

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII  
PENGURUTAN DATA**



**Disusun Oleh :**

**Muhammad Ihab Aufa Rafi / 2311102226**

**S1IF-11-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi, S.Kom., M.Kom.**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## I. DASAR TEORI

### 1.1 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	<code>i &lt;- 1</code>	<code>i = 1</code>
2	<code>while i &lt;= n-1 do</code>	<code>for i &lt;= n-1 {</code>
3	<code>    idx_min &lt;- i-1</code>	<code>    idx_min = i-1</code>
4	<code>    j &lt;- i</code>	<code>    j = i</code>
5	<code>    while j &lt; n do</code>	<code>    for j &lt; n {</code>
6	<code>        if a[idx_min] &gt; a[j] then</code>	<code>        if a[idx_min] &gt; a[j] {</code>
7	<code>            idx_min &lt;- j</code>	<code>            idx_min = j</code>
8	<code>        endif</code>	<code>        }</code>
9	<code>        j &lt;- j+1</code>	<code>        j = j + 1</code>
10	<code>    endwhile</code>	<code>    }</code>
11	<code>    t &lt;- a[idx_min]</code>	<code>    t = a[idx_min]</code>
12	<code>    a[idx_min] &lt;- a[i-1]</code>	<code>    a[idx_min] = a[i-1]</code>
13	<code>    a[i-1] &lt;- t</code>	<code>    a[i-1] = t</code>
14	<code>    i &lt;- i+1</code>	<code>    i &lt;- i + 1</code>
15	<code>endwhile</code>	<code>}</code>

### 1.2 Algoritma Selection Sort

Adapun algoritma *selection sort* pada program 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 array-nya.

```

..  ...
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 menggunakan
18     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 }

```

### 1.3 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 suatu 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	}

## 1.4 Algoritma Insertion Sort

Adapun algoritma insertion sort pada program 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. Soal Studi Case

Hercules, preman terkenal seantero ibukota, memiliki kerabat di banyak daerah. Tentunya Hercules sangat suka mengunjungi semua kerabatnya itu.

Diberikan masukan nomor rumah dari semua kerabatnya di suatu daerah, buatlah program **rumahkerabat** yang akan menyusun nomor-nomor rumah kerabatnya secara terurut membesar menggunakan algoritma selection sort.

**Masukan** dimulai dengan sebuah integer **n** ( $0 < n < 1000$ ), banyaknya daerah kerabat Hercules tinggal. Isi **n** baris berikutnya selalu dimulai dengan sebuah integer **m** ( $0 < m < 1000000$ ) yang menyatakan banyaknya rumah kerabat di daerah tersebut, diikuti dengan rangkaian bilangan bulat positif, nomor rumah para kerabat.

**Keluaran** terdiri dari **n** baris, yaitu rangkaian rumah kerabatnya terurut membesar di masing-masing daerah.

No	Masukan	Keluaran
1	3 <u>5</u> 2 1 7 9 13 <u>6</u> 189 15 27 39 75 133 <u>3</u> 4 9 1	1 2 7 9 13 15 27 39 75 133 189 1 4 9

**Keterangan:** Terdapat 3 daerah dalam contoh input, dan di masing-masing daerah mempunyai 5, 6, dan 3 kerabat.

### Sourcecode

```
package main

import (
    "fmt"
)

// Fungsi untuk mengurutkan array menggunakan Selection Sort
func selectionSort(arr []int, n int) {
    for i := 0; i < n-1; i++ {
        idxMin := i
        for j := i + 1; j < n; j++ {
            // Cari elemen terkecil
            if arr[j] < arr[idxMin] {
                idxMin = j
            }
        }
    }
}
```

```

        // Tukar elemen terkecil dengan elemen di posisi i
        arr[i], arr[idxMin] = arr[idxMin], arr[i]
    }
}

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

    // Proses tiap daerah
    for daerah := 1; daerah <= n; daerah++ {
        var m int
        fmt.Printf("\nMasukkan jumlah nomor rumah kerabat
untuk daerah %d: ", daerah)
        fmt.Scan(&m)

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

        // Urutkan array dari terkecil ke terbesar
        selectionSort(arr, m)

        // Tampilkan hasil
        fmt.Printf("Nomor rumah terurut untuk daerah %d: ",
daerah)
        for _, num := range arr {
            fmt.Printf("%d ", num)
        }
        fmt.Println()
    }
}

```

### Screenshoot Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v
PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Guided12-1\Guided12-1.go"
Masukkan jumlah daerah kerabat (n): 3

Masukkan jumlah nomor rumah kerabat untuk daerah 1: 5
Masukkan 5 nomor rumah kerabat: 2 1 7 9 13
Nomor rumah terurut untuk daerah 1: 1 2 7 9 13

Masukkan jumlah nomor rumah kerabat untuk daerah 2: 6
Masukkan 6 nomor rumah kerabat: 19 15 27 39 75 133
Nomor rumah terurut untuk daerah 2: 15 19 27 39 75 133

Masukkan jumlah nomor rumah kerabat untuk daerah 3: 3
Masukkan 3 nomor rumah kerabat: 4 9 1
Nomor rumah terurut untuk daerah 3: 1 4 9
PS D:\Alpro2Golang> 
```

## Deskripsi Program

Program ini digunakan untuk mengurutkan nomor rumah kerabat Hercules di beberapa daerah menggunakan algoritma *Selection Sort*. Proses pengurutan dilakukan untuk setiap daerah berdasarkan masukan pengguna, dan hasil akhirnya adalah daftar nomor rumah yang terurut dari kecil ke besar.

Fungsi `selectionSort` digunakan untuk mengurutkan array berisi nomor rumah dengan algoritma *Selection Sort*. Fungsi ini menerima dua parameter, yaitu `arr`, array nomor rumah yang akan diurutkan, dan `n`, jumlah elemen dalam array. Proses pengurutan dimulai dengan mencari elemen terkecil di setiap iterasi dan menukarnya dengan elemen di posisi awal subarray yang sedang diproses. Dengan demikian, elemen terkecil akan ditempatkan di urutan yang sesuai hingga seluruh array terurut.

Di dalam fungsi `main`, program memulai dengan meminta input jumlah daerah (`n`). Untuk setiap daerah, program meminta pengguna untuk memasukkan jumlah nomor rumah kerabat di daerah tersebut (`m`) serta daftar nomor rumah. Data nomor rumah disimpan dalam sebuah array, kemudian diproses menggunakan fungsi `selectionSort` untuk diurutkan.

## 2. Soal Studi Case

Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metode insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.



**Masukan** terdiri dari sekumpulan bilangan butat yang diakhiri Oleh bilangan negatif. Hanya bilangan non negatif saja yang disimpan ke dalam array.

**Keluaran** terdiri dari dua baris. Baris pertama adalah isi dari array setelah dilakukan pengurutan, sedangkan baris kedua adalah status jarak setiap bilangan yang ada di dalam array. "Data berjarak x" atau "data berjarak tidak tetap".

Contoh masukan dan keluaran:

No	Masukan	Keluaran
1	31 13 25 43 1 7 19 37 -5	1 7 13 19 25 31 37 43 Data berjarak 6
2	4 40 14 8 26 1 38 2 32 -31	1 2 4 8 14 26 32 38 40 Data berjarak tidak tetap

### Sourcecode

```
package main

import (
    "fmt"
)

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

        // Geser elemen yang lebih besar dari key ke kanan
        for j >= 0 && arr[j] > key {
            arr[j+1] = arr[j]
            j--
        }
        arr[j+1] = key
    }
}

// Fungsi untuk memeriksa apakah selisih elemen array tetap
func isConstantDifference(arr []int, n int) (bool, int) {
    if n < 2 {
        return true, 0
    }
}
```

```

        difference := arr[1] - arr[0]
        for i := 1; i < n-1; i++ {
            if arr[i+1]-arr[i] != difference {
                return false, 0
            }
        }
        return true, difference
    }

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

    // Input data hingga bilangan negatif ditemukan
    fmt.Println("Masukkan data integer (akhiri dengan bilangan negatif):")
    for {
        fmt.Scan(&num)
        if num < 0 {
            break
        }
        arr = append(arr, num)
    }

    n := len(arr)

    // Urutkan array menggunakan Insertion Sort
    insertionSort(arr, n)

    // Periksa apakah selisih elemen tetap
    isConstant, difference := isConstantDifference(arr, n)

    // Tampilkan hasil pengurutan
    fmt.Println("Array setelah diurutkan:")
    for _, val := range arr {
        fmt.Printf("%d ", val)
    }
    fmt.Println()

    // Tampilkan status jarak
    if isConstant {
        fmt.Printf("Data berjarak %d\n", difference)
    } else {
        fmt.Println("Data berjarak tidak tetap")
    }
}

```

```
} }
```

## Screenshoot Output

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code
PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Guided12-2\Guided12-2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
31 13 25 43 1 7 19 37 -5
Array setelah diurutkan:
1 7 13 19 25 31 37 43
Data berjarak 6
PS D:\Alpro2Golang>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code
PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Guided12-2\Guided12-2.go"
Masukkan data integer (akhiri dengan bilangan negatif):
4 40 14 8 26 1 38 2 32 -31
Array setelah diurutkan:
1 2 4 8 14 26 32 38 40
Data berjarak tidak tetap
PS D:\Alpro2Golang>
```

## Deskripsi Program

Program ini bertujuan untuk mengurutkan sejumlah bilangan menggunakan algoritma *Insertion Sort* dan memeriksa apakah perbedaan antara elemen-elemen dalam array memiliki selisih yang tetap. Program meminta pengguna untuk memasukkan bilangan secara berulang hingga bilangan negatif dimasukkan sebagai tanda akhir input.

Fungsi `insertionSort` digunakan untuk mengurutkan array secara menaik. Fungsi ini menerima parameter berupa array `arr` dan jumlah elemen `n`. Proses pengurutan dilakukan dengan menyisipkan elemen pada posisi yang sesuai di subarray yang telah terurut. Elemen-elemen yang lebih besar dari elemen saat ini (`key`) akan digeser ke kanan untuk memberikan ruang bagi elemen tersebut.

Fungsi `isConstantDifference` memeriksa apakah selisih antar elemen dalam array yang telah diurutkan adalah tetap. Fungsi ini menerima array `arr` dan jumlah elemen `n`, lalu menghitung selisih awal antara dua elemen pertama. Selama iterasi, fungsi membandingkan selisih elemen berikutnya dengan selisih awal. Jika ditemukan perbedaan selisih, fungsi mengembalikan nilai `false`. Jika selisih tetap, fungsi mengembalikan `true` bersama nilai selisih.

Di dalam fungsi main, pengguna diminta untuk memasukkan bilangan secara bertahap. Data bilangan ini disimpan dalam array hingga pengguna memasukkan bilangan negatif sebagai penanda akhir input. Array kemudian diurutkan menggunakan fungsi `insertionSort`, dan hasilnya diperiksa oleh fungsi `isConstantDifference`.

Setelah proses selesai, program menampilkan array yang telah diurutkan. Jika elemen-elemen array memiliki selisih tetap, program mencetak nilai selisih tersebut. Jika tidak, program akan memberikan informasi bahwa data tidak memiliki selisih tetap.

### III. UNGUIDED

#### 1. Soal Studi Case

Belakangan diketahui ternyata Hercules itu tidak berani menyeberang jalan, maka selalu diusahakan agar hanya menyeberang jalan sesedikit mungkin, hanya diujung jalan. Karena nomor rumah sisi kiri jalan selalu ganjil dan sisi kanan jalan selalu genap, maka buatlah program kerabat dekat yang akan menampilkan nomor rumah mulai dari nomor yang ganjil lebih dulu terurut membesar dan kemudian menampilkan nomor rumah dengan nomor genap terurut mengecil.

Format **Masukan** masih persis sama seperti sebelumnya.

**Keluaran** terdiri dari n baris, yaitu rangkaian rumah kerabatnya terurut membesar untuk nomor ganjil, diikuti dengan terurut mengecil untuk nomor genap, di masing-masing daerah.

No	Masukan	Keluaran
1	3 <u>5</u> 2 1 7 9 13 <u>6</u> 189 15 27 39 75 133 <u>3</u> 2 8 4	1 7 9 13 2 15 27 39 75 133 189 8 4 2

**Keterangan:** Terdapat 3 daerah dalam contoh input. 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.

#### Sourcecode

```
package main

import "fmt"

func selectionSortGanjil(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]
}
}

func selectionSortGenap(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]
    }
}

func main() {
    fmt.Print("Masukkan jumlah daerah: ")
    var n226 int
    fmt.Scan(&n226)

    resultSemuaDaerah := make([]string, n226)

    for i := 0; i < n226; i++ {
        fmt.Printf("\nUntuk daerah ke-%d:\n", i+1)
        fmt.Print("Masukkan jumlah rumah di daerah ini: ")

        var m226 int
        fmt.Scan(&m226)
        fmt.Printf("Masukkan %d nomor rumah (dipisahkan
dengan spasi): ", m226)
        numbers := make([]int, m226)
        for j := 0; j < m226; j++ {
            fmt.Scan(&numbers[j])
        }

        var odd, even []int
        for _, num := range numbers {
            if num%2 == 1 {
                odd = append(odd, num)
            } else {

```

```

        even = append(even, num)
    }
}

selectionSortGanjil(odd)
selectionSortGenap(even)

var hasil string
for j := 0; j < len(odd); j++ {
    if j > 0 {
        hasil += " "
    }
    hasil += fmt.Sprintf("%d", odd[j])
}

if len(even) > 0 {
    if len(odd) > 0 {
        hasil += " "
    }
    for j := 0; j < len(even); j++ {
        if j > 0 {
            hasil += " "
        }
        hasil += fmt.Sprintf("%d", even[j])
    }
}
resultSemuaDaerah[i] = hasil
}

fmt.Println("\nHasil pengurutan untuk setiap daerah:")
for i, hasil := range resultSemuaDaerah {
    fmt.Printf("Daerah %d: %s\n", i+1, hasil)
}
}

```

**Screenshoot Output**

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Code + v
PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Unguided12-1\Unguided12-1.go"
Masukkan jumlah daerah: 3

Untuk daerah ke-1:
Masukkan jumlah rumah di daerah ini: 5
Masukkan 5 nomor rumah (dipisahkan dengan spasi): 2 8 7 9 13

Untuk daerah ke-2:
Masukkan jumlah rumah di daerah ini: 6
Masukkan 6 nomor rumah (dipisahkan dengan spasi): 189 15 27 39 133 75

Untuk daerah ke-3:
Masukkan jumlah rumah di daerah ini: 3
Masukkan 3 nomor rumah (dipisahkan dengan spasi): 2 8 4

Hasil pengurutan untuk setiap daerah:
Daerah 1: 7 9 13 8 2
Daerah 2: 15 27 39 75 133 189
Daerah 3: 8 4 2
PS D:\Alpro2Golang>
```

## Deskripsi Program

Program ini bertujuan untuk mengurutkan nomor rumah di beberapa daerah dengan aturan khusus: nomor rumah bernomor ganjil diurutkan dari kecil ke besar, lalu nomor rumah bernomor genap diurutkan dari besar ke kecil. Hasil pengurutan ditampilkan untuk setiap daerah berdasarkan input dari pengguna.

Fungsi `selectionSortGanjil` bertugas mengurutkan array berisi nomor rumah ganjil dari kecil ke besar menggunakan algoritma Selection Sort. Fungsi ini iterasi melalui array untuk menemukan elemen terkecil, kemudian menukarnya dengan elemen di posisi awal subarray yang sedang diproses.

Fungsi `selectionSortGenap` digunakan untuk mengurutkan array berisi nomor rumah genap dari besar ke kecil. Prosesnya mirip dengan `selectionSortGanjil`, tetapi membandingkan elemen untuk menemukan elemen terbesar, bukan terkecil.

Di dalam fungsi `main`, pengguna diminta memasukkan jumlah daerah (`n226`). Untuk setiap daerah:

- Program meminta jumlah nomor rumah (`m226`) dan daftar nomor rumah.
- Nomor rumah dipisahkan menjadi dua kategori: ganjil dan genap.



- Nomor rumah ganjil diurutkan menggunakan **selectionSortGanjil**, dan nomor rumah genap diurutkan menggunakan **selectionSortGenap**.
- Hasil pengurutan digabungkan dalam satu string, dengan nomor rumah ganjil secara menaik terlebih dahulu diikuti nomor rumah genap secara menurun.

## 2. Soal Studi Case

Kompetisi pemrograman yang baru saja berlalu diikuti Oleh 17 tim dari berbagai perguruan tinggi ternama. Dalam kompetisi tersebut, setiap tim berlomba untuk menyelesaikan sebanyak mungkin problem yang diberikan. Dari 13 problem yang diberikan, ada satu problem yang menarik. Problem tersebut mudah dipahami, hampir semua tim mencoba untuk menyelesaikannya, tetapi hanya 3 tim yang berhasil. Apa sih problemnya?

*"Median adalah nilai tengah dari suatu koleksi data yang sudah terurut. Jika jumlah data genap, maka nilai median adalah rerata dari kedua nilai tengahnya. Pada problem ini, semua data merupakan bilangan bulat positif, dan karenanya rerata nilai tengah dibulatkan ke bawah. "*

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, yang berarti data berjumlah genap, maka nilai mediannya adalah  $(11+13)/2=12$

### Petunjuk:

Untuk setiap data bukan 0 (dan bukan marker -5313) simpan ke dalam array. Dan setiap kali menemukan bilangan 0, urutkanlah data yang sudah tersimpan dengan menggunakan metode selection sort dan ambil mediannya.

### Sourcecode

```
package main

import "fmt"

const nMax = 1000000

func selectionSort(arr []int, n int) {
    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]
    }
}

func hitungMedian(arr []int, n int) float64 {
    if n%2 == 0 {
        return float64(arr[n/2-1]+arr[n/2]) / 2.0
    }
    return float64(arr[n/2])
}

func main() {

    var numbers [nMax]int
    var temp [nMax]int
    var count int
    var num226 int
    fmt.Println("Masukkan rangkaian bilangan bulat (0 untuk
median sementara, akhiri dengan -5313): ")
    for {
        fmt.Scan(&num226)
        if num226 == -5313 {
            break
        }
    }
}
```

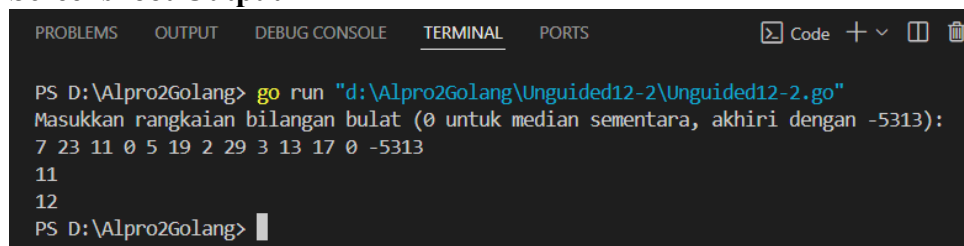
```

        if num226 != 0 {
            numbers[count] = num226
            count++
        } else {
            if count > 0 {
                for i := 0; i < count; i++ {
                    temp[i] = numbers[i]
                }
                selectionSort(temp[:], count)

                median := hitungMedian(temp[:], count)
                fmt.Printf("%.0f\n", median)
            }
        }
    }
}

```

### Screenshoot Output



```

PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Unguided12-2\Unguided12-2.go"
Masukkan rangkaian bilangan bulat (0 untuk median sementara, akhiri dengan -5313):
7 23 11 0 5 19 2 29 3 13 17 0 -5313
11
12
PS D:\Alpro2Golang>

```

### Deskripsi Program

Program ini bertujuan untuk menghitung median dari rangkaian bilangan bulat yang dimasukkan secara dinamis oleh pengguna. Median dihitung setiap kali pengguna memasukkan angka 0, dan proses input berakhir saat angka -5313 dimasukkan.

Fungsi **selectionSort** digunakan untuk mengurutkan array dari kecil ke besar menggunakan algoritma *Selection Sort*. Fungsi ini menerima array **arr** dan jumlah elemen **n** sebagai parameter. Proses pengurutan dimulai dengan mencari elemen terkecil di subarray yang belum terurut dan menukarnya dengan elemen di posisi awal subarray.

Fungsi **hitungMedian** bertugas menghitung median dari array yang telah diurutkan. Jika jumlah elemen **n** adalah genap, median dihitung sebagai rata-rata dari dua elemen tengah. Jika jumlah elemen ganjil, median adalah elemen tengah dari array.

Di dalam fungsi **main**, program meminta pengguna untuk memasukkan bilangan bulat satu per satu. Rangkaian bilangan disimpan dalam array **numbers**. Setiap kali pengguna memasukkan 0, program akan:

- Menyalin isi array **numbers** ke array sementara **temp**.
- Mengurutkan array **temp** menggunakan fungsi **selectionSort**.
- Menghitung dan mencetak median dari array **temp** menggunakan fungsi **hitungMedian**.

Proses akan terus berlanjut hingga pengguna memasukkan bilangan -5313, yang akan menghentikan program.

### 3. Soal Studi Case

Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan. Misalnya terdefinisi suatu struct dan array seperti berikut ini:

Buatlah program dengan spesifikasi subprogram sebagai berikut:

```
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 di bawah 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.}

```

## Sourcecode

```

package main

import "fmt"

const maxBuku = 7919

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

type DaftarBuku [maxBuku]Buku

func DaftarkanBuku(pustaka *DaftarBuku, n *int) {
    fmt.Println("Masukkan ID Buku: ")
    fmt.Scanln(&pustaka[*n].id)
    fmt.Println("Masukkan Judul Buku: ")
    fmt.Scanln(&pustaka[*n].judul)
    fmt.Println("Masukkan Nama Penulis: ")
    fmt.Scanln(&pustaka[*n].penulis)
    fmt.Println("Masukkan Nama Penerbit: ")
}

```

```

        fmt.Scanln(&pustaka[*n].penerbit)
        fmt.Print("Masukkan Jumlah Eksemplar: ")
        fmt.Scanln(&pustaka[*n].eksemplar)
        fmt.Print("Masukkan Tahun Terbit: ")
        fmt.Scanln(&pustaka[*n].tahun)
        fmt.Print("Masukkan Rating Buku (1-10): ")
        fmt.Scanln(&pustaka[*n].rating)
        *n++
    }

func CetakFavorit(pustaka DaftarBuku, n int) {
    fmt.Println("\nBuku Terfavorit:")
    maxRating := 0
    for i := 0; i < n; i++ {
        if pustaka[i].rating > maxRating {
            maxRating = pustaka[i].rating
        }
    }

    for i := 0; i < n; i++ {
        if pustaka[i].rating == maxRating {
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %d\n",
                pustaka[i].judul, pustaka[i].penulis,
                pustaka[i].penerbit, pustaka[i].tahun, pustaka[i].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
    }
}

func Cetak5Terbaru(pustaka DaftarBuku, n int) {
    fmt.Println("5 Judul Buku dengan Rating Tertinggi:")
    for i := 0; i < n && i < 5; i++ {
        fmt.Println(pustaka[i].judul)
    }
}

```

```

    }
}

func CariBuku(pustaka DaftarBuku, n int, r int) {
    fmt.Printf("\nBuku dengan Rating %d:\n", r)
    found := false
    for i := 0; i < n; i++ {
        if pustaka[i].rating == r {
            fmt.Printf("Judul: %s, Penulis: %s, Penerbit: %s, Tahun: %d, Rating: %d\n",
                pustaka[i].judul, pustaka[i].penulis,
                pustaka[i].penerbit, pustaka[i].tahun, pustaka[i].rating)
            found = true
        }
    }
    if !found {
        fmt.Println("Tidak ada buku dengan rating seperti itu")
    }
}

func main() {
    var pustaka DaftarBuku
    var n226, cariRating int
    var count int

    fmt.Print("Masukkan jumlah buku: ")
    fmt.Scanln(&n226)

    fmt.Println("\nMasukkan data untuk setiap buku:")
    for i := 0; i < n226; i++ {
        fmt.Printf("\nData Buku ke-%d:\n", i+1)
        DaftarkanBuku(&pustaka, &count)
    }

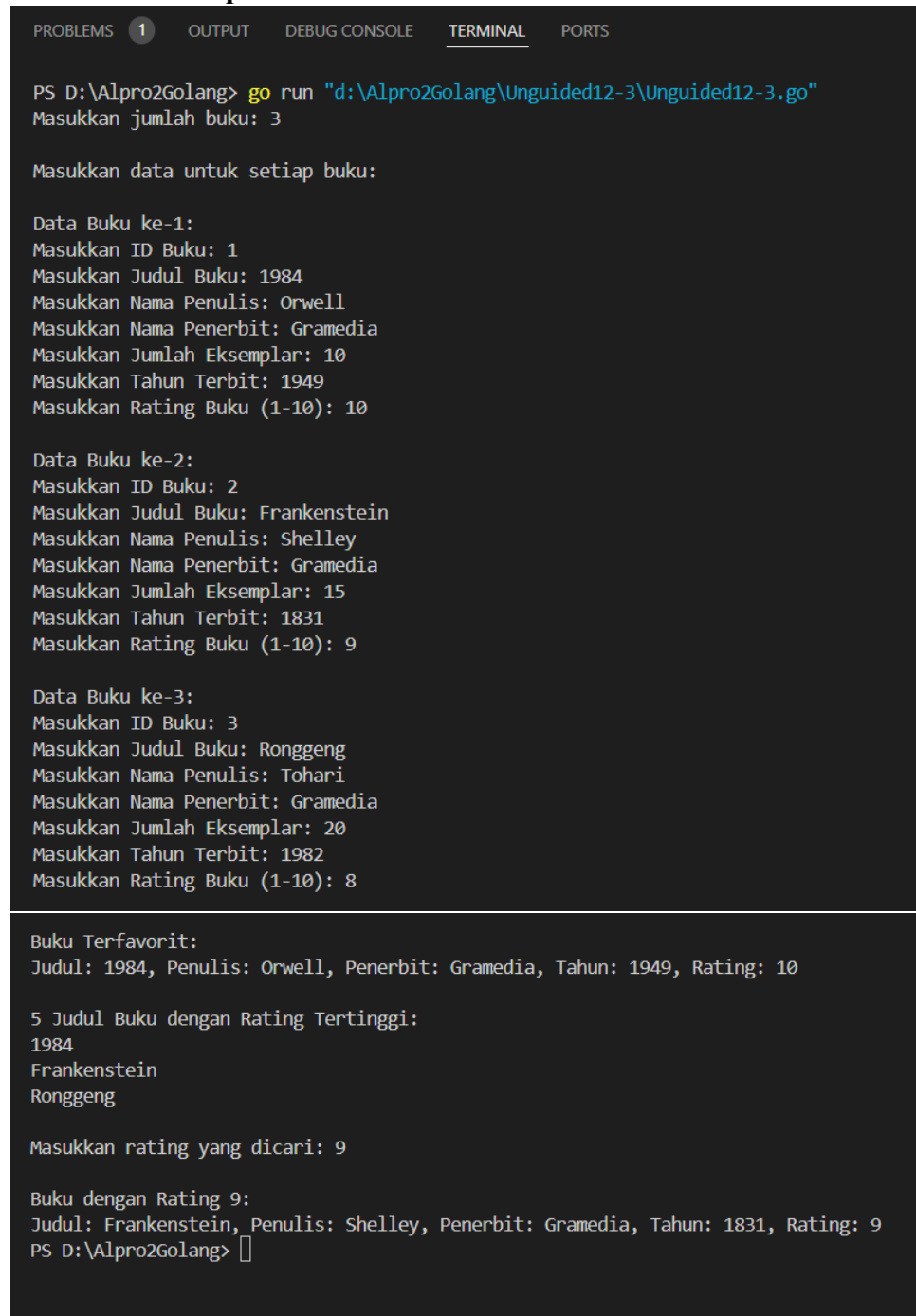
    CetakFavorit(pustaka, n226)

    fmt.Println()
    UrutBuku(&pustaka, n226)
    Cetak5Terbaru(pustaka, n226)

    fmt.Print("\nMasukkan rating yang dicari: ")
    fmt.Scanln(&cariRating)
    CariBuku(pustaka, n226, cariRating)
}

```

## Screenshoot Output



```
PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS D:\Alpro2Golang> go run "d:\Alpro2Golang\Unguided12-3\Unguided12-3.go"
Masukkan jumlah buku: 3

Masukkan data untuk setiap buku:

Data Buku ke-1:
Masukkan ID Buku: 1
Masukkan Judul Buku: 1984
Masukkan Nama Penulis: Orwell
Masukkan Nama Penerbit: Gramedia
Masukkan Jumlah Eksemplar: 10
Masukkan Tahun Terbit: 1949
Masukkan Rating Buku (1-10): 10

Data Buku ke-2:
Masukkan ID Buku: 2
Masukkan Judul Buku: Frankenstein
Masukkan Nama Penulis: Shelley
Masukkan Nama Penerbit: Gramedia
Masukkan Jumlah Eksemplar: 15
Masukkan Tahun Terbit: 1831
Masukkan Rating Buku (1-10): 9

Data Buku ke-3:
Masukkan ID Buku: 3
Masukkan Judul Buku: Ronggeng
Masukkan Nama Penulis: Tohari
Masukkan Nama Penerbit: Gramedia
Masukkan Jumlah Eksemplar: 20
Masukkan Tahun Terbit: 1982
Masukkan Rating Buku (1-10): 8

Buku Terfavorit:
Judul: 1984, Penulis: Orwell, Penerbit: Gramedia, Tahun: 1949, Rating: 10

5 Judul Buku dengan Rating Tertinggi:
1984
Frankenstein
Ronggeng

Masukkan rating yang dicari: 9

Buku dengan Rating 9:
Judul: Frankenstein, Penulis: Shelley, Penerbit: Gramedia, Tahun: 1831, Rating: 9
PS D:\Alpro2Golang>
```

## Deskripsi Program

Program ini merupakan implementasi untuk mengelola data buku dalam sebuah perpustakaan. Program menyediakan fitur untuk mendaftarkan data buku, menampilkan buku dengan rating tertinggi,



mengurutkan buku berdasarkan rating, menampilkan lima buku dengan rating tertinggi, dan mencari buku berdasarkan rating tertentu.

Fungsi **DaftarkanBuku** bertugas untuk menginput data buku baru ke dalam daftar pustaka. Fungsi ini menerima parameter berupa pointer ke array daftar buku (**pustaka**) dan pointer ke jumlah buku (**n**). Pengguna akan diminta untuk memasukkan informasi buku seperti ID, judul, penulis, penerbit, jumlah eksemplar, tahun terbit, dan rating.

Fungsi **CetakFavorit** digunakan untuk menampilkan buku-buku dengan rating tertinggi dari pustaka. Fungsi ini pertama-tama mencari nilai rating maksimum, kemudian mencetak semua buku yang memiliki rating tersebut.

Fungsi **UrutBuku** bertugas mengurutkan daftar buku berdasarkan rating dari yang tertinggi ke terendah. Fungsi ini menggunakan metode insertion sort untuk melakukan pengurutan.

Fungsi **Cetak5Terbaru** akan mencetak lima judul buku dengan rating tertinggi setelah daftar pustaka diurutkan. Jika jumlah buku kurang dari lima, maka hanya buku yang tersedia yang akan dicetak.

Fungsi **CariBuku** memungkinkan pengguna untuk mencari buku berdasarkan rating tertentu. Jika ditemukan buku dengan rating yang dicari, informasi lengkap mengenai buku tersebut akan dicetak. Jika tidak ada buku dengan rating tersebut, program akan menampilkan pesan bahwa tidak ada buku yang cocok.

Pada fungsi **main**, program dimulai dengan meminta pengguna untuk memasukkan jumlah buku yang akan dimasukkan ke dalam daftar (**n226**). Pengguna kemudian diminta untuk mengisi data setiap buku menggunakan fungsi **DaftarkanBuku**. Setelah itu, program mencetak buku terfavorit menggunakan fungsi **CetakFavorit**, mengurutkan buku berdasarkan rating menggunakan fungsi **UrutBuku**, dan mencetak lima buku dengan rating tertinggi melalui fungsi **Cetak5Terbaru**. Akhirnya, pengguna dapat mencari buku dengan rating tertentu menggunakan fungsi **CariBuku**. Program memberikan hasil berupa daftar buku sesuai dengan fitur yang dijalankan.