

**LAPORAN PRAKTIKUM
ALGORITMA PEMROGRAMAN 2**

**MODUL XII & XIII
PENGURUTAN DATA**



Oleh:

Mansyuroh

NIM:

2311102234

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

MODUL 12

I. DASAR TEORI

A. Ide Pencarian 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	}

B. Algoritma Selection Sort

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

dalam bahasa Go 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
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] > T[j] {
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     }
```

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

```
.. ...
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 }
```

II. 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()

    }

}

```

Screenshot program

```
PS C:\Users\umans> go run "c:\Semester 3\guided1modul12.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5 2 1 7 9 13
Masukkan nomor rumah kerabat untuk daerah ke-1: Hasil urutan rumah untuk daerah ke-1: 13 9 7 2 1
Masukkan jumlah rumah kerabat untuk daerah ke-2: 6 189 15 27 39 75 133
Masukkan nomor rumah kerabat untuk daerah ke-2: Hasil urutan rumah untuk daerah ke-2: 189 133 75 39 27 15
Masukkan jumlah rumah kerabat untuk daerah ke-3: 3 4 9 1
Masukkan nomor rumah kerabat untuk daerah ke-3: Hasil urutan rumah untuk daerah ke-3: 9 4 1
PS C:\Users\umans>
```

Deskripsi program

Program ini menerima input jumlah daerah dan jumlah rumah di setiap daerah, lalu mengurutkan nomor rumah setiap daerah secara menurun menggunakan algoritma selection sort. Input diverifikasi agar memenuhi batas tertentu, dan hasil pengurutan ditampilkan per daerah.

Input : Jumlah daerah (n) dan rumah (m) dengan validasi.

Proses : Urutkan nomor rumah menggunakan selection sort.

Output : Tampilkan daftar rumah yang sudah diurutkan per daerah.

III. UNGUIDED

1. Source code

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array secara ascending menggunakan selection sort
func selectionSort(arr []int, ascending bool) {
    n := len(arr)
```

```

    for i := 0; i < n-1; i++ {
        selectedIdx := i
        for j := i + 1; j < n; j++ {
            if (ascending && arr[j] < arr[selectedIdx]) || (!ascending &&
arr[j] > arr[selectedIdx]) {
                selectedIdx = j
            }
        }
        // Tukar elemen
        arr[i], arr[selectedIdx] = arr[selectedIdx], arr[i]
    }
}

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("\nMasukkan jumlah rumah untuk daerah ke-%d: ", i+1)
        fmt.Scan(&m)

        if m <= 0 {
            fmt.Println("Jumlah rumah harus lebih besar dari 0.")
            return
        }

        // Membaca array angka sebagai input
        fmt.Printf("Masukkan nomor rumah untuk daerah ke-%d:\n", i+1)

```

```

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

// Pisahkan bilangan ganjil dan genap
ganjil := make([]int, 0)
genap := make([]int, 0)
for _, num := range houses {
    if num%2 == 0 {
        genap = append(genap, num)
    } else {
        ganjil = append(ganjil, num)
    }
}

// Urutkan ganjil (ascending) dan genap (ascending)
selectionSort(ganjil, true) // Urutkan ascending (Ganjil)
selectionSort(genap, true)  // Urutkan ascending (Genap)

campuran := append(ganjil, genap...)

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

// Baris pertama: Ganjil saja
fmt.Print("Ganjil terurut: ")
for _, num := range ganjil {
    fmt.Printf("%d ", num)
}
fmt.Println()

fmt.Print("Campuran terurut: ")
for _, num := range campuran {

```



```

        fmt.Printf("%d ", num)
    }
    fmt.Println()

    fmt.Print("Genap terurut: ")
    for _, num := range genap {
        fmt.Printf("%d ", num)
    }
    fmt.Println()
}
}

```

Screenshot program

```

PS C:\Users\umans> go run "c:\Semester 3\unguided1modul12.go"
Masukkan jumlah daerah (n): 3

```

```

Masukkan jumlah rumah untuk daerah ke-1: 6
Masukkan nomor rumah untuk daerah ke-1:
5 2 1 7 9 13
Hasil urutan rumah untuk daerah ke-1:
Ganjil terurut: 1 5 7 9 13
Campuran terurut: 1 5 7 9 13 2
Genap terurut: 2

```

```

Masukkan jumlah rumah untuk daerah ke-2: 7
Masukkan nomor rumah untuk daerah ke-2:
6 189 15 27 39 75 133
Hasil urutan rumah untuk daerah ke-2:
Ganjil terurut: 15 27 39 75 133 189
Campuran terurut: 15 27 39 75 133 189 6
Genap terurut: 6

```

```

Masukkan jumlah rumah untuk daerah ke-3: 4
Masukkan nomor rumah untuk daerah ke-3:
3 4 9 1
Hasil urutan rumah untuk daerah ke-3:
Ganjil terurut: 1 3 9
Campuran terurut: 1 3 9 4
Genap terurut: 4
PS C:\Users\umans>

```

Program ini mengurutkan nomor rumah berdasarkan kategori ganjil dan genap. Untuk setiap daerah, nomor rumah dipisahkan menjadi dua kelompok: ganjil dan genap. Nomor ganjil diurutkan secara ascending, sedangkan genap juga diurutkan secara ascending. Hasil output terdiri dari tiga baris:

Input: Jumlah daerah (n) dan nomor rumah untuk setiap daerah.

Proses: Pisahkan nomor rumah menjadi ganjil dan genap. Urutkan nomor ganjil dan genap secara ascending.

Output: Urutan berdasarkan descending atau ascending

2. Source code

```
package main

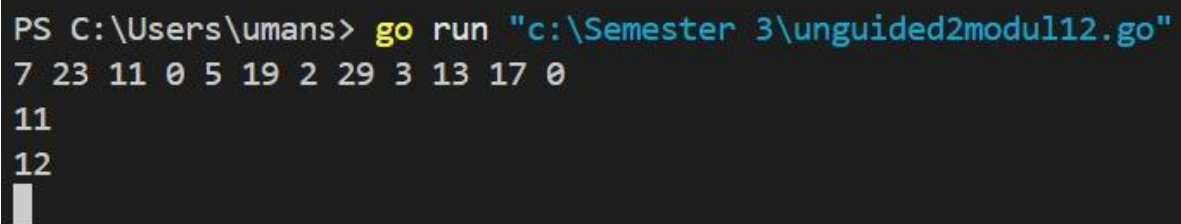
import "fmt"

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

// Fungsi untuk melakukan insertion sort
func insertionSort(arr []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 = j - 1
        }
        arr[j+1] = key
    }
}
```

```
func main() {  
    var data []int  
    var input int  
  
    for {  
        fmt.Scan(&input)  
        if input == -5313 {  
            break  
        }  
        if input == 0 {  
            insertionSort(data)  
            median := hitungMedian(data)  
            fmt.Println(median)  
        } else {  
            data = append(data, input)  
        }  
    }  
}
```

Screenshot program



```
PS C:\Users\umans> go run "c:\Semester 3\unguided2modul12.go"  
7 23 11 0 5 19 2 29 3 13 17 0  
11  
12  
█
```

Deskripsi program

Program ini menggunakan algoritma insertion sort untuk mengurutkan nomor rumah yang dimasukkan oleh pengguna. Setelah pengurutan, program menghitung median dari nomor rumah yang

telah terurut. Median dihitung berdasarkan jumlah elemen dalam array: jika jumlah elemen ganjil, median adalah elemen tengah, sementara jika jumlah elemen genap, median adalah rata-rata dari dua elemen tengah.

Input: Program menerima input angka satu per satu hingga angka -5313 dimasukkan, yang menandakan akhir input. Jika input adalah 0, program akan mengurutkan data yang telah dimasukkan dan menghitung median dari data tersebut.

Proses: Program mengurutkan data menggunakan insertion sort dan menghitung median setelah setiap input 0. Median dihitung berdasarkan jumlah elemen: jika ganjil, median adalah elemen tengah; jika genap, median adalah rata-rata dua elemen tengah.

Output: Program menampilkan median dari data yang telah diurutkan setiap kali input 0 dimasukkan

MODUL 13

I. DASAR TEORI

A. 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.

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	}

B. Algoritma Insertion Sort

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

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. 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

```

```

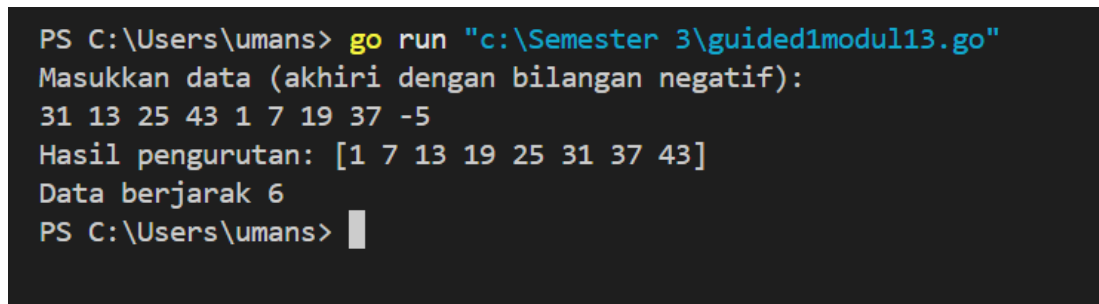
        isConsistent, diff := isDataConsistentlySpaced(data)

        // Cetak hasil
        fmt.Println("Hasil pengurutan:", data)

        if isConsistent {
            fmt.Printf("Data berjarak %d\n", diff)
        } else {
            fmt.Println("Data berjarak tidak tetap")
        }
    }
}

```

Screenshot program



```

PS C:\Users\umans> go run "c:\Semester 3\guided1modul13.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:\Users\umans>

```

Deskripsi program

Program ini meminta pengguna memasukkan angka, diakhiri dengan angka negatif. Data diurutkan menggunakan insertion sort, lalu diperiksa apakah jarak antar angka tetap sama. Output menunjukkan daftar angka terurut dan informasi tentang konsistensi jarak antar elemen.

Input : Pengguna memasukkan angka satu per satu, dan proses berhenti saat angka negatif dimasukkan.

Proses : Data yang dimasukkan diurutkan menggunakan algoritma insertion sort. Program memeriksa apakah jarak antar angka berurutan tetap konsisten setelah pengurutan.

Output : Menampilkan daftar angka yang sudah diurutkan. Memberikan informasi apakah jarak antar elemen tetap sama, dan jika ya, mencetak nilai selisihnya.

III. UNGUIDED

1. Source code


```
package main

import (
    "fmt"
)

const nMax = 7919

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

type DaftarBuku = [nMax]Buku

// Fungsi untuk mendaftarkan buku
func DaftarkanBuku(pustaka *DaftarBuku, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Buku %v: \n", i+1)
        fmt.Print("Masukkan id buku: ")
        fmt.Scan(&pustaka[i].id)
        fmt.Print("Masukkan judul buku: ")
        fmt.Scan(&pustaka[i].judul)
        fmt.Print("Masukkan penulis buku: ")
        fmt.Scan(&pustaka[i].penulis)
        fmt.Print("Masukkan penerbit buku: ")
        fmt.Scan(&pustaka[i].penerbit)
        fmt.Print("Masukkan eksemplar buku: ")
        fmt.Scan(&pustaka[i].eksemplar)
        fmt.Print("Masukkan tahun terbit: ")
        fmt.Scan(&pustaka[i].tahun)
```

```

        fmt.Print("Masukkan rating buku: ")

        fmt.Scan(&pustaka[i].rating)

        fmt.Println()

    }
}

// Fungsi untuk mencetak buku dengan rating tertinggi (favorit)
func CetakTerfavorit(pustaka DaftarBuku, n int) {
    max := pustaka[0]
    for i := 1; i < n; i++ {
        if pustaka[i].rating > max.rating {
            max = pustaka[i]
        }
    }

    fmt.Println("Buku Favorit: ")
    fmt.Printf("Judul buku: %v\n", max.judul)
    fmt.Printf("Penulis buku: %v\n", max.penulis)
    fmt.Printf("Penerbit buku: %v\n", max.penerbit)
    fmt.Println()
}

// Fungsi untuk mengurutkan buku berdasarkan rating
func UrutBuku(pustaka DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := pustaka[i]
        j := i - 1
        for j >= 0 && pustaka[j].rating > key.rating {
            pustaka[j+1] = pustaka[j]
            j--
        }
        pustaka[j+1] = key
    }
}

```

```

// Fungsi untuk mencetak 5 buku terbaru dengan rating tertinggi
func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    // Urutkan berdasarkan rating tertinggi terlebih dahulu
    UrutBuku(pustaka, n)

    // Ambil 5 buku dengan rating tertinggi
    fmt.Println("5 Judul Buku dengan Rating Tertinggi: ")
    limit := 5
    if n < limit {
        limit = n
    }
    for i := 0; i < limit; i++ {
        fmt.Printf("Buku %v: \n", i+1)
        fmt.Printf("Judul buku: %v\n", pustaka[i].judul)
        fmt.Println()
    }
}

// Fungsi untuk mencari buku berdasarkan rating
func CariBuku(pustaka DaftarBuku, n, r int) {
    fmt.Print("Masukkan rating untuk mencari buku: ")
    fmt.Scan(&r)

    low, high := 0, n-1
    found := false
    for low <= high {
        mid := (low + high) / 2
        if pustaka[mid].rating == r {
            // Menampilkan buku dengan rating tersebut
            fmt.Println("Buku dengan rating", r, ":")
            i := mid
            for i >= 0 && pustaka[i].rating == r {
                fmt.Printf("Buku %v: \n", i+1)
                fmt.Printf("Judul buku: %v\n", pustaka[i].judul)
            }
        }
    }
}

```

```

        fmt.Printf("Penulis buku: %v\n", pustaka[i].penulis)
        fmt.Printf("Penerbit buku: %v\n", pustaka[i].penerbit)
        fmt.Printf("Tahun terbit: %v\n", pustaka[i].tahun)
        fmt.Println()
        i--
    }

    i = mid + 1
    for i < n && pustaka[i].rating == r {
        fmt.Printf("Buku %v: \n", i+1)
        fmt.Printf("Judul buku: %v\n", pustaka[i].judul)
        fmt.Printf("Penulis buku: %v\n", pustaka[i].penulis)
        fmt.Printf("Penerbit buku: %v\n", pustaka[i].penerbit)
        fmt.Printf("Tahun terbit: %v\n", pustaka[i].tahun)
        fmt.Println()
        i++
    }

    found = true
    break
} else if pustaka[mid].rating < r {
    low = mid + 1
} else {
    high = mid - 1
}

}

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

}

func main() {
    var pustaka DaftarBuku
    var Npustaka, rating int

```

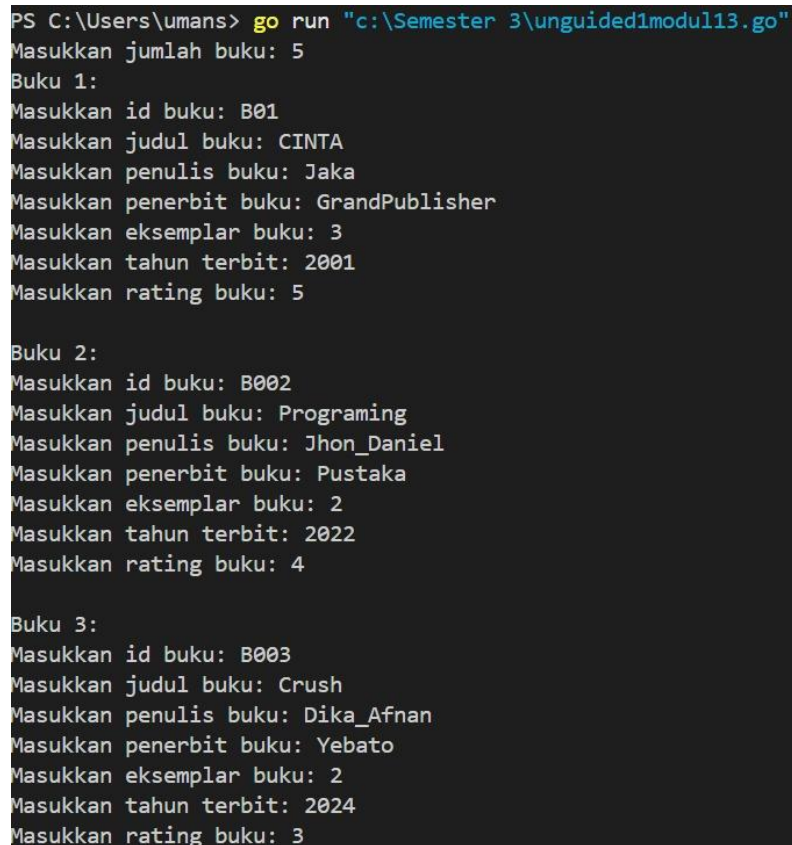
```
// Input jumlah buku

fmt.Print("Masukkan jumlah buku: ")

fmt.Scan(&Npustaka)


// Menambahkan buku-buku ke pustaka
DaftarkanBuku(&pustaka, Npustaka)
CetakTerfavorit(pustaka, Npustaka)
UrutBuku(pustaka, Npustaka)
Cetak5Terbaru(pustaka, Npustaka)
CariBuku(pustaka, Npustaka, rating)
}
```

Screenshot program



```
PS C:\Users\umans> go run "c:\Semester 3\unguided1modul13.go"
Masukkan jumlah buku: 5
Buku 1:
Masukkan id buku: B01
Masukkan judul buku: CINTA
Masukkan penulis buku: Jaka
Masukkan penerbit buku: GrandPublisher
Masukkan eksemplar buku: 3
Masukkan tahun terbit: 2001
Masukkan rating buku: 5

Buku 2:
Masukkan id buku: B002
Masukkan judul buku: Programing
Masukkan penulis buku: Jhon_Daniel
Masukkan penerbit buku: Pustaka
Masukkan eksemplar buku: 2
Masukkan tahun terbit: 2022
Masukkan rating buku: 4

Buku 3:
Masukkan id buku: B003
Masukkan judul buku: Crush
Masukkan penulis buku: Dika_Afnan
Masukkan penerbit buku: Yebato
Masukkan eksemplar buku: 2
Masukkan tahun terbit: 2024
Masukkan rating buku: 3
```

```
Masukkan judul buku: Putik
Masukkan penulis buku: Yomarakui
Masukkan penerbit buku: Atlan_Jaya
Masukkan eksemplar buku: 2
Masukkan tahun terbit: 2018
Masukkan rating buku: 4

Buku Favorit:
Judul buku: CINTA
Penulis buku: Jaka
Penerbit buku: GrandPublisher

5 Judul Buku dengan Rating Tertinggi:
Buku 1:
Judul buku: CINTA

Buku 2:
Judul buku: Programing

Buku 3:
Judul buku: Crush

Buku 4:
Judul buku: Aku

Buku 5:
Judul buku: Putik
```

Deskripsi program

Program ini digunakan untuk mengelola data buku. Pengguna dapat memasukkan informasi buku seperti id, judul, penulis, penerbit, eksemplar, tahun terbit, dan rating. Program kemudian akan menampilkan buku dengan rating tertinggi sebagai buku favorit, mengurutkan buku berdasarkan rating tertinggi, mencetak 5 buku dengan rating tertinggi, dan memungkinkan pencarian buku berdasarkan rating tertentu.

Input: Jumlah buku yang ingin didaftarkan. Untuk setiap buku, inputkan: ID buku, judul buku, penulis buku, penerbit buku, Jumlah eksemplar buku, tahun terbit, rating buku.

Proses: DaftarkanBuku: Menyimpan data buku yang dimasukkan oleh pengguna.

CetakTerfavorit: Menentukan dan menampilkan buku dengan rating tertinggi.

UrutBuku: Mengurutkan buku berdasarkan rating dari yang tertinggi.

Cetak5Terbaru: Menampilkan 5 buku teratas berdasarkan rating tertinggi.

CariBuku: Mencari buku berdasarkan rating yang dimasukkan oleh pengguna.

Output: Buku favorit (rating tertinggi).

5 buku dengan rating tertinggi.

Buku yang sesuai dengan rating yang dicari.