

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII - XIII  
PENGURUTAN DATA**



**Disusun Oleh :**

**Hamzah Ziyad Ibadurrohman / 2311102254**

**IF-11-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

### Dasar Teori

#### 12.1 Ide Algoritma 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".

- 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.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$idx\_min \leftarrow i - 1$	$idx\_min = i - 1$
4	$j \leftarrow i$	$j = i$
5	while $j < n$ do	for $j < n$ {
6	if $a[idx\_min] > a[j]$ then	if $a[idx\_min] > a[j]$ {
7	$idx\_min \leftarrow j$	$idx\_min = j$
8	endif	}
9	$j \leftarrow j + 1$	$j = j + 1$
10	endwhile	}
11	$t \leftarrow a[idx\_min]$	$t = a[idx\_min]$
12	$a[idx\_min] \leftarrow a[i-1]$	$a[idx\_min] = a[i-1]$
13	$a[i-1] \leftarrow t$	$a[i-1] = t$
14	$i \leftarrow i + 1$	$i = i + 1$
15	endwhile	}

#### 12.2 Algoritma Selection Sort

Adapun algoritma selection sort pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau ascending adalah sebagai berikut ini!

```

..   ...
5   type arrInt [4321]int
..   ...
15  func selectionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara ascending atau membesar dengan SELECTION SORT */
18     var t, i, j, idx_min int
19
20     i = 1
21     for i <= n-1 {
22         idx_min = i - 1
23         j = i
24         for j < n {
25             if T[idx_min] > T[j] {
26                 idx_min = j
27             }
28             j = j + 1
29         }
30         t = T[idx_min]
31         T[idx_min] = T[i-1]
32         T[i-1] = t
33         i = i + 1
34     }
35 }

```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan nilai ekstrim, kemudian tipe data dari variabel t sama dengan struct dari arraynya.

```

..  ...
5  type mahasiswa struct {
..      nama, nim, kelas, jurusan string
..      ipk float64
..  }
..  type arrMhs [2023]mahasiswa
..  ...
15 func selectionSort2(T * arrMhs, n int){
16 /* I.S. terdefinisi array T yang berisi n data mahasiswa
17    F.S. array T terurut secara ascending atau membesar berdasarkan ipk dengan
18    menggunakan algoritma SELECTION SORT */
19     var i, j, idx_min int
20     var t mahasiswa
21     i = 1
22     for i <= n-1 {
23         idx_min = i - 1
24         j = i
25         for j < n {
26             if T[idx_min].ipk > T[j].ipk {
27                 idx_min = j
28             }
29             j = j + 1
30         }
31         t = T[idx_min]
32         T[idx_min] = T[i-1]
33         T[i-1] = t
34         i = i + 1
35     }
36 }

```

## 12.4 Ide Algoritma 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".

Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan:

1. Geser data yang sudah terurut tersebut (ke kanan), sehingga ada satu ruang kosong untuk memasukkan data yang belum terurut ke 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.

	Notasi Algoritma	Notasi dalam bahasa Go
1	$i \leftarrow 1$	$i = 1$
2	while $i \leq n-1$ do	for $i \leq n-1$ {
3	$j \leftarrow i$	$j = i$
4	$temp \leftarrow a[j]$	$temp = a[j]$
5	while $j > 0$ and $temp > a[j-1]$ do	for $j > 0 \ \&\& \ temp > a[j-1]$ {
6	$a[j] \leftarrow a[j-1]$	$a[j] = a[j-1]$
7	$j \leftarrow j - 1$	$j = j - 1$
8	endwhile	}
9	$a[j] \leftarrow temp$	$a[j] = temp$
10	$i \leftarrow i + 1$	$i = i + 1$
11	endwhile	}

## 12.5 Algoritma Insertion Sort

Adapun algoritma insertion sort pada untuk mengurutkan array bertipe data bilangan bulat secara menaik atau ascending adalah sebagai berikut ini!

```

..  ...
5  type arrInt [4321]int
..  ...
15 func insertionSort1(T *arrInt, n int){
16  /* I.S. terdefinisi array T yang berisi n bilangan bulat
17     F.S. array T terurut secara menaik atau ascending dengan INSERTION SORT*/
18     var temp, i, j int
19     i = 1
20     for i <= n-1 {
21         j = i
22         temp = T[j]
23         for j > 0 && temp > T[j-1] {
24             T[j] = T[j-1]
25             j = j - 1
26         }
27         T[j] = temp
28         i = i + 1
29     }
30 }

```

Sama halnya apabila array yang akan diurutkan adalah bertipe data struct, maka tambahkan field pada saat proses perbandingan dalam pencarian posisi, kemudian tipe data dari variabel temp sama dengan struct dari arraynya.

```

..  ...
5  type mahasiswa struct {
..   nama, nim, kelas, jurusan string
..   ipk float64
.. }
.. type arrMhs [2023]mahasiswa
..  ...
15 func insertionSort2(T * arrMhs, n int){
16  /* I.S. terdefinisi array T yang berisi n data mahasiswa
17     F.S. array T terurut secara mengecil atau descending berdasarkan nama dengan
18     menggunakan algoritma INSERTION SORT */
19     var temp i, j int
20     var temp mahasiswa
21     i = 1
22     for i <= n-1 {
23         j = i
24         temp = T[j]
25         for j > 0 && temp.nama > T[j-1].nama {
26             T[j] = T[j-1]
27             j = j - 1
28         }
29         T[j] = temp
30         i = i + 1

```

## II. GUIDED

### 1. Soal Studi Case

Suatu lingkaran didefinisikan dengan koordinat titik pusat (cx, cy) dengan radius r. Apabila diberikan dua buah lingkaran, maka tentukan posisi sebuah titik sembarang (x, y) berdasarkan dua lingkaran tersebut. Gunakan tipe bentukan titik untuk menyimpan koordinat, dan tipe bentukan lingkaran untuk menyimpan titik pusat lingkaran dan radiusnya.

Masukan terdiri dari beberapa tiga baris. Baris pertama dan kedua adalah koordinat titik pusat dan radius dari lingkaran 1 dan lingkaran 2, sedangkan baris ketiga adalah koordinat titik sembarang. Asumsi sumbu x dan y dari semua titik dan juga radius direpresentasikan dengan bilangan bulat.

Keluaran berupa string yang menyatakan posisi titik "Titik di dalam lingkaran 1 dan 2", "Titik di dalam lingkaran 1", "Titik di dalam lingkaran 2", atau "Titik di luar lingkaran 1 dan 2".

#### 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 C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12> go run guided1.go
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 4
Masukkan 4 nomor rumah kerabat: 16
76
89
100
Nomor rumah terurut untuk daerah 1: 16 76 89 100

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 3
Masukkan 3 nomor rumah kerabat: 20
9
99
Nomor rumah terurut untuk daerah 2: 9 20 99

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 4
Masukkan 4 nomor rumah kerabat: 28
48
32
19
Nomor rumah terurut untuk daerah 3: 19 28 32 48
PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12>

```

## Deskripsi Program

Ini adalah Program yang dibuat untuk mengurutkan nomor rumah kerabat di beberapa daerah menggunakan algoritma Selection Sort. selectionSort bekerja dengan cara mencari elemen terkecil di subarray yang belum



diurutkan dan menempatkannya di posisi yang sesuai dalam array. Proses ini diulang untuk setiap elemen array hingga array terurut sepenuhnya. Setelah pengurutan selesai, program mencetak nomor rumah yang sudah terurut untuk daerah tersebut. Hasil ditampilkan untuk setiap daerah dalam baris baru, dengan nomor rumah diurutkan dari kecil ke besar. Program ini membaca jumlah daerah kerabat, lalu untuk setiap daerah, membaca jumlah nomor rumah kerabat serta daftar nomor rumah tersebut, dan mengurutkannya dalam urutan terkecil ke terbesar. Setelah itu, program menampilkan nomor rumah yang sudah terurut untuk setiap daerah.

Pada fungsi utama main, program dimulai dengan meminta pengguna memasukkan jumlah daerah kerabat (n). Kemudian, untuk setiap daerah, pengguna diminta memasukkan jumlah nomor rumah (m) dan daftar nomor rumahnya. Array untuk menyimpan nomor rumah ini dibuat menggunakan fungsi make, dan elemen-elemen di dalamnya diisi melalui masukan pengguna. Setelah nomor rumah dibaca, fungsi selectionSort dipanggil untuk mengurutkan array tersebut.

## 2. Soal Studi Case

```
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")  
        }  
    }  
}
```

## Screenshot Output

```
PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12> go run guided2.go  
Masukkan data integer (akhiri dengan bilangan negatif):  
9 7 5 3 1 -1 -3 -5  
Array setelah diurutkan:  
1 3 5 7 9  
Data berjarak 2  
PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12>
```

## Deskripsi Program

Ini adalah program yang dibuat untuk mengurutkan array bilangan bulat yang dimasukkan oleh pengguna dan memeriksa apakah elemen-elemen dalam array memiliki selisih tetap di antara mereka. Program ini menggunakan Insertion Sort untuk mengurutkan array dan fungsi tambahan untuk memeriksa apakah selisih antara elemen-elemen array adalah konstan.

Pada bagian awal main, pengguna diminta untuk memasukkan sejumlah bilangan bulat, dan input akan terus diterima hingga bilangan negatif dimasukkan, yang menandakan akhir dari input. Setiap bilangan yang dimasukkan disimpan dalam slice arr. Setelah input selesai, panjang slice disimpan dalam variabel n. Kemudian, array diurutkan menggunakan fungsi insertionSort. Fungsi ini bekerja dengan cara membandingkan elemen saat ini (key) dengan elemen-elemen sebelumnya dan memindahkan elemen yang lebih besar ke kanan, sehingga elemen terkecil ditempatkan di posisi yang benar. Setelah array diurutkan, program memanggil fungsi isConstantDifference, yang memeriksa apakah selisih antara elemen-elemen berturut-turut dalam array adalah sama. Fungsi ini mengembalikan true dan selisih tersebut jika selisihnya tetap, atau false jika tidak. Program kemudian mencetak array yang sudah diurutkan dan menampilkan apakah data memiliki selisih tetap atau tidak. Jika selisihnya tetap, program mencetak "Data berjarak [selisih]", sedangkan jika tidak, program mencetak "Data berjarak tidak tetap".

### III. UNGUIDED

1. Berisi source code dan output dari kegiatan praktikum yang telah dilaksanakan. Source Code diberi penjelasan maka akan menjadi nilai ++

#### Soal 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 nomor 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.

#### Sourcecode

```
package main

import "fmt"

func selectionSort(arr []int, ascending bool) []int {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        extremeIdx := i
        for j := i + 1; j < n; j++ {
            if (ascending && arr[j] <
arr[extremeIdx]) || (!ascending && arr[j] >
arr[extremeIdx]) {
                extremeIdx = j
            }
        }
        arr[i], arr[extremeIdx] = arr[extremeIdx],
arr[i]
    }
    return arr
}

func prosesrumah(rumah []int) {
    var angkaganjil, angkagenap []int

    for _, num := range rumah {
        if num%2 == 0 {
            angkagenap = append(angkagenap, num)
        } else {
            angkaganjil = append(angkaganjil, num)
        }
    }
}
```

```

    }

    angkaganjil = selectionSort(angkaganjil, true)
    angkagenap = selectionSort(angkagenap, false)

    for _, num := range angkaganjil {
        fmt.Printf("%d ", num)
    }
    for _, num := range angkagenap {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}

func main() {
    var t int
    fmt.Scan(&t)

    for i := 0; i < t; i++ {
        var n int
        fmt.Scan(&n)

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

```

## Screenshoot Output

```

PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12> go run unguided1.go
2
5
1 3 7 9 12
1 3 7 9 12
5
10 20 13 24 50
13 50 24 20 10
PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12>

```

## Deskripsi Program

Ini adalah program yang dibuat untuk untuk memproses data berupa nomor rumah yang dimasukkan oleh pengguna, kemudian mengurutkan dan menampilkan nomor rumah yang ada dalam dua kategori: nomor ganjil yang diurutkan secara menaik, dan nomor genap yang diurutkan secara

menurun. Program ini menggunakan algoritma Selection Sort untuk mengurutkan array.

Program ini meminta pengguna untuk memasukkan jumlah daerah (t) dan nomor rumah (n) untuk setiap daerah. Setiap daftar nomor rumah diproses dengan memisahkan angka ganjil dan genap ke dalam dua slice. Angka ganjil diurutkan secara menaik, sedangkan angka genap diurutkan secara menurun menggunakan fungsi selectionSort. Setelah diurutkan, program mencetak angka ganjil diikuti angka genap, masing-masing dipisahkan oleh spasi untuk setiap daerah.

## 2. Soal Studi Case

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"

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.

## Sourcecode

```
package main

import (
    "fmt"
)
```

```

// Fungsi untuk menghitung median
func hitungMedian(arr []int) int {
    n := len(arr)
    if n%2 == 1 {
        return arr[n/2]
    } else {
        return (arr[n/2-1] + arr[n/2]) / 2
    }
}

// Fungsi untuk menyisipkan elemen ke dalam array menggunakan
insertion sort
func insertionSort(arr []int) []int {
    for i := 1; i < len(arr); i++ {
        key := arr[i]
        j := i - 1
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
    return arr
}

func main() {
    var angka int
    data := []int{}

```

```

        for {

            fmt.Scan(&angka)

            if angka == 0 {

                // Ketika menemukan 0, urutkan dan hitung
median
                data = insertionSort(data)

                median := hitungMedian()(data)

                fmt.Println(median)

            } else if angka == -5313 {

                // Jika menemukan marker akhir, keluar dari
loop
                break

            } else {

                // Tambahkan bilangan ke array

                data = append(data, angka)

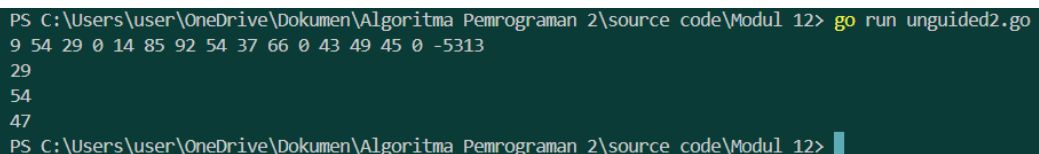
            }

        }

    }
}

```

## Screenshot output



```

PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12> go run unguided2.go
9 54 29 0 14 85 92 54 37 66 0 43 49 45 0 -5313
29
54
47
PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12>

```

## Deskripsi Output

Ini adalah program yang dibuat untuk menghitung median dari data bilangan bulat yang terus bertambah hingga ditemukan bilangan 0, di situ median akan dihitung



dan ditampilkan. Penghitungan akan terus berlanjut hingga bilangan penanda akhir -5313 ditemukan. Program ini memakai insert sort untuk mengurutkan data setiap kali 0 ditemukan. Data diurutkan secara bertahap, dan setelah urutan lengkap, median dihitung. Jika jumlah elemen ganjil, median adalah elemen tengah. Jika genap, median dihitung sebagai rata-rata dua elemen tengah yang dibulatkan ke bawah. Setelah menemukan bilangan 0, program mencetak median saat itu, dan proses dilanjutkan dengan data baru.

### 3. Soal Studi Case

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan.

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.

#### Sourcecode

```
package main

import "fmt"

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

const nMax = 7919
```

```

type DaftarBuku [nMax]Buku

func DaftarkanBuku(pustaka *DaftarBuku, n *int) {
    fmt.Scan(n) // Membaca jumlah buku yang didaftar.
    ada berapa buku trus dibaca ama program

    for i := 0; i < *n; i++ {
        var buku Buku

        fmt.Scan(&buku.id, &buku.judul,
        &buku.penulis, &buku.penerbit, &buku.eksemplar,
        &buku.tahun, &buku.rating)

        pustaka[i] = buku
    }
}

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

        bukuFavorit.judul, bukuFavorit.penulis,
        bukuFavorit.penerbit, bukuFavorit.tahun)
}

```

```

func UrutBuku(pustaka *DaftarBuku, n int) {

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

        temp := pustaka[i]

        j := i - 1

        for j >= 0 && pustaka[j].rating <
temp.rating {

            pustaka[j+1] = pustaka[j]

            j--

        }

        pustaka[j+1] = temp

    }

}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {

    m := 5

    if n < 5 {

        m = n

    }

    for i := 0; i < m; i++ {

        fmt.Println(pustaka[i].judul)

    }

}

func CariBuku(pustaka DaftarBuku, n int, r int) {

    low, high := 0, n-1

    for low <= high {

        mid := (low + high) / 2

        if pustaka[mid].rating == r {

```

```

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

                pustaka[mid].judul,
pustaka[mid].penulis, pustaka[mid].penerbit,
pustaka[mid].tahun, pustaka[mid].eksemplar,
pustaka[mid].rating)

        return

    } else if pustaka[mid].rating < r {

        low = mid + 1

    } else {

        high = mid - 1

    }

}

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

func main() {

    var pustaka DaftarBuku

    var n int

    DaftarkanBuku(&pustaka, &n)

    CetakTerfavorit(pustaka, n)

    UrutBuku(&pustaka, n)

    Cetak5Terbaru(pustaka, n)

    var rating int

    fmt.Scan(&rating)

```

```
CariBuku(pustaka, n, rating)

}
```

## Screenshot Output

```
PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12> go run unguided3.go
2
1 "Bumi" "Tere" "Gramedia" 2 2014 5
2 "Roshidere" "siapa" "Gramedia" 2 2023 6
Judul: "Roshidere", Penulis: "siapa", Penerbit: "Gramedia", Tahun: 2023
"Roshidere"
"Bumi"
6
Judul: "Roshidere", Penulis: "siapa", Penerbit: "Gramedia", Tahun: 2023, Eksemplar: 2, Rating: 6
PS C:\Users\user\OneDrive\Dokumen\Algoritma Pemrograman 2\source code\Modul 12> █
```

## Deskripsi Program

Ini adalah program yang dibuat untuk mengelola data buku di perpustakaan dengan beberapa fitur: mengurutkan buku berdasarkan rating secara menurun, menampilkan buku dengan rating tertinggi, mencetak lima buku terbaik berdasarkan rating, dan mencari buku dengan rating tertentu. Di Program ini memakai insert sort yang digunakan untuk mengurutkan array buku berdasarkan rating secara descending. Setelah data diurutkan, buku dengan rating tertinggi menjadi elemen pertama array, lima buku terbaik adalah lima elemen pertama, dan pencarian buku dilakukan menggunakan binary search pada array yang sudah terurut. contoh, jika terdapat 5 buku dengan data seperti BukuA memiliki rating 90, BukuC memiliki rating 95, dan seterusnya, maka outputnya akan mencetak BukuC sebagai buku terfavorit, diikuti oleh lima buku dengan rating tertinggi, serta data buku yang sesuai dengan rating yang dicari atau pesan jika buku tersebut tidak ditemukan.

