LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 12

PENGURUTAN DATA

Oleh:

Ervan hapiz

2311102206

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 ← 1	i = 1
2	while i <= n-1 do	for i <= n-1 {
3	idx_min ← i - 1	idx_min = i - 1
4	j ← 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 ← j	idx_min = j
8	endif	}
9	j ← j + 1	j = j + 1
10	endwhile	}
11	t ← a[idx_min]	t = a[idx_min]
12	a[idx_min] ← a[i-1]	a[idx_min] = a[i-1]
13	a[i-1] ← t	a[i-1] = t
14	i ← 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 asceding 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 asceding 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 ← 1`	`i = 1`
2	`while i <= n-1 do`	`for i <= n-1 {`
3	`    j ← i`	`    j = i`
4	`    temp ← 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] ← a[j-1]`	`        a[j] = a[j-1]`
7	`        j ← j - 1`	`        j = j - 1`
8	`    endwhile`	`    }`
9	`    a[j] ← temp`	`    a[j] = temp`
10	`    i ← 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

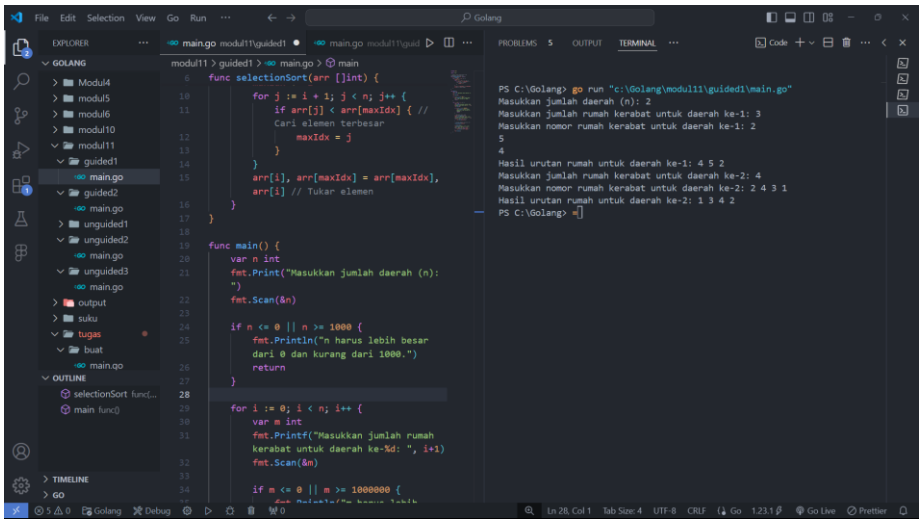
1. Guided 1

Source Code

```
package main

import "fmt"

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

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

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

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

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

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

```
                fmt.Scan(&houses[j])
            }

            // Urutkan dengan selection sort
            selectionSort(houses)

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

Screenshot



Deskripsi

Program ini adalah Program untuk mengurutkan nomer rumah kerabat dengan selectionsort. Dalam program terdapat fungsi selection sort untuk mengurutkan nilai dari terkecil. Pertama program akan menerima input dari user untuk menginput jumlah daerah. Kemudian user Kembali memasukan nilai variable m sebagai jumlah rumah keras. Jika inputan kurang dari nol maka program akan keluar. Selanjutnya program akan meminta user untuk menginpu t nomer rumah kerabat di daerah ke- i. setelah itu inputan akan dimasukan ke dalam slince. Setelah itu program memanggil fungsi SelectionSort untuk mengurutkan nilai nomer rumah kerabat. Kemudian akan ditampilkan nomer kerabat rumah dengan data urut. Kemudian program akan melanjutkan ke daerah selanjutnya sesuai banyaknya inputan user.

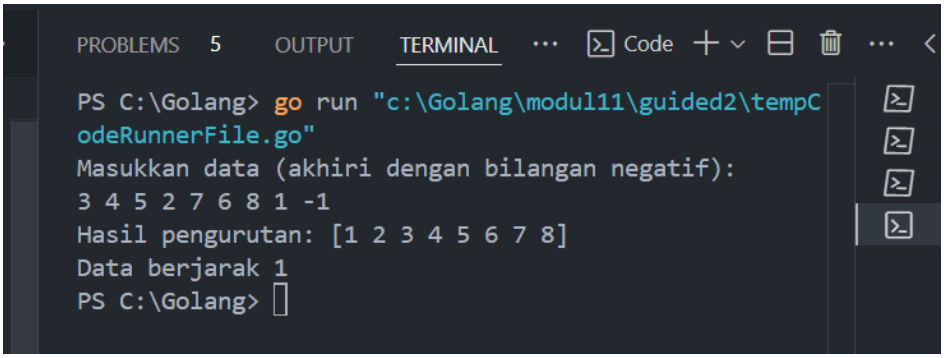
Guided 2

Source Code

```
package main

import (
    "fmt"
    "math"
)

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

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

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

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

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

    return true, diff
}

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

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

        // Urutkan data menggunakan insertion sort
        insertionSort(data)

        // Periksa apakah data berjarak tetap
        isConsistent, diff :=
    isDataConsistentlySpaced(data)

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

Screenshot



Deskripsi

Program ini adalah program untuk mengurutkan data yang diinput dan melihat jarak antara bilangan yang sudah di urutkan. Terdapat fungsi Insertion sort yang digunakan untuk mengurutkan data. Pertama program akan menerima input data berulang hingga inputan kurang dari nol. Kemudian terdapat juga fungsi untuk menentukan jarak antara bilangan, jika jarak antara bilangan tetap maka program akan menampilkan jarak berupa bilangan bulat. Setelah menginput program akan memanggil fungsi Insertion sort dan data akan terurut. Kemudian program

memanggil fungsi untuk menghitung jarak antar bilangan. Dan program akan menampilkan data yang sudah terurut dan jarak antar bilangan.

III. UNGUIDED

1. Unguided 1

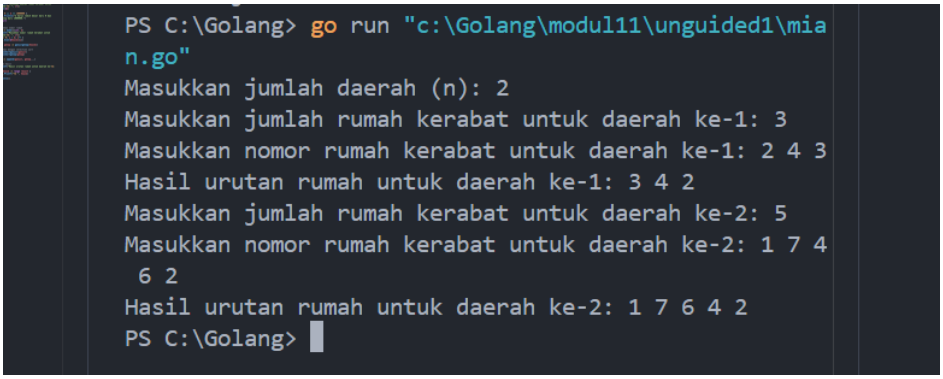
Source Code

```
package main

import "fmt"

// Fungsi untuk mengurutkan array menggunakan
selection sort
func ganjilgenap(arr []int) ([]int, []int) {
    n := len(arr)
    var ganjil []int
    var genap []int
    for i := 0; i < n; i++ {
        if arr[i]%2 == 0 {
            genap = append(genap, arr[i])
        } else {
            ganjil = append(ganjil, arr[i])
        }
    }
    return ganjil, genap
}

func selectionSortganjil(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 selectionSortgenap(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])
            }
            ganjil, genap := ganjilgenap(houses)

            // Urutkan dengan selection sort
            selectionSortganjil(ganjil)
            selectionSortgenap(genap)

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

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

```
            }
        }
```

Screenshot



Deskripsi

Program ini adalah program untuk mengurutkan nomer rumah kerabat pada daerah. Pada program ini terdapat fungsi selection sort untuk mengurut kan data. Pada program ini pengurutan dilakukan dengan bilangan ganjil terlebih dahulu secara ascending dan dilanjutkan bilangan genap dengan urutan descending. Terdapat juga fungsi untuk memisahgkan bilangan ganjil dan genap. Pertama program akan menrima input seperti program pada guided 1 namun pada program ini, bilangan ganjil dan bilangan genap dipisahkan dengan memanggil fungsi ganjilgenap. Kemudian bilangan ganjil akan diurutkan secara ascending kemudian dilanjutkan dengan bilangan genap secara discending. Kemudian program akan menampilkan program dengan data nomer rumah kerabat.

2. **Unguided 2**

Source Code

```
package main

import "fmt"

func selectionSortgenap(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 arr []int
            var arr_Baru []int
            var n int

            var median int
            for {
                fmt.Scan(&n)
                arr = append(arr, n)
                if n == -5313 {
                    break
                }
            }
            panjang := len(arr)

            for i := 0; i < panjang; i++ {
                if arr[i] == 0 {
                    selectionSortgenap(arr_Baru)
                    if len(arr_Baru)%2 == 0 {

                        median =
        (arr_Baru[(len(arr_Baru)/2)-1] +
        arr_Baru[len(arr_Baru)/2]) / 2

                    } else {

                        fmt.Println("gan", len(arr_Baru))
                        median =
        arr_Baru[len(arr_Baru)/2]
                        fmt.Print()
                    }
                    fmt.Println(median)
                } else {
                    arr_Baru = append(arr_Baru, arr[i])
                }
            }

        }
```

Screenshot

Deskripsi

Program ini adalah program yang mencari median dari data yang sudah diinput. Pada program data akan mengurutkan dan mencari median data ketika inputan 0. Dan program akan berakhir ketikan inputan -5313. Pertama user menginput bilangan bulat hingga -5313. Kemudian program akan melakukan pengurutan dan mencari median ketika data sama dengan 0. Kemudian ketikan inputan belum -5313 program akan melakukan hal yang sama ketikan menemukan data 0 sehingga ketika 0 dua maka median yang ditampilkan sebanyak dua. Jadi program akan menampilkan median sebanyak dua buah median.

3. **Unguided 1**

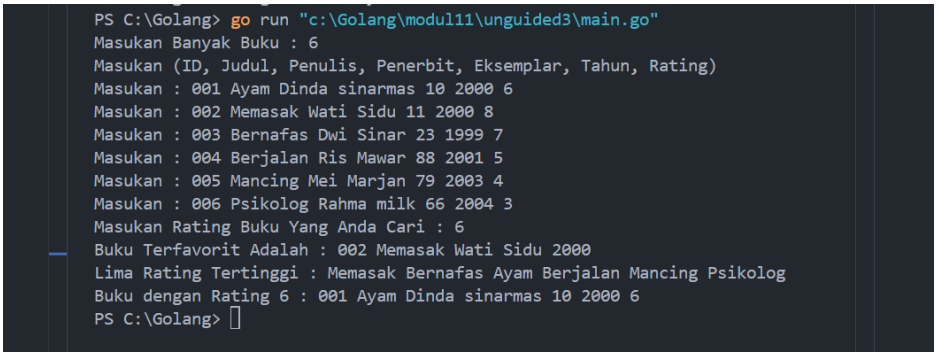
Source Code

```
package main

import "fmt"

const nmax = 7919

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

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

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

func FavoritBuku(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) {
        insertionSort(&Pustaka, n)
        fmt.Print("Lima Rating Tertinggi : ")
        for i := 0; i < n; i++ {
            fmt.Print(Pustaka[i].Judul, " ")

        }
        fmt.Println()
    }

    func BinarySearch(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 // Target ditemukan
            } else if Pustaka[mid].Ranting < target {
                low = mid + 1 // Cari di sisi kanan
            } else {
                high = mid - 1 // Cari di sisi kiri
            }
        }

        return -1 // Target tidak ditemukan
    }
    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)

        FavoritBuku(Buku, n)
```

```
        LimaRating(Buku, n)
        Temukan := BinarySearch(Buku, n, Cari)
        if Temukan != -1 {
            fmt.Printf("Buku dengan Rating %v : %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



Deskripsi

Program adalah program perpustakaan. Dalam progam akan menampilkan buku terfavorit, lima rating tertinggi dan mencari buku dengan nilai rating. Pertama program akan meminta user untuk menginput banyak buku. Kemudian user menginput ID, judul, penulis, penerbit, eksemplar, Tahun, Rating. Setelah itu User akan menginput rating buku yang akan dicari. Program akan memanggil fungsi FavoritBuku untuk menampilkan buku terfavorit berdasarkan reting tertinggi. Fungsi berjalan sesuai dengan logika nilai ekstrim. Selanjutnya program memanggil fungsi LimaRating untuk menampilkan 5 judul buku dengan rating dari tinggi ke rendah sebanyak 5. Dalam fungsi Lima Rating terdapat ppemanggilan insertionsort yang mengurutkan rating buku secara descenting dari terbesar ke terkecil kemudian ditampilkan 5 judul buku. Setelah itu pemanggilan fungsi binary searching untuk mencari buku berdasarkan rating. Kemudian program akan menampilkan buku dengan rating yang di cari.