

**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK  
MODUL XII & XIII  
“PENGURUTAN DATA”**



**Oleh:**

**MUHAMMAD RAGIEL PRASTYO**

**2311102183**

**S1IF-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 Selectton 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 }

```

### 13.1 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	}

### 13.2 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. Source Code

```
// MUHAMMAD RAGIEL PRASTYO
// 2311102183
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 Output

```

PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul112\Guided\guided1.go"
Masukkan jumlah daerah (n): 3
Masukkan jumlah rumah kerabat untuk daerah ke-1: 4
Masukkan nomor rumah kerabat untuk daerah ke-1: 2 6 4 1
Hasil urutan rumah untuk daerah ke-1: 6 4 2 1
Masukkan jumlah rumah kerabat untuk daerah ke-2: 3
Masukkan nomor rumah kerabat untuk daerah ke-2: 3 9 5
Hasil urutan rumah untuk daerah ke-2: 9 5 3
Masukkan jumlah rumah kerabat untuk daerah ke-3: 5
Masukkan nomor rumah kerabat untuk daerah ke-3: 1 4 2 9 4
Hasil urutan rumah untuk daerah ke-3: 9 4 4 2 1
PS C:\Users\USER\OneDrive\Desktop\Alpro 2>

```

### Penjelasan:

Program diatas mengurutkan nomor rumah kerabat di beberapa daerah menggunakan *selection sort*. Pengguna diminta memasukkan jumlah daerah ( $n$ ) dan jumlah rumah kerabat ( $m$ ) di setiap daerah, diikuti oleh daftar nomor rumah. Input divalidasi agar  $n$  berada di antara 1-999 dan  $m$  di antara 1-999.999. Nomor rumah untuk setiap daerah kemudian diurutkan dari terbesar ke terkecil menggunakan fungsi `selectionSort`. Hasil urutan nomor rumah ditampilkan untuk masing-masing daerah. Program ini memastikan input valid sebelum melanjutkan proses.

## 2. Source Code

```
// MUHAMMAD RAGIEL PRASTYO
// 2311102183
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)
    }

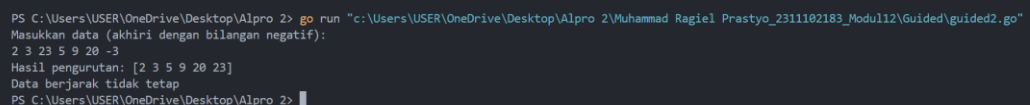
    // Urutkan data menggunakan insertion sort
    insertionSort(data)

    // 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 Output



```

PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul12\Guided\guided2.go"
Masukkan data (akhiri dengan bilangan negatif):
2 3 23 5 9 20 -3
Hasil pengurutan: [2 3 5 9 20 23]
Data berjarak tidak tetap
PS C:\Users\USER\OneDrive\Desktop\Alpro 2>

```

### Penjelasan:

Program diatas mengurutkan data yang dimasukkan pengguna dan memeriksa apakah elemen-elemennya memiliki jarak tetap. Pengguna memasukkan sejumlah bilangan positif yang diakhiri dengan bilangan negatif. Data tersebut diurutkan menggunakan algoritma *insertion sort*. Setelah data terurut, program memeriksa apakah selisih antar elemen dalam array selalu sama. Jika jaraknya tetap, program mencetak jarak tersebut; jika tidak, program menyatakan bahwa

jarak tidak tetap. Program juga menangani input dengan kurang dari dua elemen, yang secara otomatis dianggap memiliki jarak tetap.

### III. UNGUIDED

#### 1. Source Code

```
// MUHAMMAD RAGIEL PRASTYO
// 2311102183

package main
import (
    "fmt"
)

// Fungsi untuk mengurutkan array secara ascending menggunakan
selection sort
func selectionSortAsc(arr []int) {
    n := len(arr)
    for i := 0; i < n-1; i++ {
        minIdx := i
        for j := i + 1; j < n; j++ {
            if arr[j] < arr[minIdx] {
                minIdx = j
            }
        }
        arr[i], arr[minIdx] = arr[minIdx], arr[i]
    }
}

// Fungsi untuk mengurutkan array secara descending menggunakan
selection sort
func selectionSortDesc(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 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
        var ganjil []int
        var genap []int
        for _, num := range houses {
            if num%2 == 0 {
                genap = append(genap, num)
            } else {
                ganjil = append(ganjil, num)
            }
        }

        // Urutkan ganjil (ascending) dan genap (descending)
        selectionSortAsc(ganjil)
        selectionSortDesc(genap)

        // Cetak hasil
        fmt.Printf("Hasil urutan rumah untuk daerah ke-%d: ", i+1)
        for _, num := range ganjil {
            fmt.Printf("%d ", num)
        }
        for _, num := range genap {

```

```

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

```

## Screenshot Output

```

PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul12\Unguided\unguided1.go"
Masukkan jumlah daerah (n): 3

Masukkan jumlah rumah untuk daerah ke-1: 6
Masukkan nomor rumah untuk daerah ke-1:
12 43 76 2 45 5
Hasil urutan rumah untuk daerah ke-1: 5 43 45 76 12 2

Masukkan jumlah rumah untuk daerah ke-2: 9
Masukkan nomor rumah untuk daerah ke-2:
32 45 32 1 6 90 65 34 69
Hasil urutan rumah untuk daerah ke-2: 1 45 65 69 90 34 32 32 6

Masukkan jumlah rumah untuk daerah ke-3: 11
Masukkan nomor rumah untuk daerah ke-3:
12 13 45 89 97 99 34 1 5 76 89
Hasil urutan rumah untuk daerah ke-3: 1 5 13 45 89 97 99 76 34 12
PS C:\Users\USER\OneDrive\Desktop\Alpro 2>

```

## Penjelasan:

Program diatas mengurutkan nomor rumah di beberapa daerah dengan memisahkan bilangan ganjil dan genap. Pengguna memasukkan jumlah daerah ( $n$ ), jumlah rumah di setiap daerah ( $m$ ), dan nomor rumah. Nomor rumah ganjil diurutkan secara *ascending* dan genap secara *descending* menggunakan *selection sort*. Hasilnya ditampilkan dengan nomor ganjil diikuti nomor genap untuk setiap daerah. Program juga memvalidasi input agar sesuai dengan batasan.

## 2. Source Code

```

// MUHAMMAD RAGIEL PRASTYO
// 2311102183

package main
import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan insertion sort
func insertionSort(arr []int) {
    n := len(arr)
    for i := 1; i < n; i++ {
        key := arr[i]
        j := i - 1

```

```

        // Pindahkan elemen yang lebih besar dari key ke satu posisi di
        // depan
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

// Fungsi untuk menghitung median dari array yang sudah terurut
func getMedian(arr []int) int {
    n := len(arr)
    if n%2 == 1 {
        // Jika jumlah elemen ganjil, median adalah elemen tengah
        return arr[n/2]
    }
    // Jika jumlah elemen genap, median adalah rata-rata dua elemen
    // tengah
    return (arr[n/2-1] + arr[n/2]) / 2
}

func main() {
    var input int
    data := []int{} // Array untuk menyimpan data yang valid

    fmt.Println("Masukkan bilangan bulat (akhiri dengan -5313):")

    for {
        fmt.Scan(&input)

        if input == -5313 {
            // Marker untuk mengakhiri program
            break
        } else if input == 0 {
            // Jika menemukan angka 0, urutkan array dan cetak median
            insertionSort(data)
            if len(data) > 0 {
                median := getMedian(data)
                fmt.Println("Median saat ini:", median)
            }
        } else if input > 0 {
            // Hanya tambahkan bilangan bulat positif ke array
            data = append(data, input)
        }
    }
}

```

```
}  
}
```

### Screenshot Output

```
PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul12\Unguided\unguided2.go"  
Masukkan bilangan bulat (akhiri dengan -5313):  
23 09 05 20 2 3 9 5 -5313  
Median saat ini: 23  
PS C:\Users\USER\OneDrive\Desktop\Alpro 2> |
```

### Penjelasan:

Program diatas menerima bilangan bulat positif, menghitung median data yang diinput, dan berhenti saat pengguna memasukkan **-5313**. Angka **0** memicu pengurutan data dengan *insertion sort* dan menghitung median: elemen tengah jika jumlah data ganjil, atau rata-rata dua elemen tengah jika genap. Bilangan negatif selain **-5313** diabaikan. Median dicetak hanya jika ada data valid. Program terus menerima input hingga angka *marker* diberikan.

### 3. Source Code

```
// MUHAMMAD RAGIEL PRASTYO  
// 2311102183  
package main  
import (  
    "fmt"  
)  
  
// Definisi struct untuk Buku  
type Buku struct {  
    id      int  
    judul   string  
    penulis string  
    penerbit string  
    eksemplar int  
    tahun   int  
    rating  int  
}  
  
// Fungsi untuk menambahkan data buku ke pustaka  
func DaftarkanBuku(pustaka *[]Buku, n int) {  
    for i := 0; i < n; i++ {  
        var buku Buku  
        fmt.Printf("Masukkan data untuk buku ke-%d (id, judul, penulis, penerbit, eksemplar, tahun, rating):\n", i+1)
```

```

        fmt.Scan(&buku.id, &buku.judul, &buku.penulis,
        &buku.penerbit, &buku.eksemplar, &buku.tahun, &buku.rating)
        *pustaka = append(*pustaka, buku)
    }
}

// Fungsi untuk mencetak buku dengan rating tertinggi
func CetakFavorit(pustaka []Buku, n int) {
    if len(pustaka) == 0 {
        fmt.Println("Pustaka kosong.")
        return
    }
    terfavorit := pustaka[0]
    for _, buku := range pustaka {
        if buku.rating > terfavorit.rating {
            terfavorit = buku
        }
    }
    fmt.Println("Buku dengan rating tertinggi:")
    fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s,
    Eksemplar: %d, Tahun: %d, Rating: %d\n",
        terfavorit.id, terfavorit.judul, terfavorit.penulis,
        terfavorit.penerbit, terfavorit.eksemplar, terfavorit.tahun,
        terfavorit.rating)
}

// Fungsi untuk mengurutkan array buku berdasarkan rating secara
descending
func UrutkanBuku(pustaka *[]Buku, n int) {
    for i := 1; i < len(*pustaka); 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 lima buku dengan rating tertinggi
func Cetak5Terbaik(pustaka []Buku, n int) {
    fmt.Println("Lima buku dengan rating tertinggi:")
    for i := 0; i < 5 && i < len(pustaka); i++ {
        buku := pustaka[i]

```



```

        fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s,
Eksemplar: %d, Tahun: %d, Rating: %d\n",
            buku.id, buku.judul, buku.penulis, buku.penerbit,
            buku.eksemplar, buku.tahun, buku.rating)
    }
}

// Fungsi untuk mencari buku dengan rating tertentu
func CariBuku(pustaka []Buku, n int, r int) {
    ditemukan := false
    for _, buku := range pustaka {
        if buku.rating == r {
            ditemukan = true
            fmt.Printf("ID: %d, Judul: %s, Penulis: %s, Penerbit: %s,
Eksemplar: %d, Tahun: %d, Rating: %d\n",
                buku.id, buku.judul, buku.penulis, buku.penerbit,
                buku.eksemplar, buku.tahun, buku.rating)
        }
    }
    if !ditemukan {
        fmt.Println("Tidak ada buku dengan rating tersebut.")
    }
}

func main() {
    var n int
    fmt.Print("Masukkan jumlah buku di perpustakaan: ")
    fmt.Scan(&n)

    if n <= 0 || n > 7919 {
        fmt.Println("Jumlah buku harus antara 1 hingga 7919.")
        return
    }

    var pustaka []Buku

    // Input data buku
    DaftarkanBuku(&pustaka, n)

    // Cetak buku dengan rating tertinggi
    CetakFavorit(pustaka, n)

    // Urutkan buku berdasarkan rating
    UrutkanBuku(&pustaka, n)

    // Cetak lima buku dengan rating tertinggi

```

```
Cetak5Terbaik(pustaka, n)
```

```
// Cari buku dengan rating tertentu
```

```
var rating int
```

```
fmt.Print("Masukkan rating buku yang ingin dicari: ")
```

```
fmt.Scan(&rating)
```

```
CariBuku(pustaka, n, rating)
```

```
}
```

## Screenshot Output

```
PS C:\Users\USER\OneDrive\Desktop\Alpro 2> go run "c:\Users\USER\OneDrive\Desktop\Alpro 2\Muhammad Ragiel Prastyo_2311102183_Modul12\Unguided\unguided3.go"
Masukkan jumlah buku di perpustakaan: 5
Masukkan data untuk buku ke-1 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
123 WHY Ibra BUMI 95 2020 85
Masukkan data untuk buku ke-2 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
345 WHERE Siti SUTERA 101 2018 87
Masukkan data untuk buku ke-3 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
567 WHEN Aloy DRILL 132 2020 90
Masukkan data untuk buku ke-4 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
789 WHUT Snow Aleee 97 2021 88
Masukkan data untuk buku ke-5 (id, judul, penulis, penerbit, eksemplar, tahun, rating):
654 SLAY Gaby Snite 100 2020 99
Buku dengan rating tertinggi:
ID: 654, Judul: SLAY, Penulis: Gaby, Penerbit: Snite, Eksemplar: 100, Tahun: 2020, Rating: 99
Lima buku dengan rating tertinggi:
ID: 654, Judul: SLAY, Penulis: Gaby, Penerbit: Snite, Eksemplar: 100, Tahun: 2020, Rating: 99
ID: 567, Judul: WHEN, Penulis: Aloy, Penerbit: DRILL, Eksemplar: 132, Tahun: 2020, Rating: 90
ID: 789, Judul: WHUT, Penulis: Snow, Penerbit: Aleee, Eksemplar: 97, Tahun: 2021, Rating: 88
ID: 345, Judul: WHERE, Penulis: Siti, Penerbit: SUTERA, Eksemplar: 101, Tahun: 2018, Rating: 87
ID: 123, Judul: WHY, Penulis: Ibra, Penerbit: BUMI, Eksemplar: 95, Tahun: 2020, Rating: 85
Masukkan rating buku yang ingin dicari: █
```

## Penjelasan:

Program ini mengelola data buku di perpustakaan dengan fitur-fitur utama seperti menampilkan buku dengan rating tertinggi, mengurutkan buku berdasarkan rating secara menurun, menampilkan lima buku dengan rating tertinggi, dan mencari buku dengan rating tertentu. Setiap buku memiliki atribut seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating. Pengguna memasukkan jumlah buku, kemudian data buku dimasukkan satu per satu. Program menggunakan algoritma *insertion sort* untuk mengurutkan buku berdasarkan rating dan menyediakan fitur pencarian buku berdasarkan rating tertentu. Semua data buku dikelola menggunakan array *struct* untuk mempermudah pengelolaan dan pengolahan informasi.