

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 12  
PENGURUTAN DATA**



**Universitas  
Telkom**

Oleh:

**HENDWI SAPUTRA**

2311102218

IF-11-02

**S1 TEKNIK INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. 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".

- 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	}

## 12.5 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
}
```

## II. GUIDED

### GUIDED I

#### 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)
```

```

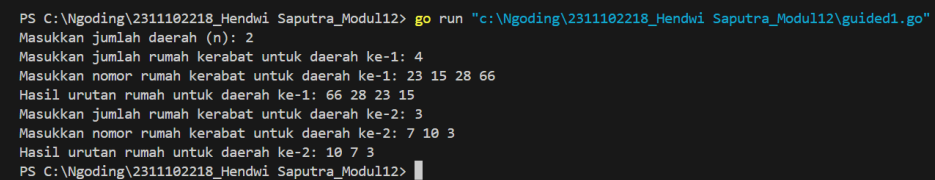
        for j := 0; j < m; j++ {
            fmt.Scan(&houses[j])
        }

        // 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



```

PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul12\guided1.go"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 4
Masukkan nomor rumah kerabat untuk daerah ke-1: 23 15 28 66
Hasil urutan rumah untuk daerah ke-1: 66 28 23 15
Masukkan jumlah rumah kerabat untuk daerah ke-2: 3
Masukkan nomor rumah kerabat untuk daerah ke-2: 7 10 3
Hasil urutan rumah untuk daerah ke-2: 10 7 3
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12>

```

## Deskripsi:

Program ini mengurutkan nomor-nomor rumah kerabat di berbagai daerah menggunakan algoritma Selection Sort. Program dimulai dengan meminta pengguna memasukkan jumlah daerah (n) yang harus berada dalam rentang 1 hingga 999. Untuk setiap daerah, program akan meminta jumlah rumah kerabat (m) yang harus berada dalam rentang 1 hingga 999999. Setelah itu, pengguna diminta untuk memasukkan nomor-nomor rumah kerabat untuk daerah tersebut. Program menggunakan fungsi selectionSort untuk mengurutkan nomor-nomor rumah secara descending (dari besar ke kecil), di mana algoritma ini bekerja dengan mencari elemen terbesar dalam array dan menukarnya dengan posisi yang sedang diproses. Setelah pengurutan selesai, program akan menampilkan hasil urutan nomor rumah untuk setiap daerah. Program ini memiliki validasi input untuk memastikan nilai n dan

m berada dalam rentang yang ditentukan, dan menggunakan slice (array dinamis) untuk menyimpan nomor-nomor rumah.

## GUIDED II

### 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

```

PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul12\Guided2\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
2
5
8
11
14
-1
Hasil pengurutan: [2 5 8 11 14]
Data berjarak 3
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12>

```

```

PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul12\Guided2\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
4
9
12
20
25
01
-1
Hasil pengurutan: [1 4 9 12 20 25]
Data berjarak tidak tetap
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12>

```

**Deskripsi:**

Program ini menganalisis serangkaian data numerik dan menentukan apakah data tersebut memiliki jarak yang konsisten antara setiap elemennya. Program menggunakan dua fungsi utama: `insertionSort` untuk mengurutkan data dan `isDataConsistentlySpaced` untuk memeriksa konsistensi jarak antar data. Pada awal eksekusi, program meminta pengguna memasukkan sejumlah bilangan, dengan input bilangan negatif sebagai penanda akhir input. Data yang dimasukkan kemudian diurutkan menggunakan algoritma Insertion Sort, di mana setiap elemen dibandingkan dengan elemen sebelumnya dan disisipkan ke posisi yang tepat. Setelah pengurutan, program memeriksa apakah selisih antara setiap elemen berurutan konsisten menggunakan fungsi `isDataConsistentlySpaced`. Fungsi ini menghitung selisih absolut antara elemen berurutan dan membandingkan selisih tersebut dengan selisih awal. Program menggunakan package `math` untuk menghitung nilai absolut dari selisih antar elemen. Hasil analisis kemudian ditampilkan, menunjukkan data yang telah diurutkan dan apakah data tersebut memiliki jarak yang tetap beserta nilai jaraknya, atau memberitahu jika data memiliki jarak yang tidak konsisten.

### III. UNGUIDED

#### UNGUIDED I

##### Source Code

```
package main

import "fmt"

func selectionSort_Ganjil(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] {
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i]
    }
}

func selectionSort_Genap(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] {
                maxIdx = j
            }
        }
        arr[i], arr[maxIdx] = arr[maxIdx], arr[i]
    }
}

func main() {
    var n, input 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
    }

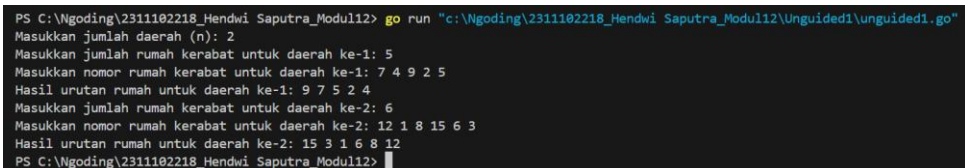
    nomerGanjil := make([]int, 0)
    nomerGenap := make([]int, 0)
    fmt.Printf("Masukkan nomor rumah kerabat untuk
daerah ke-%d: ", i+1)
    for j := 0; j < m; j++ {
        fmt.Scan(&input)
        if input%2 == 0 {
            nomerGenap = append(nomerGenap, input)
        } else {
            nomerGanjil = append(nomerGanjil, input)
        }
    }

    selectionSort_Ganjil(nomerGanjil)
    selectionSort_Genap(nomerGenap)

    fmt.Printf("Hasil urutan rumah untuk daerah ke-%d:
", i+1)
    for _, rumah := range nomerGanjil {
        fmt.Printf("%d ", rumah)
    }
    for _, rumah := range nomerGenap {
        fmt.Printf("%d ", rumah)
    }
    fmt.Println()
}
}

```

## Screenshot



```

PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul12\Unguided1\unguided1.go"
Masukkan jumlah daerah (n): 2
Masukkan jumlah rumah kerabat untuk daerah ke-1: 5
Masukkan nomor rumah kerabat untuk daerah ke-1: 7 4 9 2 5
Hasil urutan rumah untuk daerah ke-1: 9 7 5 2 4
Masukkan jumlah rumah kerabat untuk daerah ke-2: 6
Masukkan nomor rumah kerabat untuk daerah ke-2: 12 1 8 15 6 3
Hasil urutan rumah untuk daerah ke-2: 15 3 1 6 8 12
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12>

```

## Deskripsi:

Program ini bertujuan untuk mengurutkan nomor rumah kerabat dalam beberapa daerah berdasarkan nomor ganjil dan genap menggunakan algoritma selection sort. Pertama, pengguna diminta memasukkan jumlah daerah, kemudian untuk setiap daerah, pengguna memasukkan jumlah rumah kerabat dan nomor rumah tersebut. Nomor rumah dipisahkan menjadi dua kelompok: ganjil dan genap. Nomor rumah ganjil diurutkan

dalam urutan menurun, sementara nomor rumah genap diurutkan dalam urutan menaik. Setelah diurutkan, program mencetak hasil urutan nomor rumah untuk masing-masing daerah.

## UNGUIDED II

### Source Code

```
package main

import "fmt"

func insertion_Sort(arr []int) {
    n := len(arr)
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1

        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

func mencariMedian(arr []int) int {
    n := len(arr)
    if n%2 == 0 {
        return (arr[(n/2)-1] + arr[(n/2)]) / 2
    } else {
        return arr[(n / 2)]
    }
}

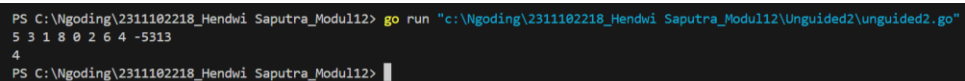
func main() {
    var input int
    var kumpulan_Angka = make([]int, 0)
    var angka_Sliced = make([]int, 0)
    for input != -5313 {
        fmt.Scan(&input)
        if input != -5313 {
            kumpulan_Angka = append(kumpulan_Angka, input)
        }
    }
    for _, angka := range kumpulan_Angka {
        if angka == 0 {
            insertion_Sort(angka_Sliced)
            fmt.Println(mencariMedian(angka_Sliced))
        }
    }
}
```

```

    } else {
        angka_Sliced = append(angka_Sliced, angka)
    }
}
}

```

## Screenshot



```

PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul12\Unguided2\unguided2.go"
5 3 1 8 0 2 6 4 -5313
4
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12>

```

## Deskripsi:

Program ini bertujuan untuk menghitung median dari sekumpulan angka yang dimasukkan oleh pengguna, dengan menggunakan algoritma insertion sort untuk pengurutan. Pengguna dapat terus memasukkan angka hingga memasukkan angka -5313 untuk mengakhiri input. Setelah itu, program menyimpan angka yang dimasukkan dalam dua array terpisah: kumpulan\_Angka untuk semua angka yang dimasukkan, dan angka\_Sliced untuk subset angka yang perlu diurutkan dan dihitung median-nya. Setiap kali pengguna memasukkan angka 0, program akan mengurutkan angka-angka dalam angka\_Sliced dan mencetak nilai median dari subset tersebut.

## UNGUIDED III

### Source Code

```

package main

import "fmt"

const nmax = 7919

type Buku struct {
    ID, Judul, Penulis, Penerbit string
    Eksemplar, Tahun, Ranting    int
}
type DaftarBuku [nmax]Buku

func insertion_Sort(Pustaka *DaftarBuku, n int) {
    for i := 1; i < n; i++ {
        key := Pustaka[i]

```

```

        j := i - 1

        for j >= 0 && Pustaka[j].Ranting < key.Ranting {
            Pustaka[j+1] = Pustaka[j]
            j--
        }
        Pustaka[j+1] = key
    }
}

func cari_Favorit(Pustaka DaftarBuku, n int) {
    max := Pustaka[0].Ranting
    favorit := 0
    for i := 0; i < n; i++ {
        if Pustaka[i].Ranting > max {
            max = Pustaka[i].Ranting
            favorit = i
        }
    }
    fmt.Printf("Buku Terfavorit Adalah : %s %s %s %s %v\n", Pustaka[favorit].ID, Pustaka[favorit].Judul, Pustaka[favorit].Penulis, Pustaka[favorit].Penerbit, Pustaka[favorit].Tahun)
}

func LimaRating(Pustaka DaftarBuku, n int) {
    insertion_Sort(&Pustaka, n)
    fmt.Print("Lima Rating Tertinggi : ")
    for i := 0; i < n; i++ {
        fmt.Print(Pustaka[i].Judul, " ")
    }
    fmt.Println()
}

func binary_Search(Pustaka DaftarBuku, n, target int) int
{
    low, high := 0, n-1

    for low <= high {
        mid := (low + high) / 2

        if Pustaka[mid].Ranting == target {
            return mid
        } else if Pustaka[mid].Ranting < target {
            low = mid + 1
        } else {
            high = mid - 1
        }
    }

    return -1
}

func main() {
    var Buku DaftarBuku
    var n, Cari int

```

```

        fmt.Print("Masukan Banyak Buku : ")
        fmt.Scan(&n)
        fmt.Println("Masukan (ID, Judul, Penulis, Penerbit,
Eksemplar, Tahun, Rating)")
        for i := 0; i < n; i++ {
            fmt.Print("Masukan : ")
            fmt.Scan(&Buku[i].ID, &Buku[i].Judul,
&Buku[i].Penulis, &Buku[i].Penerbit, &Buku[i].Eksemplar,
&Buku[i].Tahun, &Buku[i].Ranting)
        }

        fmt.Print("Masukan Rating Buku Yang Anda Cari : ")
        fmt.Scan(&Cari)

        cari_Favorit(Buku, n)
        LimaRating(Buku, n)
        Temukan := binary_Search(Buku, n, Cari)
        if Temukan != -1 {
            fmt.Printf("Buku dengan Rating %v : %s %s %s %s %s %v
%v %v\n", Cari, Buku[Temukan].ID, Buku[Temukan].Judul,
Buku[Temukan].Penulis, Buku[Temukan].Penerbit,
Buku[Temukan].Eksemplar, Buku[Temukan].Tahun,
Buku[Temukan].Ranting)
        } else {
            fmt.Printf("Buku dengan Reting %v tidak
ditemukan", Cari)
        }
    }
}

```

## Screenshot

```

PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12> go run "c:\Ngoding\2311102218_Hendwi Saputra_Modul12\Unguided3\unguided3.go"
Masukan Banyak Buku : 5
Masukan (ID, Judul, Penulis, Penerbit, Eksemplar, Tahun, Rating)
Masukan : B001 SangPemimpin AhmadSukandi Gramedia 10 2020 95
Masukan : B002 TeoriKebahagiaan RamdanSyahputra Mizan 15 2019 88
Masukan : B003 ManajemenDiri DwiSaputra Republika 8 2021 92
Masukan : B004 PemikiranMaju AndiPrasetyo Erlangga 12 2018 85
Masukan : B005 InovasiTerkini RizkyNugraha Kompas 7 2022 90
Masukan Rating Buku Yang Anda Cari : 92
Buku Terfavorit Adalah : B001 SangPemimpin AhmadSukandi Gramedia 2020
Lima Rating Tertinggi : SangPemimpin ManajemenDiri InovasiTerkini TeoriKebahagiaan PemikiranMaju
Buku dengan Rating 92 : B003 ManajemenDiri DwiSaputra Republika 8 2021 92
PS C:\Ngoding\2311102218_Hendwi Saputra_Modul12>

```

## Deskripsi:

Program ini dirancang untuk mengelola dan mencari buku berdasarkan rating di dalam sebuah perpustakaan. Program meminta pengguna untuk memasukkan jumlah buku beserta detailnya seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun, dan rating. Ada beberapa fungsi yang digunakan: `insertion_Sort` untuk mengurutkan buku berdasarkan rating secara menurun, `cari_Favorit` untuk mencari buku dengan rating tertinggi,



LimaRating untuk menampilkan lima buku dengan rating tertinggi, dan binary\_Search untuk mencari buku berdasarkan rating yang dimasukkan oleh pengguna. Setelah mengurutkan dan menampilkan buku terfavorit serta lima buku dengan rating tertinggi, program akan mencari dan menampilkan buku dengan rating tertentu jika ditemukan, atau memberikan pesan bahwa buku dengan rating tersebut tidak ditemukan.