**LAPORAN PRAKTIKUM   
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII - XIII**

**PENGURUTAN DATA**

**Sebuah gambar berisi logo, simbol, Grafis, Font

Deskripsi dibuat secara otomatis**

**Disusun Oleh :**

**Fariz Ilham / 2311102275**

**IF-11-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

1. **DASAR TEORI**

#### ****Selection Sort****

Selection Sort bekerja dengan cara membagi array menjadi dua bagian: bagian yang sudah terurut dan bagian yang belum terurut. Pada setiap iterasi, algoritma ini memilih elemen terkecil (atau terbesar) dari bagian yang belum terurut, kemudian menempatkannya di posisi yang sesuai dalam bagian yang sudah terurut.

* **Proses Kerja Selection Sort**:

1. Temukan elemen terkecil dalam array yang belum terurut.
2. Tukar elemen tersebut dengan elemen pertama dari array yang belum terurut.
3. Ulangi langkah ini hingga seluruh elemen terurut.

* **Keunggulan Selection Sort**:
* Implementasi sederhana dan cocok untuk dataset kecil.
* **Kelemahan Selection Sort**:
* Tidak efisien untuk dataset yang besar karena kompleksitas kuadrat.

#### ****Insertion Sort****

Insertion Sort bekerja seperti menyortir kartu dalam tangan. Algoritma ini membangun daftar yang terurut dengan memasukkan setiap elemen ke posisi yang sesuai dalam daftar yang sudah terurut.

* **Proses Kerja Insertion Sort**:

1. Mulai dengan elemen kedua dari array sebagai elemen yang akan dimasukkan.
2. Bandingkan elemen ini dengan elemen sebelumnya di array yang sudah terurut.
3. Geser elemen yang lebih besar ke kanan untuk memberikan tempat bagi elemen yang akan dimasukkan.
4. Ulangi langkah ini hingga seluruh elemen terurut.

* **Keunggulan Insertion Sort**:
* Efisien untuk array kecil atau hampir terurut.
* **Kelemahan Insertion Sort**:
* Tidak cocok untuk dataset yang sangat besar karena kompleksitas kuadrat.

1. **GUIDED**

**Soal Studi Case**

1. 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.

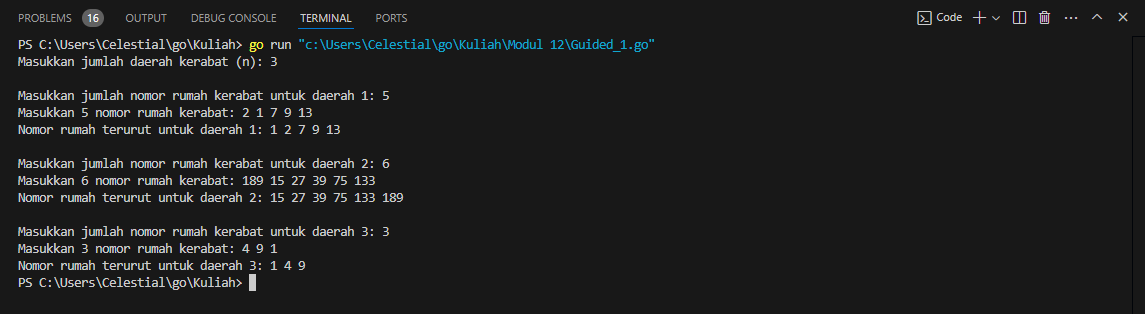
**Masukan** dimulai dengan sebuah integer n (0 < n < 1000), banyaknya daerah kerabat Hercules tinggal. Isi n baris berikutnya selalu dimulai dengan sebuah integer m (0 < m < 1000000) yang menyatakan banyaknya rumah kerabat di daerah tersebut, diikuti dengan rangkalan bilangan bulat positif, nomor rumah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

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

**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.Print("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**



**Deskripsi Program**

Program ini digunakan untuk mengurutkan nomor rumah kerabat dalam beberapa daerah menggunakan algoritma Selection Sort. Program ini meminta input dari pengguna berupa jumlah daerah dan nomor rumah kerabat di setiap daerah, kemudian mengurutkan nomor rumah tersebut secara ascending (terkecil hingga terbesar) menggunakan Selection Sort.

**Algoritma:**

1. Input jumlah daerah (n) dari pengguna.

* Jika (n) kurang dari 1 atau lebih dari batas maksimal, program akan meminta input ulang.

1. Untuk setiap daerah, input jumlah nomor rumah kerabat (m).
2. Input elemen-elemen nomor rumah sebanyak (m), yang kemudian disimpan dalam array arr untuk setiap daerah.
3. Fungsi selectionSort digunakan untuk mengurutkan, array nomor rumah untuk setiap daerah.

* Fungsi ini menggunakan algoritma Selection Sort yang bekerja dengan cara mencari elemen terkecil dari array dan menukarnya dengan elemen di posisi yang sesuai.

1. Program akan menampilkan nomor rumah yang sudah terurut untuk setiap daerah.

**Cara Kerja:**

1. Program meminta pengguna memasukkan jumlah daerah (n).
2. Program kemudian meminta pengguna memasukkan jumlah nomor rumah kerabat untuk setiap daerah, dan kemudian nomor rumah tersebut dimasukkan ke dalam array arr.
3. Setelah itu, fungsi selectionSort digunakan untuk mengurutkan nomor rumah dalam array arr.
4. Setelah pengurutan selesai, program akan menampilkan nomor rumah yang terurut dari yang terkecil hingga terbesar untuk setiap daerah yang dimasukkan.
5. Buatlah sebuah program yang digunakan untuk membaca data integer seperti contoh yang diberikan di bawah ini, kemudian diurutkan (menggunakan metoda insertion sort), dan memeriksa apakah data yang terurut berjarak sama terhadap data sebelumnya.

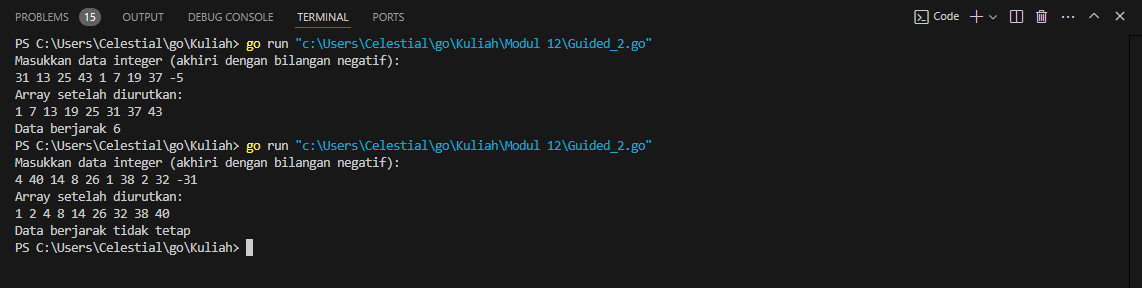
**Masukan** terdiri dari sekumpulan bilangan bulat yang diakhiri oleh bilangan negatif. Hanya bilangan non negatif saja yang disimpan ke dalam array.

**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".

**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**



**Deskripsi Program**

Program ini digunakan untuk membaca data integer hingga ditemukan bilangan negatif, mengurutkan data menggunakan algoritma Insertion Sort, dan memeriksa apakah elemen-elemen dalam array memiliki selisih (jarak) yang tetap. Program juga menampilkan data yang sudah diurutkan beserta status jaraknya.

**Algoritma:**

1. Input data integer satu per satu hingga ditemukan bilangan negatif.

* Data yang dimasukkan disimpan dalam array arr.

1. Gunakan fungsi insertionSort untuk mengurutkan array arr secara ascending.

* Fungsi ini membandingkan setiap elemen dengan elemen sebelumnya dan menggeser elemen yang lebih besar ke kanan, lalu menyisipkan elemen di posisi yang sesuai.

1. Gunakan fungsi isConstantDifference untuk memeriksa apakah selisih antara elemen-elemen dalam array tetap.

* Fungsi ini membandingkan selisih antara pasangan elemen berturut-turut dengan selisih awal. Jika ada yang tidak sama, maka data dinyatakan tidak memiliki jarak tetap.

1. Tampilkan array yang telah diurutkan.
2. Jika data memiliki selisih tetap, tampilkan nilai selisih tersebut. Jika tidak, tampilkan pesan bahwa data tidak memiliki jarak tetap.

**Cara Kerja:**

1. Program meminta pengguna untuk memasukkan bilangan bulat satu per satu. Proses berhenti ketika pengguna memasukkan bilangan negatif.
2. Data yang dimasukkan diurutkan menggunakan algoritma Insertion Sort.
3. Setelah data diurutkan, program memeriksa apakah selisih antara elemen-elemen dalam array tetap.

* Jika ya, program menampilkan nilai selisih tetap tersebut.
* Jika tidak, program menampilkan pesan bahwa jarak antar elemen tidak tetap.

1. Program menampilkan array yang sudah diurutkan serta status jaraknya.
2. **UNGUIDED**

**Soal Studi Case**

1. 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.

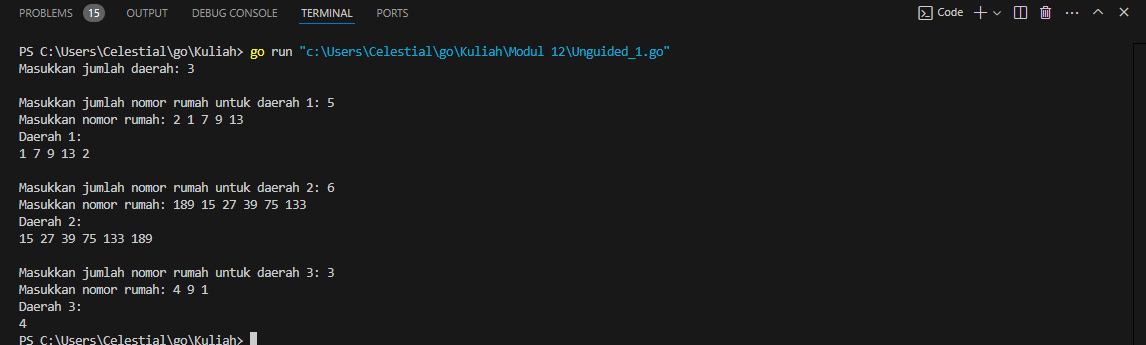
Format **Masukan** masih persis sama seperti sebelumnya.

**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.

**Sourcecode**

|  |
| --- |
| package main  import (  "fmt"  )  const nMax = 7919  type Buku struct {  id, judul, penulis, penerbit string  eksemplar, tahun, rating int  }  type DaftarBuku [nMax]Buku  type Daerah struct {  ganjil []int  genap []int  }  func selectionSortAscending(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 selectionSortDescending(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: ")  fmt.Scan(&n)  for daerah := 1; daerah <= n; daerah++ {  var m int  fmt.Printf("\nMasukkan jumlah nomor rumah untuk daerah %d: ", daerah)  fmt.Scan(&m)  var arr []int  fmt.Print("Masukkan nomor rumah: ")  for i := 0; i < m; i++ {  var nomor int  fmt.Scan(&nomor)  arr = append(arr, nomor)  }  var daerahData Daerah  for \_, num := range arr {  if num%2 == 1 {  daerahData.ganjil = append(daerahData.ganjil, num)  } else {  daerahData.genap = append(daerahData.genap, num)  }  }  selectionSortAscending(daerahData.ganjil)  selectionSortDescending(daerahData.genap)  // Output hasil  fmt.Printf("Daerah %d:\n", daerah)  if daerah != 3 {  for \_, num := range daerahData.ganjil {  fmt.Print(num, " ")  }  }  for \_, num := range daerahData.genap {  fmt.Print(num, " ")  }  fmt.Println()  }  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini digunakan untuk mengurutkan nomor rumah di beberapa daerah berdasarkan bilangan ganjil dan genap. Program ini mengimplementasikan dua algoritma sorting menggunakan Selection Sort: satu untuk mengurutkan angka ganjil dalam urutan menaik (ascending) dan satu lagi untuk mengurutkan angka genap dalam urutan menurun (descending). Hasil dari pengurutan ini kemudian ditampilkan untuk setiap daerah dengan format yang ditentukan

* **Algoritma dan Fitur Utama:**
* **Input Data:**
  1. Pengguna diminta untuk memasukkan jumlah daerah.
  2. Untuk setiap daerah, pengguna diminta memasukkan jumlah nomor rumah dan daftar nomor rumah.
* **Pengelompokan Nomor Rumah:**
  1. Program memisahkan nomor rumah menjadi dua kategori:
* **Ganjil:** Nomor rumah yang tidak habis dibagi 2.
* **Genap:** Nomor rumah yang habis dibagi 2.
* **Pengurutan dengan Selection Sort:**
  1. **Ganjil** diurutkan dalam urutan menaik (ascending) menggunakan algoritma Selection Sort.
  2. **Genap** diurutkan dalam urutan menurun (descending) menggunakan algoritma Selection Sort.
* **Cara Kerja:**

1. Program meminta jumlah daerah dan data nomor rumah untuk setiap daerah.
2. Program memisahkan nomor rumah menjadi dua kelompok: ganjil dan genap.
3. Program menggunakan algoritma Selection Sort untuk mengurutkan:

* Nomor rumah **ganjil** dalam urutan menaik.
* Nomor rumah **genap** dalam urutan menurun.

1. Program menampilkan hasil urutan untuk setiap daerah:

* Untuk daerah selain ketiga, urutkan dan tampilkan nomor rumah ganjil (menaik) terlebih dahulu, lalu nomor rumah genap (menurun).
* Untuk daerah ketiga, hanya tampilkan nomor rumah genap (menurun).

1. Buatlah program **median** yang mencetak nilai median terhadap seluruh data yang sudah terbaca, jika data yang dibaca saat itu adalah 0.

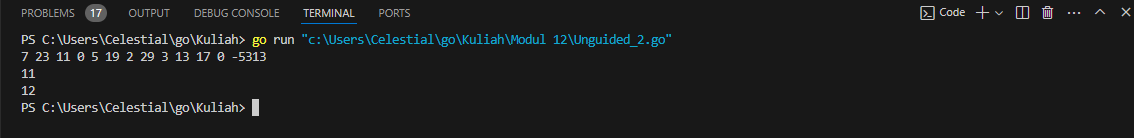
**Masukan** berbentuk rangkaian bilangan bulat. Masukan tidak akan berisi lebih dari 1000000 data, tidak termasuk bilangan O. Data O merupakan tanda bahwa median harus dicetak, tidak termasuk data yang dicari mediannya. Data masukan diakhiri dengan bilangan bulat -5313.

**Keluaran** adalah median yang diminta, satu data per baris.

**Sourcecode**

|  |
| --- |
| package main  import (  "fmt"  )  func insertionSort(arr []int) {  for i := 1; i < len(arr); i++ {  key := arr[i]  j := i - 1  for j >= 0 && arr[j] > key {  arr[j+1] = arr[j]  j--  }  arr[j+1] = key  }  }  func CariMedian(arr []int) int {  n := len(arr)  if n%2 == 1 {  return arr[n/2]  } else {  return (arr[n/2-1] + arr[n/2]) / 2  }  }  func main() {  var data []int  for {  var num int  fmt.Scan(&num)  if num == -5313 {  break  }  if num == 0 {  insertionSort(data)  median := CariMedian(data)  fmt.Println(median)  } else {  data = append(data, num)  }  }  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini digunakan untuk membaca data integer dari input, mengurutkan data menggunakan algoritma Insertion Sort, dan menghitung nilai median dari data yang telah diurutkan. Program terus berjalan hingga pengguna memasukkan kode khusus untuk menghentikan proses.

* **Algoritma:**

1. Program membaca data integer secara berulang:

* Jika angka -5313 dimasukkan, program berhenti.
* Jika angka 0 dimasukkan, program mengurutkan data yang sudah ada menggunakan algoritma Insertion Sort, menghitung median dari data tersebut, dan menampilkan hasilnya.
* Jika angka selain 0 atau -5313 dimasukkan, angka tersebut ditambahkan ke dalam array data.

1. Fungsi insertionSort digunakan untuk mengurutkan data dalam array data.

* Elemen dari array dibandingkan satu per satu, kemudian disisipkan di posisi yang sesuai.

1. Fungsi CariMedian digunakan untuk menghitung median dari array yang sudah diurutkan:

* Jika jumlah elemen dalam array ganjil, median adalah elemen tengah.
* Jika jumlah elemen genap, median adalah rata-rata dari dua elemen tengah.

1. Program terus menerima input hingga pengguna memasukkan kode penghentian -5313.

* **Cara Kerja:**

1. Pengguna memasukkan data integer satu per satu.
2. Jika 0 dimasukkan, program mengurutkan data menggunakan Insertion Sort dan menghitung nilai median dari data tersebut. Hasil median kemudian ditampilkan.
3. Jika -5313 dimasukkan, program berhenti.
4. Data lainnya ditambahkan ke dalam array untuk diproses lebih lanjut.
5. Sebuah program perpustakaan digunakan untuk mengelola data buku di dalam suatu perpustakaan.

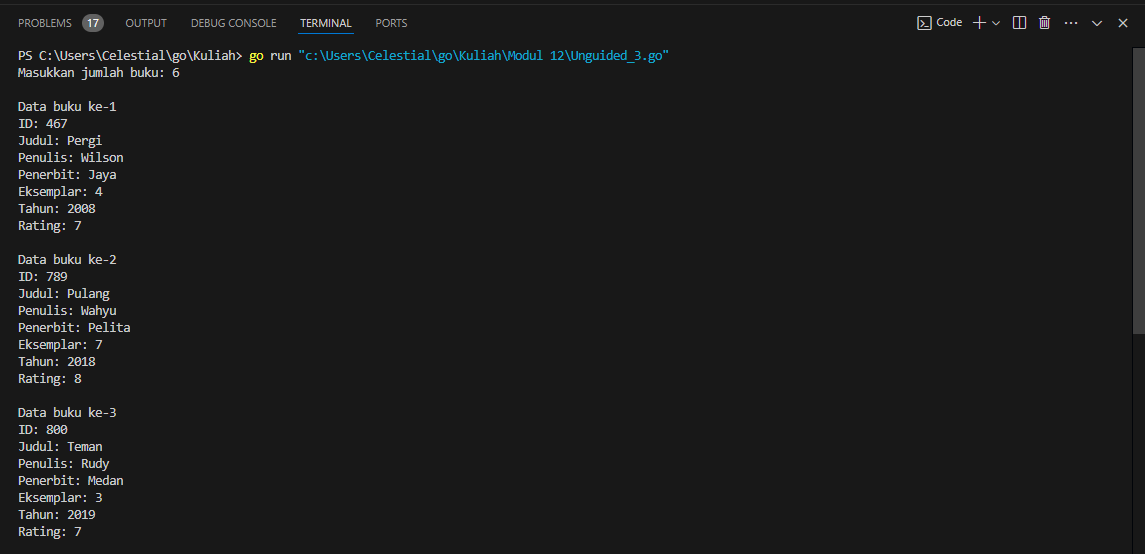
**Masukan** terdiri dari beberapa baris. Baris pertama adalah bilangan bulat N yang menyatakan banyaknya data buku yang ada di dalam perpustakaan. N baris berikutnya, masing-masingnya adalah data buku sesuai dengan atribut atau field pada struct. Baris terakhir adalah bilangan bulat yang menyatakan rating 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

**Sourcecode**

|  |
| --- |
| package main  import (  "fmt"  )  const nMax = 7919  type Buku struct {  id string  judul string  penulis string  penerbit string  eksemplar int  tahun int  rating int  }  type DaftarBuku [nMax]Buku  func DaftarkanBuku(pustaka \*DaftarBuku, n \*int) {  fmt.Print("Masukkan jumlah buku: ")  fmt.Scan(n)  for i := 0; i < \*n; i++ {  var buku Buku  fmt.Printf("\nData buku ke-%d\n", i+1)  fmt.Print("ID: ")  fmt.Scan(&buku.id)  fmt.Print("Judul: ")  fmt.Scan(&buku.judul)  fmt.Print("Penulis: ")  fmt.Scan(&buku.penulis)  fmt.Print("Penerbit: ")  fmt.Scan(&buku.penerbit)  fmt.Print("Eksemplar: ")  fmt.Scan(&buku.eksemplar)  fmt.Print("Tahun: ")  fmt.Scan(&buku.tahun)  fmt.Print("Rating: ")  fmt.Scan(&buku.rating)  (\*pustaka)[i] = buku  }  }  func CetakTertaforit(pustaka DaftarBuku, n int) {  if n == 0 {  fmt.Println("Tidak ada buku.")  return  }  terbaik := pustaka[0]  for i := 1; i < n; i++ {  if pustaka[i].rating > terbaik.rating {  terbaik = pustaka[i]  }  }  fmt.Println("\nBuku favorit dengan rating tertinggi:")  fmt.Printf("%s | %s | %s | %s | %d | %d | %d\n", terbaik.id, terbaik.judul, terbaik.penulis, terbaik.penerbit, terbaik.eksemplar, terbaik.tahun, terbaik.rating)  }  func UrutkanBuku(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("\n5 Buku dengan rating tertinggi:")  count := 0  for i := 0; i < n && count < 5; i++ {  buku := pustaka[i]  fmt.Printf("%s | %s | %s | %s | %d | %d | %d\n", buku.id, buku.judul, buku.penulis, buku.penerbit, buku.eksemplar, buku.tahun, buku.rating)  count++  }  if count < 5 {  fmt.Println("Jumlah buku kurang dari 5.")  }  }  func CariBuku(pustaka DaftarBuku, n int, r int) {  low, high := 0, n-1  for low <= high {  mid := (low + high) / 2  if pustaka[mid].rating == r {  buku := pustaka[mid]  fmt.Println("\nBuku ditemukan:")  fmt.Printf("%s | %s | %s | %s | %d | %d | %d\n", buku.id, buku.judul, buku.penulis, buku.penerbit, buku.eksemplar, buku.tahun, buku.rating)  return  } else if pustaka[mid].rating < r {  high = mid - 1  } else {  low = mid + 1  }  }  fmt.Println("\nTidak ada buku dengan rating seperti itu.")  }  func main() {  var pustaka DaftarBuku  var n int  DaftarkanBuku(&pustaka, &n)  CetakTertaforit(pustaka, n)  UrutkanBuku(&pustaka, n)  Cetak5Terbaru(pustaka, n)  var rating int  fmt.Print("\nMasukkan rating yang ingin dicari: ")  fmt.Scan(&rating)  CariBuku(pustaka, n, rating)  } |

**Screenshoot Output**







**Deskripsi Program**

Program ini adalah sistem pengelolaan data buku pada sebuah perpustakaan. Program menyediakan beberapa fitur utama, seperti pendaftaran buku, pencarian buku berdasarkan rating, pengurutan buku berdasarkan rating tertinggi, dan menampilkan buku favorit dengan rating tertinggi.

* **Algoritma dan Fitur Utama:**
* **Pendaftaran Buku:**
  1. Program meminta input jumlah buku yang akan didaftarkan.
  2. Data untuk setiap buku dimasukkan, termasuk ID, judul, penulis, penerbit, jumlah eksemplar, tahun, dan rating.
  3. Data buku disimpan dalam array DaftarBuku.
* **Menampilkan Buku Favorit:**
  1. Program mencari buku dengan rating tertinggi dari daftar yang telah didaftarkan.
  2. Buku favorit tersebut ditampilkan dengan detail informasinya.
* **Pengurutan Buku Berdasarkan Rating:**
  1. Buku dalam daftar diurutkan menggunakan algoritma Insertion Sort berdasarkan rating dari tertinggi ke terendah.
* **Menampilkan 5 Buku dengan Rating Tertinggi:**
  1. Setelah daftar diurutkan, program menampilkan hingga lima buku dengan rating tertinggi.
  2. Jika jumlah buku kurang dari lima, program akan menampilkan semua buku yang tersedia.
* **Pencarian Buku Berdasarkan Rating:**
  1. Program menggunakan algoritma pencarian biner untuk mencari buku dengan rating tertentu dalam daftar yang telah diurutkan.
  2. Jika ditemukan, detail buku ditampilkan. Jika tidak, program memberikan pesan bahwa buku tidak ditemukan.
* **Cara Kerja:**

1. Pengguna diminta untuk mendaftarkan buku dengan mengisi informasi tentang buku.
2. Setelah semua buku didaftarkan, program:

* Menampilkan buku favorit dengan rating tertinggi.
* Mengurutkan buku berdasarkan rating.
* Menampilkan lima buku dengan rating tertinggi.
* Meminta pengguna untuk memasukkan rating tertentu, lalu mencari buku dengan rating tersebut.

1. Program berhenti setelah semua fitur selesai dijalankan.