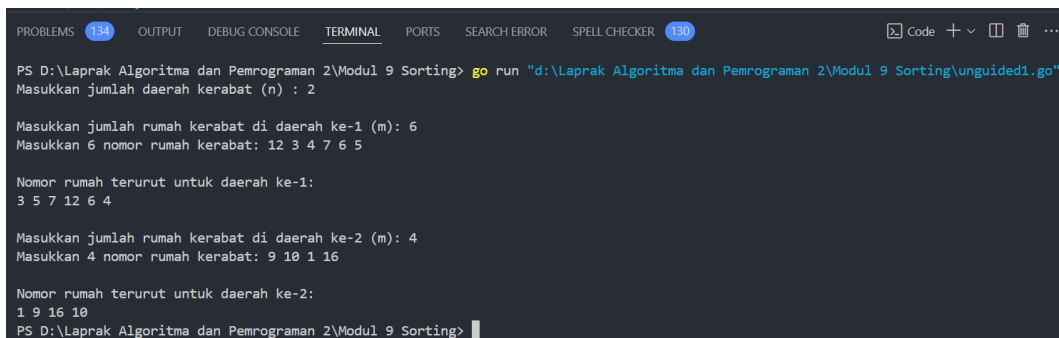
    }

    fmt.Println()

}

}
```

## Screenshoot Output



```
PROBLEMS 134 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR SPELL CHECKER 130
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting\unguided1.go"
Masukkan jumlah daerah kerabat (n) : 2

Masukkan jumlah rumah kerabat di daerah ke-1 (m): 6
Masukkan 6 nomor rumah kerabat: 12 3 4 7 6 5

Nomor rumah terurut untuk daerah ke-1:
3 5 7 12 6 4

Masukkan jumlah rumah kerabat di daerah ke-2 (m): 4
Masukkan 4 nomor rumah kerabat: 9 10 1 16

Nomor rumah terurut untuk daerah ke-2:
1 9 16 10
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting> |
```

## Soal Studi Case

Program untuk mengurutkan nomer rumah dengan algoritma selection sort dan membedakan nomer rumah ganjil diurut dengan acending dan genap dengan decending.

## Deskripsi Program

Sintaks tersebut adalah program untuk mengurutkan nomer rumah dengan algoritma selection sort dan membedakan nomer rumah ganjil diurut dengan acending dan genap dengan decending. Pertama kita membuat fungsi yang dinamai selectionSort digunakan untuk mengurutkan array menggunakan

algoritma selection Sort. Urutan dapat berupa ascending atau descending tergantung pada parameter asc. Jika asc adalah true, array diurutkan secara menaik, dan jika asc adalah false, array diurutkan secara menurun. Cara kerjanya adalah indeks i dianggap sebagai posisi tempat elemen terkecil atau terbesar berikutnya akan ditempatkan. Selanjutnya membandingkan elemen di indeks j dengan elemen di indeks idx. Jika elemen lebih kecil ditemukan maka pengurutan ascending, dan jika elemen lebih besar ditemukan maka pengurutan descending. Dan selanjutnya Menukar elemen di indeks i dengan elemen terkecil/terbesar yang ditemukan. Pada func main terdapat algoritma untuk memisahkan nomer rumah yang ganjil dan yang genap. Cara kerjanya adalah pertama kita mendeklarasikan array yang dimanai odd dan even, array odd untuk menyimpan nilai ganjil dan array even untuk menyimpan nilai genap. Jika hasil modulus bagi dua = 0, maka nilai tersebut ditambahkan ke array even, dan elsenya adalah kebalikannya.

## V. UNGUIDED 2

```
package main

import "fmt"

// Fungsi untuk mengurutkan array dengan selection sort
func selectionSort(arr []int) {

    n := len(arr)

    for i := 0; i < n-1; i++ {

        // Cari elemen terkecil di subarray yang belum terurut
        minIdx := i

        for j := i + 1; j < n; j++ {

            if arr[j] < arr[minIdx] {

                minIdx = j

            }

        }

        // Tukar elemen terkecil dengan elemen pertama subarray
        arr[i], arr[minIdx] = arr[minIdx], arr[i]

    }

}

// Fungsi untuk menghitung nilai median dari array yang sudah
// diurutkan
func cariMedian(arr []int) int {

    n := len(arr)

    if n%2 == 1 {

        // Jika jumlah elemen bernilai ganjil, maka ambil elemen
        // tengah
        return arr[n/2]

    }
```

```
// Jika jumlah elemen bernilai genap, maka ambil rata-rata dari
dua elemen tengah dan dibulatkan

    return (arr[(n/2)-1] + arr[n/2]) / 2
}

func main() {

    var data []int

    fmt.Print("Masukkan angka-angka: ")

    for {

        var num int

        _, err := fmt.Scan(&num)

        if num == -5313 {

            break

        }

        if err != nil {

            break

        }

        if num == 0 {

            // untuk mengurutkan data dan mencetak median

            selectionSort(data)

        }

    }

}
```

```

fmt.Println(cariMedian(data))

    } else {

        // menambahkan bilangan ke array

        data = append(data, num)

    }

}

}

```

## Screenshoot Output

The screenshots show the execution of a Go program in a terminal window. The first screenshot shows the program running with the input "Masukkan angka-angka: 14 3 17 6 1 0 -5313" and outputting "6". The second screenshot shows the program running with the input "Masukkan angka-angka: 9 15 3 18 0 -5313" and outputting "12".

## Soal Studi Case

Mencari nilai median pada deret bilangan.

## Deskripsi Program

Sintaks tersebut adalah program untuk mencari nilai median dari suatu deret angka, jika jumlah deret ganjil mediannya diambil pada nilai tengahnya, jika deret berjumlah genap maka kedua bilangan yang menjadi nilai tengah ditambah dan dibagi 2. pertama kita membuat fungsi yang dinamai selectionSort yang berfungsi mengurutkan deret angka yang diinputkan oleh user, karna untuk mencari median deret harus diurutkan. Kemudian kita membuat fungsi cariMedian untuk mencari nilai median pada deret. Cara kerjanya yaitu jika elemen tengah ada di posisi ganjil ( $n = n\%2 == 1$ ), maka median Median langsung diambil dari elemen di posisi tersebut. Dan jika elemen tengah ada di posisi genap ( $n = n\%2 == 0$ ), maka nilai mediannya adalah rata-rata kedua elemen tengah dihitung dan dibulatkan ke



bawah. Pada func main jika jika user menginput -5313 maka looping dihentikan. Dan jika user menginput 0 maka array yang berisi deret angka akan diurutkan dan dicari nilai median dari array tersebut.

## VI. UNGUIDED 3

```
package main

import "fmt"

const nMax235 = 7919

type Buku235 struct {
    id235      string
    judul235   string
    penulis235 string
    penerbit235 string
    eksemplar235, tahun235, rating235 int
}

type DaftarBuku235 [nMax235]Buku235

// Prosedur untuk mendaftarkan buku ke perpustakaan
func DaftarkanBuku235(pustaka235 *DaftarBuku235, n235 *int) {
    fmt.Print("Masukkan jumlah buku: ")

    fmt.Scan(n235)

    for i235 := 0; i235 < *n235; i235++ {
        fmt.Printf("\nMasukkan data buku ke-%d (id, judul, penulis, penerbit, eksemplar, tahun, rating):\n", i235+1)

        fmt.Scan(&pustaka235[i235].id235,
            &pustaka235[i235].judul235, &pustaka235[i235].penulis235,
            &pustaka235[i235].penerbit235,

            &pustaka235[i235].eksemplar235,
            &pustaka235[i235].tahun235, &pustaka235[i235].rating235)
```

```

}

}

// Prosedur untuk menampilkan buku dengan rating tertinggi
func CetakTerfavorit235(pustaka235 DaftarBuku235, n235 int) {
    if n235 == 0 {
        fmt.Println("Tidak ada data buku.")
        return
    }
    terfavorit235 := pustaka235[0]
    for i235 := 1; i235 < n235; i235++ {
        if pustaka235[i235].rating235 > terfavorit235.rating235 {
            terfavorit235 = pustaka235[i235]
        }
    }
    fmt.Println("\nBuku Terfavorit:")
    fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d,
Rating: %d\n",
        terfavorit235.judul235, terfavorit235.penulis235,
        terfavorit235.penerbit235, terfavorit235.tahun235,
        terfavorit235.rating235)
}

// Prosedur untuk mengurutkan buku berdasarkan rating secara
descending
func UrutBuku235(pustaka235 *DaftarBuku235, n235 int) {
    for i235 := 1; i235 < n235; i235++ {
        key235 := pustaka235[i235]
        j235 := i235 - 1

```

```

for j235 >= 0 && pustaka235[j235].rating235 < key235.rating235
{
    pustaka235[j235+1] = pustaka235[j235]
    j235--
}
pustaka235[j235+1] = key235
}
}

// Prosedur untuk mencetak 5 buku dengan rating tertinggi
func CetakTerbaru235(pustaka235 DaftarBuku235, n235 int) {
    fmt.Println("\n5 Judul Buku Dengan Rating Tertinggi:")
    count235 := 5
    if n235 < 5 {
        count235 = n235
    }

    for i235 := 0; i235 < count235; i235++ {
        fmt.Printf("%d. %s (Rating: %d)\n", i235+1,
pustaka235[i235].judul235, pustaka235[i235].rating235)
    }
}

// Prosedur untuk mencari buku berdasarkan ratingnya
func CariBuku235(pustaka235 DaftarBuku235, n235, r235 int) {
    found235 := false
    fmt.Printf("\nBuku dengan Rating %d:\n", r235)

```

```

for i235 := 0; i235 < n235; i235++ {

    if pustaka235[i235].rating235 == r235 {

        found235 = true

        fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s,
Tahun: %d, Eksemplar: %d, Rating: %d\n",

            pustaka235[i235].judul235,
pustaka235[i235].penulis235, pustaka235[i235].penerbit235,

            pustaka235[i235].tahun235,
pustaka235[i235].eksemplar235, pustaka235[i235].rating235)

        }

    }

    if !found235 {

        fmt.Println("Tidak ada buku dengan rating seperti itu.")

    }

}

func main() {

    var pustaka235 DaftarBuku235

    var n235 int

    var ratingCari235 int

    // Untuk menjalankan Prosedur Daftarkan Buku

    DaftarkanBuku235(&pustaka235, &n235)

    // Untuk menjalankan Prosedur Cetak Buku Terfavorit

    CetakTerfavorit235(pustaka235, n235)


    // Untuk menjalankan Prosedur Urutkan Buku berdasarkan
Rating

    UrutBuku235(&pustaka235, n235)

```

```
// Untuk menjalankan Prosedur Cetak 5 buku dengan rating tertinggi

    CetakTerbaru235(pustaka235, n235)

// Untuk mencari buku berdasarkan rating

    fmt.Print("\nMasukkan rating yang ingin dicari: ")

    fmt.Scan(&ratingCari235)

    CariBuku235(pustaka235, n235, ratingCari235)

}
```

## Screenshoot Output

```
PROBLEMS 111 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR SPELL CHECKER 100
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting> go run "d:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting\unguided3.go"
Masukkan jumlah buku: 6

Masukkan data buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
001 siputih tereliye gramedia 4 2022 8

Masukkan data buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
002 aldebaran tereliye adiksimba 3 2024 10

Masukkan data buku ke-3 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
003 batozar tereliye gramedia 6 2020 8

Masukkan data buku ke-4 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
004 hujan tereliye gramedia 5 2018 7

Masukkan data buku ke-5 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
005 berlayar mgmataram gramedia 3 2023 6

Masukkan data buku ke-6 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
006 ily tereliye adiksimba 6 2023 8
```

```
Buku Terfavorit:
Judul: aldebaran, Penulis: tereliye, Penerbit: adiksimba, Tahun: 2024, Rating: 10

5 Judul Buku Dengan Rating Tertinggi:
1. aldebaran (Rating: 10)
2. siputih (Rating: 8)
3. batozar (Rating: 8)
4. ily (Rating: 8)
5. hujan (Rating: 7)

Masukkan rating yang ingin dicari: 10

Buku dengan Rating 10:
Judul: aldebaran, Penulis: tereliye, Penerbit: adiksimba, Tahun: 2024, Eksemplar: 3, Rating: 10
PS D:\Laprak Algoritma dan Pemrograman 2\Modul 9 Sorting>
```

## Soal Studi Case

Membuat program untuk mencari rating pada suatu buku dan menampilkan top 5 buku terfavorit.

## **Deskripsi Program**

Sintaks tersebut adalah program untuk mengetahui top 5 buku favorit yang ada di perpustakaan dan mencari buku sesuai rating yang kita inginkan. Pertama kita membuat struct buku dengan isi id, judul, penulis, penerbit, eksemplar, dan rating untuk mempresentasikan informasi buku. Kemudian kita membuat prosedur `daftarBuku` untuk memasukkan jumlah buku yang akan didaftarkan dan mengisi data setiap buku ke dalam array `DaftarBuku`. Kemudian kita membuat prosedur `cetakTerfavorit` untuk mencari buku dengan rating tertinggi dalam daftar buku dengan cara membandingkan rating setiap buku untuk menemukan buku dengan rating tertinggi dan menampilkannya. Kemudian kita membuat prosedur `urutBuku` yang berfungsi untuk mengurutkan buku dengan rating lebih tinggi ditempatkan lebih awal dalam array menggunakan algoritma insertion sort. Kemudian kita membuat prosedur `cetakTerbaru` untuk menampilkan 5 buku dengan rating tertinggi berdasarkan daftar yang sudah diurutkan, jika jumlah kurang dari 5 maka yang dicetak sesuai yang diinputkan. Kemudian kita membuat prosedur `cariBuku` untuk mencari buku yang mempunyai rating yang sesuai apa yang diinginkan, jika rating yang ingin dicari tidak sesuai maka program akan menampilkan “tidak ada buku dengan rating seperti itu”.