

**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI  
OBJEK**

**MODUL XII & XIII  
PENGURUTAN DATA**



**Oleh :**

**NAMA : FAISAL KHOIRUDDIN**

**NIM : 2311102046**

**Kelas : IF-11-02**

**S1 TEKNIK INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
2024**

## I. DASAR TEORI

### 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	}

### 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 }

```

```

.. ...
5  type arrInt [2023]int
.. ...
15
16 func terkecil_2(tabInt arrInt, n int) int {
17  /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi
18  n bilangan bulat */
19      var idx int = 0          // idx berisi indeks data pertama
20      var j int = 1           // pencarian dimulai dari data berikutnya
21      for j < n {
22          if tabInt[idx] > tabInt[j] { // pengecekan apakah nilai minimum valid
23              idx = j                // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return idx                // returnkan indeks nilai minimumnya
28 }

```

### 3. 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".

- 1) Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan: 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	}

#### 4. Algoritma Insertion Sort

Adapun algoritma insertion sort pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau descending 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 mengecil atau descending 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
31     }
32 }

```

## II. GUIDED

### 1. Guided 1

#### Source Code

```
package main

import "fmt"

// Fungsi untuk mengurutkan array
menggunakan selection sort
func selectionSort(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        maxIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] > arr[maxIdx] { //
Cari elemen terbesar
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx],
arr[i] // Tukar elemen
    }
}

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

```
    if n <= 0 || n >= 1000 {
        fmt.Println("n harus lebih besar
dari 0 dan kurang dari 1000.")
        return
    }

    for i := 0; i < n; i++ {
        var m int
        fmt.Printf("Masukkan jumlah rumah
kerabat untuk daerah ke-%d: ", i+1)
        fmt.Scan(&m)

        if m <= 0 || m >= 1000000 {
            fmt.Println("m harus lebih
besar dari 0 dan kurang dari 1000000.")
            return
        }

        // Masukkan nomor rumah
        houses := make([]int, m)
        fmt.Printf("Masukkan nomor rumah
kerabat untuk daerah ke-%d: ", i+1)
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }

        // Urutkan dengan selection sort
        selectionSort(houses)

        // Cetak hasil
```



```

        fmt.Printf("Hasil urutan rumah
untuk daerah ke-%d: ", i+1)

        for _, house := range houses {
            fmt.Printf("%d ", house)
        }

        fmt.Println()
    }
}

```

### Screenshots Output

```

PS C:\Semester3Go\Praktikum\Modul 12\Guided\No_1> go run Guided1.go
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 2
Masukkan nomor rumah kerabat untuk daerah ke-1: 2 34
Hasil urutan rumah untuk daerah ke-1: 34 2
Masukkan jumlah rumah kerabat untuk daerah ke-2: 2
Masukkan nomor rumah kerabat untuk daerah ke-2: 9 8
Hasil urutan rumah untuk daerah ke-2: 9 8
PS C:\Semester3Go\Praktikum\Modul 12\Guided\No_1>

```

### Deskripsi:

Program tersebut merupakan program rumahkerabat yang akan menyusun nomor-nomor rumah kerabatnya menggunakan algoritma selection sort. Pengguna diminta menginputkan jumlah daerah, jumlah rumah kerabat untuk daerah ke-i, dan masukkan nomor rumah dan masukkan nomor jumlah kerabat untuk daerah ke-i. Output dari program tersebut yaitu hasil urutan rumah pada masing-masing daerah.

- package main → paket utama program golang
- import "fmt" → mengimpor fmt
- func selectionSort(arr []int) { → Fungsi untuk mengurutkan array menggunakan selection sort
- n := len(arr) → menyimpan Panjang array ke dalam variabel n
- for i := 0; i < n-1; i++ { → perulangan untuk menemukan elemen terbesar

- `maxIdx := I` ➔ menyimpan indeks elemen sementara
- `for j := i + 1; j < n; j++ {` ➔ perulangan untuk menemukan elemen terbesar
- `if arr[j] > arr[maxIdx] {` ➔ percabangan if jika `arr[j]` lebih besar dari `arr[maxIdx]`
- `maxIdx = j` ➔ menyimpan indeks `j` sebagai indeks elemen terbesar sementara
- `arr[i], arr[maxIdx] = arr[maxIdx], arr[i]` ➔ Tukar elemen
- `func main() {` ➔ merupakan fungsi utama
- `var n int` ➔ deklarasi variabel `n` bertipe data integer
- `fmt.Print("Masukkan jumlah daerah (n): ")` ➔ menampilkan statement Masukkan jumlah daerah (n):
- `fmt.Scan(&n)` ➔ membaca input dan menyimpan di variabel `n`
- `if n <= 0 || n >= 1000 {` ➔ percabangan if jika `n <= 0` atau `n >= 1000`
- `fmt.Println("n harus lebih besar dari 0 dan kurang dari 1000.")` ➔ menampilkan statement `n` harus lebih besar dari 0 dan kurang dari 1000
- `return` ➔ return
- `for i := 0; i < n; i++ {` ➔ perulangan sebanyak jumlah daerah
- `var m int` ➔ deklarasi variabel `m` bertipe data integer
- `fmt.Printf("Masukkan jumlah rumah kerabat untuk daerah ke-%d: ", i+1)` ➔ menampilkan statement Masukkan jumlah rumah kerabat untuk daerah ke-
- `fmt.Scan(&m)` ➔ membaca input dan menyimpan di variabel `m`
- `if m <= 0 || m >= 1000000 {` ➔ percabangan if jika `m <= 0` atau `m >= 1000000`
- `fmt.Println("m harus lebih besar dari 0 dan kurang dari 1000000.")` ➔ menampilkan statement `m` harus lebih besar dari 0 dan kurang dari 1000000
- `return` ➔ return

- `houses := make([]int, m)` ➔ membuat array `houses` dengan Panjang `m`
- `fmt.Printf("Masukkan nomor rumah kerabat untuk daerah ke-%d: ", i+1)` ➔ menampilkan statement Masukkan nomor rumah kerabat untuk daerah ke-
- `for j := 0; j < m; j++ {` ➔ perulangan sebanyak jumlah kerabat
- `fmt.Scan(&houses[j])` ➔ membaca input pengguna untuk setiap nomor rumah dan menyimpan di array `houses`
- `selectionSort(houses)` ➔ memanggil fungsi `selectionSort` untuk mengurutkan array `houses`
- `fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)` ➔ menampilkan judul hasil urutan rumah untuk daerah ke `i`
- `for _, house := range houses {` ➔ perulangan melalui setiap elemen di array `house`
- `fmt.Printf("%d ", house)` ➔ menampilkan setiap nomor rumah yang sudah diurutkan
- `fmt.Println()` ➔ membuat baris baru

## 2. Guided 2

### Source Code

```
package main

import (
    "fmt"
    "math"
)

// Fungsi insertion sort untuk mengurutkan
array
func insertionSort(arr []int) {
    n := len(arr)
```

```

    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 data
berjarak tetap
func isDataConsistentlySpaced(arr []int)
(bool, int) {
    if len(arr) < 2 {
        return true, 0 // Array dengan
kurang dari 2 elemen dianggap berjarak
tetap
    }

    // Hitung selisih awal
    diff := int(math.Abs(float64(arr[1] -
arr[0])))

    for i := 1; i < len(arr)-1; i++ {
        currentDiff :=

```

```

int(math.Abs(float64(arr[i+1] - arr[i])))
    if currentDiff != diff {
        return false, 0 // Jika ada
selisih yang berbeda, tidak berjarak tetap
    }
}

return true, diff
}

func main() {
    var data []int
    var input int

    fmt.Println("Masukkan data (akhiri
dengan bilangan negatif):")
    for {
        fmt.Scan(&input)
        if input < 0 {
            break
        }
        data = append(data, input)
    }

    // Urutkan data menggunakan insertion
sort
    insertionSort(data)

    // Periksa apakah data berjarak tetap
isConsistent, diff :=

```

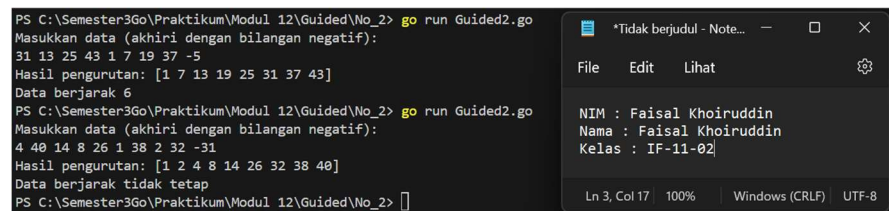
```

isDataConsistentlySpaced(data)

// Cetak hasil
fmt.Println("Hasil pengurutan:", data)
if isConsistent {
    fmt.Printf("Data berjarak %d\n",
diff)
} else {
    fmt.Println("Data berjarak tidak
tetap")
}
}
}

```

### Screenshots Output



The screenshot shows a terminal window on the left and a Notepad window on the right. The terminal displays the execution of a Go program named 'Guided2.go'. It prompts the user to enter data, which is then sorted and checked for consistent spacing. The first run shows data [31 13 25 43 1 7 19 37 -5] being sorted to [1 7 13 19 25 31 37 43] and identified as having a consistent spacing of 6. The second run shows data [4 40 14 8 26 1 38 2 32 -31] being sorted to [1 2 4 8 14 26 32 38 40] and identified as not having a consistent spacing.

```

PS C:\Semester3Go\Praktikum\Modul 12\Guided\No_2> go run Guided2.go
Masukkan data (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Hasil pengurutan: [1 7 13 19 25 31 37 43]
Data berjarak 6
PS C:\Semester3Go\Praktikum\Modul 12\Guided\No_2> go run Guided2.go
Masukkan data (akhiri dengan bilangan negatif):
4 40 14 8 26 1 38 2 32 -31
Hasil pengurutan: [1 2 4 8 14 26 32 38 40]
Data berjarak tidak tetap
PS C:\Semester3Go\Praktikum\Modul 12\Guided\No_2>

```

The Notepad window on the right contains the following text:

```

NIM : Faisal Khoiruddin
Nama : Faisal Khoiruddin
Kelas : IF-11-02

```

The status bar at the bottom of the Notepad window shows 'Ln 3, Col 17', '100%', 'Windows (CRLF)', and 'UTF-8'.

### Deskripsi:

Program tersebut merupakan program yang digunakan untuk membaca data integer, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya. Pada program tersebut, pengguna diminta menginputkan dua bilangan bulat x dan y pada baris pertama. Bilangan x menyatakan banyaknya ikan yang akan dijual, sedangkan y adalah banyaknya ikan yang akan dimasukkan ke dalam wadah. Kemudian meinputkan bilangan riil yang menyatakan banyaknya ikan yang akan dijual. Program tersebut menampilkan kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah pada baris pertama dan bilangan riil yang menyatakan berat

rata-rata ikan di setiap wadah pada baris kedua.

- `package main` → paket utama program golang
- `import` → mengimpor
- `"fmt"` → mengimpor `fmt`
- `"math"` → mengimpor `math`
- `func insertionSort(arr []int) {` → fungsi insertion sort
- `n := len(arr)` → mendapatkan Panjang array
- `for i := 1; i < n; i++ {` → perulangan for untuk `for i := 1; i < n;`  
`i++(increment)`
- `key := arr[i]` → menyimpan elemen saat ini sebagai kunci
- `j := i - 1` → indeks elemen sebelumnya
- `for j >= 0 && arr[j] > key {` → perulangan untuk menggeser  
elemen ke kanan jika lebih besar dari kunci
- `arr[j+1] = arr[j]` → geser elemen ke kanan
- `j--` → mengurangi indeks
- `arr[j+1] = key` → menempatkan kunci pada posisi yang tepat
- `func isDataConsistentlySpaced(arr []int) (bool, int) {` → Fungsi  
untuk memeriksa apakah data berjarak tetap
- `if len(arr) < 2 {` → jika Panjang array kurang dari 2
- `return true, 0` → Array dengan kurang dari 2 elemen dianggap  
berjarak tetap
- `diff := int(math.Abs(float64(arr[1] - arr[0])))` → Hitung selisih  
awal
- `for i := 1; i < len(arr)-1; i++ {` → perulangan for untuk `for i := 1;`

`i < n; i++(increment)`

- `currentDiff := int(math.Abs(float64(arr[i+1] - arr[i])))` → memeriksa jika selisih berbeda dari selisih awal
- `if currentDiff != diff {` → memeriksa jika selisih berbeda dari selisih awal
- `return false, 0` → Jika ada selisih yang berbeda, tidak berjarak tetap
- `return true, diff` → mengembalikan true dan selisih jika semua selisih sama
- `func main() {` → merupakan fungsi utama
- `var data []int` → deklarasi slice untuk menyimpan data
- `var input int` → deklarasi variabel input bertipe data integer
- `fmt.Println("Masukkan data (akhiri dengan bilangan negatif):")`  
→ menampilkan statement untuk Masukkan data (akhiri dengan bilangan negatif)
- `for {` → perulangan for
- `fmt.Scan(&input)` → membaca input pengguna
- `if input < 0 {` → jika input kurang dari 0
- `break` → keluar dari perulangan
- `data = append(data, input)` → menambahkan input ke slice data
- `insertionSort(data)` → Urutkan data menggunakan insertion sort
- `isConsistent, diff := isDataConsistentlySpaced(data)` → Periksa apakah data berjarak tetap
- `fmt.Println("Hasil pengurutan:", data)` → menampilkan hasil pengurutan



- `if isConsistent {` → jika data berjarak tetap
- `fmt.Printf("Data berjarak %d\n", diff)` → menampilkan jarak data
- `}` `else {` → kalau tidak
- `fmt.Println("Data berjarak tidak tetap")` → menampilkan statement Data berjarak tidak tetap

### 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.

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.

Keterangan: Terdapat 3 daerah dalam contoh masukan. Baris kedua berisi campuran bilangan ganjil dan genap. Baris berikutnya hanya berisi bilangan ganjil, dan baris terakhir hanya berisi bilangan genap. Petunjuk:

- Waktu pembacaan data, bilangan ganjil dan genap dipisahkan ke dalam dua array yang berbeda, untuk kemudian masing-masing diurutkan tersendiri.
- Atau, tetap disimpan dalam satu array, diurutkan secara

keseluruhan. Tetapi pada waktu pencetakan, mulai dengan mencetak semua nilai ganjil lebih dulu, kemudian setelah selesai cetaklah semua nilai genapnya.

### Source Code

```
package main

import "fmt"

func selectionSort(arr []int, menaik bool)
{
    n := len(arr)
    for i := 0; i < n-1; i++ {
        targetIdx := i
        for j := i + 1; j < n; j++ {
            if menaik {
                if arr[j] < arr[targetIdx]
            {
                targetIdx = j
            }
            } else {
                if arr[j] > arr[targetIdx]
            {
                targetIdx = j
            }
        }
        arr[i], arr[targetIdx] =
        arr[targetIdx], arr[i]
    }
}
```

```
func main() {  
    var jumlahDaerah int  
    fmt.Print("Masukkan jumlah daerah (n):  
    ")  
    fmt.Scan(&jumlahDaerah)  
  
    if jumlahDaerah <= 0 || jumlahDaerah >= 1000 {  
        fmt.Println("Jumlah daerah harus lebih besar dari 0 dan kurang dari 1000.")  
        return  
    }  
  
    for i := 0; i < jumlahDaerah; i++ {  
        var jumlahRumah int  
        fmt.Printf("Masukkan jumlah rumah kerabat untuk daerah ke-%d: ", i+1)  
        fmt.Scan(&jumlahRumah)  
  
        if jumlahRumah <= 0 || jumlahRumah >= 1000000 {  
            fmt.Println("jumlah rumah harus lebih besar dari 0 dan kurang dari 1000000.")  
            return  
        }  
  
        rumah := make([]int, jumlahRumah)
```

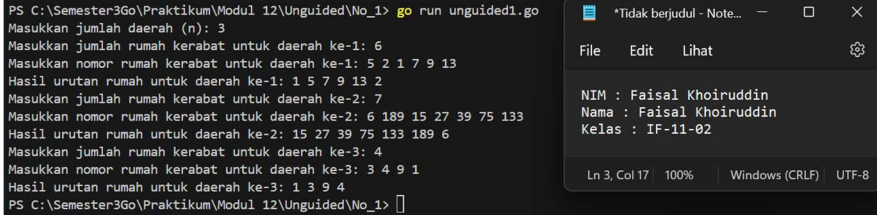
```
        fmt.Printf("Masukkan nomor rumah  
kerabat untuk daerah ke-%d: ", i+1)  
        for j := 0; j < jumlahRumah; j++ {  
            fmt.Scan(&rumah[j])  
        }  
  
        var rumahGanjil, rumahGenap []int  
        for _, nomorRumah := range rumah {  
            if nomorRumah%2 == 1 {  
                rumahGanjil =  
append(rumahGanjil, nomorRumah)  
            } else {  
                rumahGenap =  
append(rumahGenap, nomorRumah)  
            }  
        }  
  
        selectionSort(rumahGanjil, true)  
        selectionSort(rumahGenap, false)  
  
        fmt.Printf("Hasil urutan rumah  
untuk daerah ke-%d: ", i+1)  
        for _, nomorRumah := range  
rumahGanjil {  
            fmt.Printf("%d ", nomorRumah)  
        }  
        for _, nomorRumah := range  
rumahGenap {  
            fmt.Printf("%d ", nomorRumah)  
        }  
    }
```

```

        fmt.Println()
    }
}

```

## Screenshots Output



```

PS C:\Semester3Go\Praktikum\Modul 12\Unguided\No_1> go run unguided1.go
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 6
Masukkan nomor rumah kerabat untuk daerah ke-1: 5 2 1 7 9 13
Hasil urutan rumah untuk daerah ke-1: 1 5 7 9 13 2
Masukkan jumlah rumah kerabat untuk daerah ke-2: 7
Masukkan nomor rumah kerabat untuk daerah ke-2: 6 189 15 27 39 75 133
Hasil urutan rumah untuk daerah ke-2: 15 27 39 75 133 189 6
Masukkan jumlah rumah kerabat untuk daerah ke-3: 4
Masukkan nomor rumah kerabat untuk daerah ke-3: 3 4 9 1
Hasil urutan rumah untuk daerah ke-3: 1 3 9 4
PS C:\Semester3Go\Praktikum\Modul 12\Unguided\No_1>

```

## Deskripsi:

Program tersebut merupakan 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. Program tersebut meminta pengguna menginput jumlah daerah, jumlah rumah kerabat dan masukkan nomor rumah kerabat. Program tersebut pada urutan rumah untuk daerah ke-1 menampilkan hasil urutan rumah untuk daerah dengan inputan 5 2 1 7 9 13 maka outputnya yaitu 1 5 7 9 13 2 yang mana rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap pada urutan rumah untuk daerah ke-1. Pada urutan rumah untuk daerah ke-2 menampilkan hasil urutan rumah untuk daerah dengan inputan 6 189 15 27 39 75 133 maka outputnya yaitu 15 27 39 75 133 189 6 yang mana rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap pada urutan rumah untuk daerah ke-2. Pada urutan rumah untuk daerah ke-3 menampilkan hasil urutan rumah untuk daerah dengan inputan 3 4 9 1 maka outputnya yaitu 1 3 9 4 yang mana rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap pada urutan rumah

untuk daerah ke-3.

- `package main` → paket utama pada program golang
- `import "fmt"` → import fmt
- `func selectionSort(arr []int, menaik bool) {` → fungsi selection sort
- `n := len(arr)` → mendapatkan array
- `for i := 0; i < n-1; i++ {` → perulangan for `i := 0; i < n-1; i++(increment)`
- `targetIdx := i` → menyimpan indeks target
- `for j := i + 1; j < n; j++ {` → perulangan for `j := i + 1; j < n; j++(increment)`
- `if menaik {` → percabangan if jika menaik
- `if arr[j] < arr[targetIdx] {` → jika elemen j lebih kecil dari elemen target
- `targetIdx = j` → perbarui indeks target
- `else {` → kalau tidak
- `if arr[j] > arr[targetIdx] {` → jika elemen j lebih besar dari elemen target
- `targetIdx = j` → perbarui elemen j
- `arr[i], arr[targetIdx] = arr[targetIdx], arr[i]` → tukar elemen di indeks i dengan elemen target
- `func main() {` → merupakan fungsi utama
- `var jumlahDaerah int` → deklarasi variabel jumlah daerah bertipe data integer

- `fmt.Print("Masukkan jumlah daerah (n): ")` → menampilkan statement untuk Masukkan jumlah daerah (n):
- `fmt.Scan(&jumlahDaerah)` → membaca input pengguna untuk jumlah rumah
- `if jumlahDaerah <= 0 || jumlahDaerah >= 1000 {` → jika jumlah daerah kurang dari sama dengan 0 atau jumlah daerah lebih dari sama dengan 1000
- `fmt.Println("Jumlah daerah harus lebih besar dari 0 dan kurang dari 1000.")` → menampilkan statement Jumlah daerah harus lebih besar dari 0 dan kurang dari 1000."
- `return` → menghentikan program jika input tidak valid
- `for i := 0; i < jumlahDaerah; i++ {` → perulangan untuk setiap daerah
- `var jumlahRumah int` → deklarasi variabel jumlahRumah bertipe data integer
- `fmt.Printf("Masukkan jumlah rumah kerabat untuk daerah ke-%d: ", i+1)` → menampilkan statement Masukkan jumlah rumah kerabat untuk daerah ke-i
- `fmt.Scan(&jumlahRumah)` →
- `if jumlahRumah <= 0 || jumlahRumah >= 1000000 {` → jika jumlahRumah kurang dari sama dengan 0 atau jumlahRumah lebih sama dengan 1000000
- `fmt.Println("jumlah rumah harus lebih besar dari 0 dan kurang dari 1000000.")` → menampilkan sstatement jumlah rumah harus lebih besar dari 0 dan kurang dari 1000000
- `return` → menghentikan program jika tidak valid

- `rumah := make([]int, jumlahRumah)` ➔ membuat slice untuk menyimpan nomor rumah ganjil dan genap
- `fmt.Printf("Masukkan nomor rumah kerabat untuk daerah ke-%d: ", i+1)` ➔ menampilkan statement untuk Masukkan nomor rumah kerabat untuk daerah ke-i
- `for j := 0; j < jumlahRumah; j++ {` ➔ perulangan untuk membaca setiap nomor rumah
- `fmt.Scan(&rumah[j])` ➔ membaca nomor rumah dari input user
- `var rumahGanjil, rumahGenap []int` ➔ deklarasi slice untuk nomor rumah ganjil dan genap
- `for _, nomorRumah := range rumah {` ➔ perulangan untuk setiap nomor rumah
- `if nomorRumah%2 == 1 {` ➔ memeriksa apakah nomor rumah ganjil
- `rumahGanjil = append(rumahGanjil, nomorRumah)` ➔ menambahkan nomor rumah ganjil ke slice rumahGanjil
- `}` `else {` ➔ kalau tidak
- `rumahGenap = append(rumahGenap, nomorRumah)` ➔ menambahkan nomor rumah genap ke slice rumahGenap
- `selectionSort(rumahGanjil, true)` ➔ mengurutkan nomor rumah ganjil secara menaik
- `selectionSort(rumahGenap, false)` ➔ mengurutkan nomor rumah genap secara menurun
- `fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)` ➔ menampilkan hasil urutan rumah untuk daerah ke-i



- `for _, nomorRumah := range rumahGanjil {` ➔ perulangan untuk menampilkan nomor rumah ganjil
- `fmt.Printf("%d ", nomorRumah)` ➔ menampilkan nomor ganjil
- `for _, nomorRumah := range rumahGenap {` ➔ perulangan untuk menampilkan nomor rumah genap
- `fmt.Printf("%d ", nomorRumah)` ➔ menampilkan nomor genap
- `fmt.Println()` ➔ menampilkan baris baru

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.

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 tersusun diperoleh: 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.

### Source Code

```
package main

import (
    "fmt"
    "sort"
)

func hitungMedian(arr []int) int {
    sort.Ints(arr)
    n := len(arr)

    if n%2 == 0 {
        return (arr[n/2-1] + arr[n/2]) / 2
    }
}
```

```

    }

    return arr[n/2]
}

func main() {
    var num int
    var data []int

    for {
        fmt.Scan(&num)
        if num == -5313 {
            break
        }
        if num == 0 {
            if len(data) > 0 {
                median :=
hitungMedian(data)
                fmt.Println(median)
            }
        } else {
            data = append(data, num)
        }
    }
}

```

## Screenshots Output

```

PS C:\Semester3Go\Praktikum\Modul 12\Unguided\No_2> go run Unguided2.go
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS C:\Semester3Go\Praktikum\Modul 12\Unguided\No_2>

```

Deskripsi:

Program tersebut merupakan program median yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0. User menginputkan bilangan bulat dan diakhiri dengan bilangan bulat -5313. Output menampilkan median yang diminta, satu data per baris.

- `package main` → paket utama pada program golang
- `import` → mengimport
- `"fmt"` → import fmt
- `"sort"` → import sort
- `func hitungMedian(arr []int) int {` → fungsi untuk menghitung median dari array integer
- `sort.Ints(arr)` → mengurutkan array dalam urutan naik
- `n := len(arr)` → mendapatkan Panjang array
- `if n%2 == 0 {` → jika jumlah elemen array genap
- `return (arr[n/2-1] + arr[n/2]) / 2` → mengembalikan rata-rata dua elemen tengah
- `return arr[n/2]` → jika jumlah elemen array ganjil, mengembalikan elemen tengah
- `func main() {` → merupakan fungsi utama
- `var num int` → deklarasi variabel num bertipe data integer
- `var data []int` → deklarasi slice data untuk menyimpan kumpulan angka yang diinput
- `for {` → perulangan hingga kondisi break terpenuhi
- `fmt.Scan(&num)` → membaca input angka dari pengguna
- `if num == -5313 {` → jika variabel input sama dengan -5313

- break → keluar dari perulangan
- if num == 0 { → jika input sama dengan 0
- if len(data) > 0 { → jika slice data tidak kosong
- median := hitungMedian(data) → menghitung median dari data
- fmt.Println(median) → menampilkan nilai median
- } else { → kalau tidak
- data = append(data, num) menambahkan angka ke slice data

3. 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.}
```

## Source Code

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "sort"
    "strconv"
    "strings"
)

const nMax = 7019

type Buku struct {
    id, judul, penulis, penerbit string
```

```

        eksemplar, tahun, rating      int
    }

type DaftarBuku [nMax]Buku

var pustaka DaftarBuku

func daftarkanBuku(pustaka *DaftarBuku, n
    int) {
    scanner := bufio.NewScanner(os.Stdin)
    for i := 0; i < n; i++ {
        scanner.Scan()
        data :=
            strings.Split(scanner.Text(), ",")
        for j := range data {
            data[j] =
                strings.TrimSpace(data[j])
        }
        pustaka[i] = Buku{
            id:      data[0],
            judul:   data[1],
            penulis: data[2],
            penerbit: data[3],
            eksemplar: func() int {
                val, _ :=
                    strconv.Atoi(data[4])
                return val
            }(),
            tahun: func() int {
                val, err :=

```

```

        strconv.Atoi(data[5])
            if err != nil {
                fmt.Println("Error
parsing year:", err)
            }
            return val
        }(),
        rating: func() int {
            val, _ :=
strconv.Atoi(data[6])
            return val
        }(),
    }
}

func cetakTerfavorit(pustaka *DaftarBuku, n
int) {
    var maxRating = -1
    var favoritIndeks = -1
    for i := 0; i < n; i++ {
        if pustaka[i].rating > maxRating {
            maxRating = pustaka[i].rating
            favoritIndeks = i
        }
    }
    fmt.Printf("Buku Terfavorit: %s, %s,
%s, %d\n",
pustaka[favoritIndeks].judul,
pustaka[favoritIndeks].penulis,

```



```

        pustaka[favoritIndeks].penerbit,
        pustaka[favoritIndeks].tahun)
    }

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

func cetak5Terbaru(pustaka *DaftarBuku, n
    int) {
    batas := 5
    if n < batas {
        batas = n
    }
    fmt.Println("\n5 Judul Buku dengan
        rating tertinggi:")
    for i := 0; i < batas; i++ {
        fmt.Printf("%d. %s (%d)\n", i+1,
            pustaka[i].judul, pustaka[i].rating)
    }
}

func cariBuku(pustaka *DaftarBuku, n int,
    rating int) {
    ketemu := false

```

```

no := 1
for i := 0; i < n; i++ {
    if pustaka[i].rating == rating {
        fmt.Printf("%d. %s, %s, %s, %d,
%d, %d\n", no, pustaka[i].judul,
pustaka[i].penulis,
pustaka[i].penerbit,
pustaka[i].eksemplar,
pustaka[i].tahun, pustaka[i].rating)
        ketemu = true
        no++
    }
}
if !ketemu {
    fmt.Println("Tidak ada buku dengan
rating seperti itu")
}
}

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Print("Masukkan jumlah buku: ")
    scanner.Scan()
    N, _ := strconv.Atoi(scanner.Text())

    fmt.Println("Masukkan data buku (id,
    judul, penulis, penerbit, eksemplar,
    tahun, rating):")
    daftarkanBuku(&pustaka, N)
}

```

```

urutBuku(&pustaka, N)

cetakTerfavorit(&pustaka, N)

cetak5Terbaru(&pustaka, N)


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

scanner.Scan()

ratingCari, _ :=
    strconv.Atoi(scanner.Text())

cariBuku(&pustaka, N, ratingCari)
}

```

## Screenshots Output

```

PS C:\Semester3Go\Praktikum\Modul 12\Unguided\No.3> go run Unguided3.go
Masukkan jumlah buku: 8
Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):
1, Laskar Pelangi, Andrea Hirata, Bentang Pustaka, 10, 2005, 7
2, Bumi Manusia, Pramoedya Ananta Toer, Hasta Mitra, 15, 1980, 8
3, Harry Potter and the Philosopher's Stone, J.K. Rowling, Bloomsbury, 12, 1997, 10
4, Negeri 5 Menara, Ahmad Fuadi, Gramedia Pustaka Utama, 7, 2009, 9
5, Da Vinci Code, Dan Brown, Doubleday, 8, 2003, 8
6, Sang Pemimpi, Andrea Hirata, Bentang Pustaka, 10, 2006, 7
7, The Hobbit, J.R.R. Tolkien, George Allen & Unwin, 12, 1937, 9
8, Petualangan Sherlock Holmes, Arthur Conan Doyle, Gramedia Pustaka Utama, 9, 1892, 7
Buku Terfavorit: Harry Potter and the Philosopher's Stone, J.K. Rowling, Bloomsbury, 1997

5 Judul Buku dengan rating tertinggi:
1. Harry Potter and the Philosopher's Stone (10)
2. Negeri 5 Menara (9)
3. The Hobbit (9)
4. Bumi Manusia (8)
5. Da Vinci Code (8)

Masukkan rating buku yang dicari: 7
1. Laskar Pelangi, Andrea Hirata, Bentang Pustaka, 10, 2005, 7
2. Sang Pemimpi, Andrea Hirata, Bentang Pustaka, 10, 2006, 7
3. Petualangan Sherlock Holmes, Arthur Conan Doyle, Gramedia Pustaka Utama, 9, 1892, 7
PS C:\Semester3Go\Praktikum\Modul 12\Unguided\No.3>

```

## Deskripsi:

Program tersebut merupakan program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Program tersebut meminta pengguna menginput id, judul, penulis, penerbit, eksemplar, tahun, rating. Program tersebut menampilkan Buku Terfavorit, 5 Judul Buku dengan rating tertinggi, dan menampilkan daftar buku dari rating yang diinputkan.

- `package main` → paket utama pada program golang
- `import` → mengimpor
- `"fmt"` → import fmt
- `"bufio"` → import bufio
- `"os"` → import os
- `"sort"` → import sort
- `"strconv"` → import strconv
- `"strings"` → import strings
- `const nMax = 7019` → deklarasi const nMax dengan nilai maksimum 7019
- `type Buku struct {` → mendefinisikan tipe data buku
- `id, judul, penulis, penerbit string` → deklarasi atribut dengan tipe data string
- `eksemplar, tahun, rating int` → deklarasi atribut dengan tipe data integer
- `type DaftarBuku [nMax]Buku` → array buku dengan jumlah maksimum nMax
- `var pustaka DaftarBuku` → deklarasi variabel Pustaka bertipe data DaftarBuku
- `func daftarkanBuku(pustaka *DaftarBuku, n int) {` → fungsi daftarkan buku
- `scanner := bufio.NewScanner(os.Stdin)` → membuat scanner pada input
- `for i := 0; i < n; i++ {` → perulangan sebanyak jumlah buku yang didaftarkan

- `scanner.Scan()` → membaca input
- `data := strings.Split(scanner.Text(), ",")` → memisahkan input berdasarkan koma
- `for j := range data {` → menghilangkan spasi setiap elemen data
- `data[j] = strings.TrimSpace(data[j])` → `data[j] = strings.TrimSpace(data[j])`
- `pustaka[i] = Buku{` → mengisi data buku ke dalam pustaka
- `id: data[0],` → mengisi ID buku
- `judul: data[1],` → mengisi judul buku
- `penulis: data[2],` → mengisi penulis buku
- `penerbit: data[3],` → mengisi penerbit buku
- `eksemplar: func() int {` → konversi string ke integer untuk eksemplar dan mengisinya
- `val, _ := strconv.Atoi(data[4])` → `val, _ := strconv.Atoi(data[4])`
- `return val` → return val
- `tahun: func() int {` → konversi string ke integer untuk tahun dan mengisinya
- `val, err := strconv.Atoi(data[5])` → `val, err := strconv.Atoi(data[5])`
- `if err != nil {` → jika err tidak sama dengan kosong
- `fmt.Println("Error parsing year:", err)` → menampilkan statement Error parsing year:", err
- `return val` → return val
- `rating: func() int {` → konversi string ke integer untuk rating dan

mengisinya

- `val, _ := strconv.Atoi(data[6])` → `val, _ := strconv.Atoi(data[6])`
- `return val` → `return val`
- `func cetakTerfavorit(pustaka *DaftarBuku, n int) {` → fungsi untuk mencetak buku terfavorit berdasarkan rating tertinggi
- `var maxRating = -1` → inisialisasi `maxRating` dengan -1 untuk perbandingan awal
- `var favoritIndeks = -1` → inisialisasi favorit indeks dengan -1 sebagai nilai awal
- `for i := 0; i < n; i++ {` → perulangan untuk memeriksa semua buku
- `if pustaka[i].rating > maxRating {` → jika rating buku lebih besar dari `maxRating` saat ini
- `maxRating = pustaka[i].rating` → update rating buku lebih besar dari `maxRating` saat ini
- `favoritIndeks = i` → update favorit indeks dengan indeks buku saat ini
- `fmt.Printf("\nBuku Terfavorit: %s, %s, %s, %d\n",  
pustaka[favoritIndeks].judul, pustaka[favoritIndeks].penulis,  
pustaka[favoritIndeks].penerbit, pustaka[favoritIndeks].tahun)`  
→ menampilkan buku terfavorit
- `func urutBuku(pustaka *DaftarBuku, n int) {` → fungsi untuk mengurutkan buku
- `sort.Slice(pustaka[:n], func(i, j int) bool {` → mengurutkan slice buku dari pustaka
- `return pustaka[i].rating > pustaka[j].rating` → urutkan

berdasarkan rating secara descending

- `func cetak5Terbaru(pustaka *DaftarBuku, n int) {` → fungsi untuk mencetak 5 buku dengan rating tertinggi
- `batas := 5` → inisialisasi batas sama dengan 5
- `if n < batas {` → jika jumlah buku kurang dari 5
- `batas = n` → update batas dengan jumlah buku yang ada
- `fmt.Println("\n5 Judul Buku dengan rating tertinggi:")` → menampilkan statement 5 Judul Buku dengan rating tertinggi:
- `for i := 0; i < batas; i++ {` → perulangan untuk mencetak 5 buku
- `fmt.Printf("%d. %s (%d)\n", i+1, pustaka[i].judul, pustaka[i].rating)` → menampilkan judul buku beserta rating
- `func cariBuku(pustaka *DaftarBuku, n int, rating int) {` → fungsi untuk mencari buku berdasarkan rating tertentu
- `ketemu := false` → inisialisasi variabel ketemu sama dengan false
- `no := 1` → inisialisasi nomor pencarian dengan 1
- `for i := 0; i < n; i++ {` → perulangan for untuk memeriksa semua buku
- `if pustaka[i].rating == rating {` → jika rating buku sesuai dengan yang dicari
- `fmt.Printf("%d. %s, %s, %s, %d, %d, %d\n", no, pustaka[i].judul, pustaka[i].penulis, pustaka[i].penerbit, pustaka[i].eksemplar, pustaka[i].tahun, pustaka[i].rating)` → menampilkan informasi buku dengan penomoran
- `ketemu = true` → update variabel ketemu menjadi true

- `no++` → `no++(increment)`
- `if !ketemu {` → jika tidak ada buku tidak ketemu
- `fmt.Println("Tidak ada buku dengan rating seperti itu")` → menampilkan statement Tidak ada buku dengan rating seperti itu
- `func main() {` → merupakan fungsi utama
- `scanner := bufio.NewScanner(os.Stdin)` → membuat scanner baru untuk input dari pengguna
- `fmt.Print("Masukkan jumlah buku: ")` → menampilkan statement Masukkan jumlah buku
- `scanner.Scan()` → membaca input pengguna
- `N, _ := strconv.Atoi(scanner.Text())` → konversi input string ke integer
- `fmt.Println("Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating):")` → menampilkan statement untuk Masukkan data buku (id, judul, penulis, penerbit, eksemplar, tahun, rating)
- `daftarkanBuku(&pustaka, N)` → memanggil fungsi untuk mendaftarkan buku ke dalam pustaka
- `urutBuku(&pustaka, N)` →
- `cetakTerfavorit(&pustaka, N)` → memanggil fungsi untuk mengurutkan buku berdasarkan rating
- `cetak5Terbaru(&pustaka, N)` → memanggil fungsi untuk menampilkan 5 buku dengan rating tertinggi
- `fmt.Print("\nMasukkan rating buku yang dicari: ")` → menampilkan statement untuk nMasukkan rating buku yang dicari:



- `scanner.Scan()` ➔ membaca input dari pengguna
- `ratingCari, _ := strconv.Atoi(scanner.Text())` ➔ konversi input ke string ke integer
- `cariBuku(&pustaka, N, ratingCari)` ➔ memanggil fungsi untuk memanggil buku berdasarkan rating yang dicari