**LAPORAN PRAKTIKUM**

**ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XII & XIII**

**PENCARIAN NILAI EKSTRIM DAN HIMPUNAN DATA**

****

**DISUSUN OLEH :**

**WILDAN MAULANA ZIDAN**

**2311102162**

**IF 11 02**

**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

1. **DASAR TEORI**

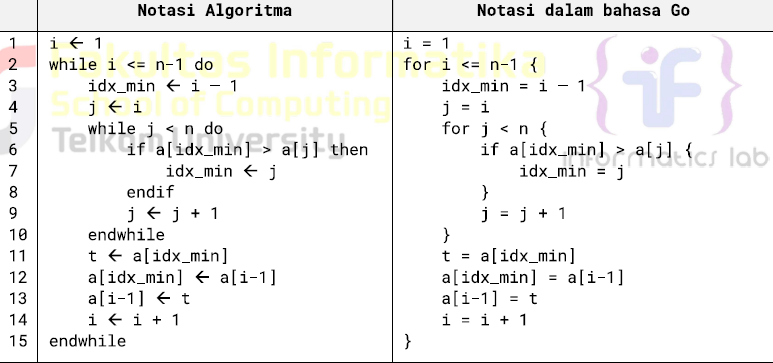
**12.1 Ide Algoritma Selection Sort**

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".

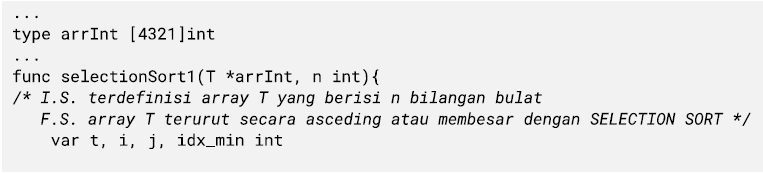
* + 1. Cari nilai terkecil di dalam rentang data tersisa
    2. Pindahkanltukar tempat dengan data yang berada pada posisi paling kiri pada rentang data tersisa tersebut.
    3. ulangi proses ini sampai tersisa hanya satu data saja.

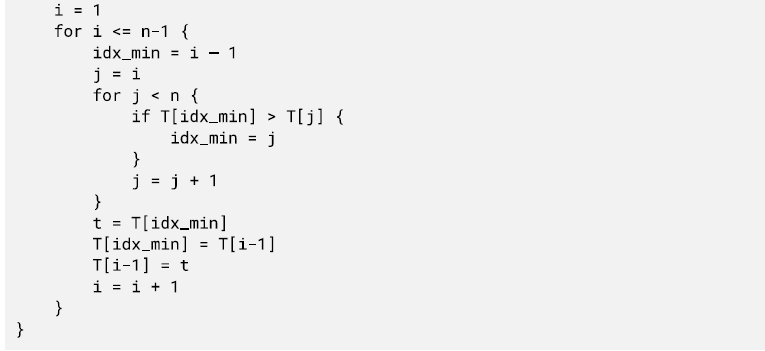
Algoritma ini dikenal juga dengan nama Selectton Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian indeks nilai ekstrim dan proses pertukaran dua nilai atau swap.



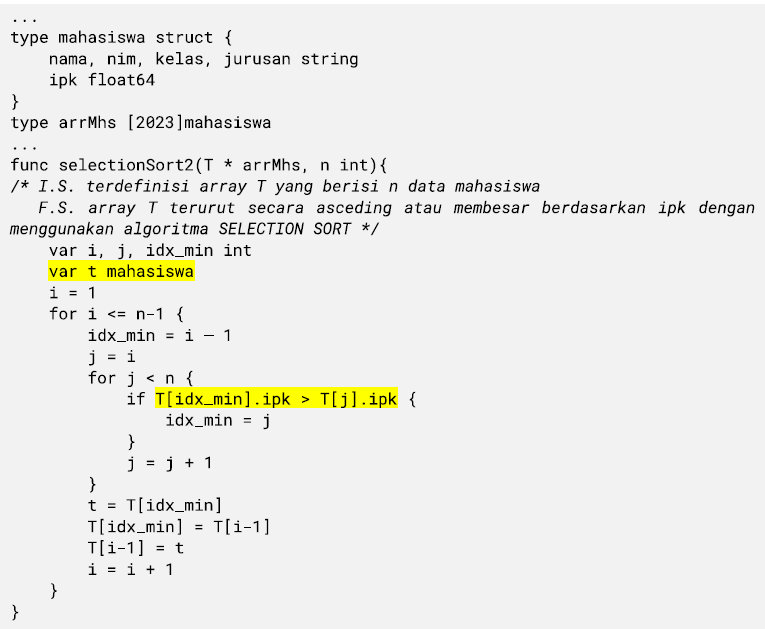
* **Algortima Selction Sort**

Adapun algoritma selection sort pada untuk mengurutkan array bertipe data bilangan bulat secara membesar atau ascending adalah sebagai berikut ini!





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.



**12.2 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".

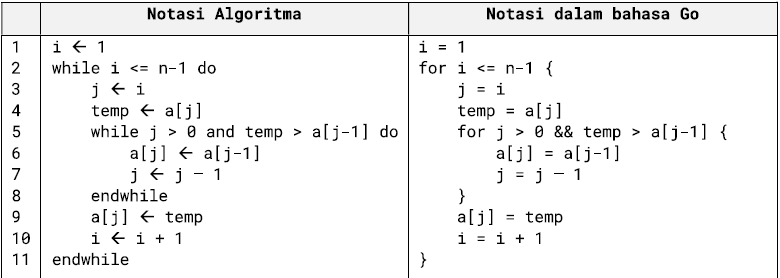
1. Untuk satu data yang belum terurut dan sejumlah data yang sudah diurutkan:

* 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 keturunan

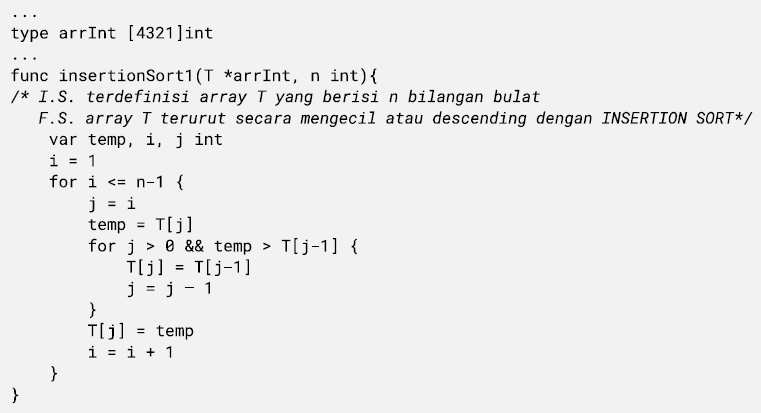
1. ulangi proses tersebut untuk setiap data yang belum terurut terhadap rangkaian data yang sudah terurut.

Algoritma ini dikenal juga dengan nama Insertion Sort, yang mana pada algoritma ini melibatkan dua proses yaitu pencarian sekuensial dan penyisipan.



* Algoritma Insertion Sort

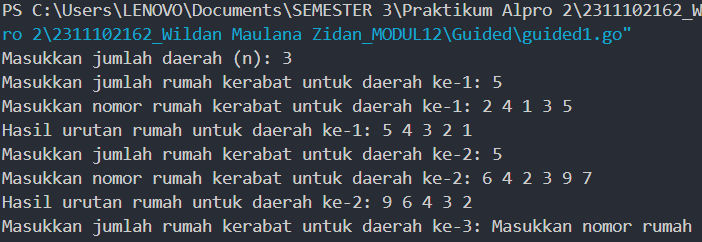
Adapun algoritma insertion sort pada untuk mengurutkan array bertipe data bilangan bulat secara mengecil atau descending adalah sebagai berikut ini!



1. **GUIDED**

Guided 1 Source code

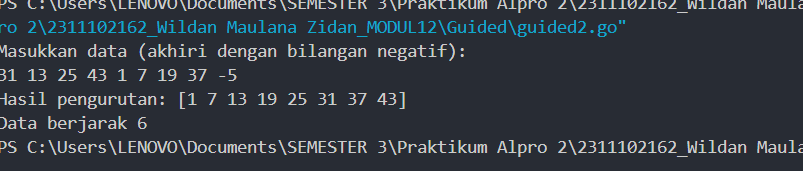
|  |
| --- |
| package main  import "fmt"  // Fungsi untuk mengurutkan array menggunakan selection sort  func selectionSort(arr []int) {  n := len(arr)  for i := 0; i < n-1; i++ {  maxIdx := i  for j := i + 1; j < n; j++ {  if arr[j] > arr[maxIdx] { // Cari elemen terbesar  maxIdx = j  }  }  arr[i],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



Kode di atas meminta pengguna memasukkan data jumlah daerah dan nomor rumah kerabat untuk setiap daerah, kemudian menggunakan algoritma pemilihan sort untuk mengurutkan nomor rumah tersebut. Pertama, pengguna diminta memasukkan jumlah daerah (n), kemudian memasukkan jumlah rumah (m) dan nomor rumah untuk setiap daerah. Program akan memeriksa validitas input untuk memastikan jumlah daerah dan rumah berada dalam batas yang ditentukan. Kemudian, nomor rumah akan diurutkan secara menurun menggunakan sort pilihan, dan hasil urutan akan ditampilkan untuk setiap daerah. Program ini berjalan untuk setiap area secara terpisah hingga semua input selesai diproses.

Guided 2 Source code

|  |
| --- |
| Kode di atas adalah program Go yang 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")  }  } |

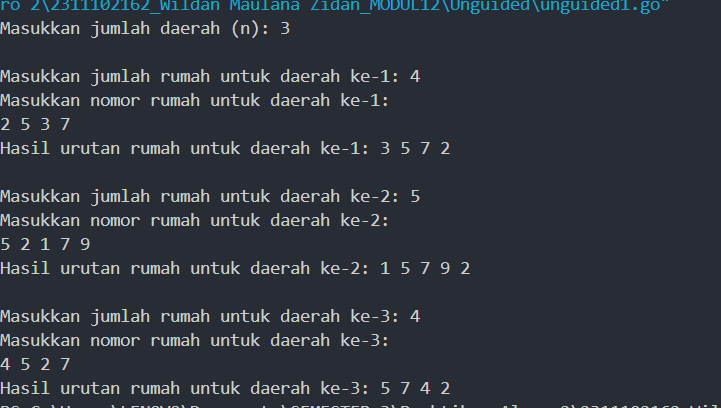


Kodi di atas meminta pengguna memasukkan serangkaian bilangan bulat (tidak memasukkan bilangan negatif) dan kemudian mengurutkan data menggunakan algoritma penambahan sort. Program memeriksa apakah selisih antar elemen dalam array, atau jarak tetap, konsisten setelah data diurutkan. Fungsi "isDataConsistentlySpaced" memeriksa jarak antara elemen berurutan dan memastikan bahwa jarak tersebut konsisten. Program akan mencetak perbedaan antar elemen jika jarak data tetap; jika tidak, program akan mencetak pesan bahwa jarak tidak tetap. Hasil pengurutan dan informasi tentang jarak antar elemen diberikan oleh program ini.

1. **UNGUIDED**

Unguided 1 Source code

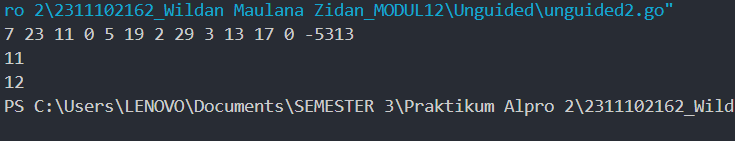
|  |
| --- |
| Kode di atas adalah program Go yang package main  import (  "fmt"  )  // Fungsi untuk mengurutkan array secara ascending menggunakan selection sort  func selectionSortAsc(arr []int) {  n := len(arr)  for i := 0; i < n-1; i++ {  minIdx := i  for j := i + 1; j < n; j++ {  if arr[j] < arr[minIdx] { // Cari elemen terkecil  minIdx = j  }  }  arr[i], arr[minIdx] = arr[minIdx], arr[i] // Tukar elemen  }  }  // 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] { // 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("\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()  }  } |



Kode di atas dapat digunakan untuk mengurutkan nomor rumah di area tertentu berdasarkan kriteria tertentu. Jumlah daerah (n) dan jumlah rumah di setiap daerah harus dimasukkan, kemudian nomor rumah masing-masing. Program akan berhenti jika input tidak valid (misalnya, \(n = 0 \) atau \(n = 1000 \). Untuk setiap daerah, nomor rumah dibagi menjadi bilangan ganjil dan genap. Nomor ganjil diurutkan naik dengan menggunakan "pilihan sort", dan nomor genap diurutkan turun dengan metode yang sama. Hasil akhir ditampilkan dengan format nomor ganjil terlebih dahulu, diikuti nomor genap untuk setiap daerah. Selain menggunakan struktur iteratif untuk mengelola berbagai wilayah, program ini menjamin validitas input.

Unguided 2 Source code

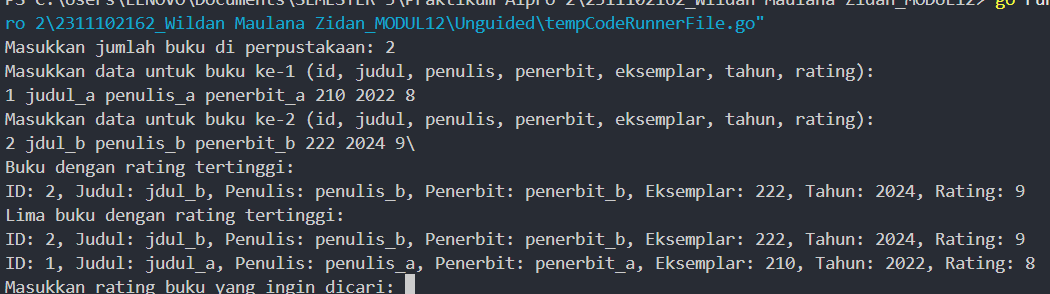
|  |
| --- |
| Kode di atas adalah program Go yang package main  import "fmt"  func insertionSort(arr []int) {  n := len(arr)  for i := 1; i < n; i++ {  key := arr[i]  j := i - 1  // Geser elemen yang lebih besar dari key ke kanan  for j >= 0 && arr[j] > key {  arr[j+1] = arr[j]  j--  }  arr[j+1] = key  }  }  func median(arr []int) int {  n := len(arr)  if n%2 == 0 {  return (arr[(n/2)-1] + arr[(n/2)]) / 2  } else {  return arr[(n / 2)]  }  }  func main() {  var input int  var kumpulanAngka = make([]int, 0)  var angkaSliced = make([]int, 0)  for input != -5313 {  fmt.Scan(&input)  if input != -5313 {  kumpulanAngka = append(kumpulanAngka, input)  }  }  for \_, angka := range kumpulanAngka {  if angka == 0 {  insertionSort(angkaSliced)  fmt.Println(median(angkaSliced))  } else {  angkaSliced = append(angkaSliced, angka)  }  }  } |



Kode di atas untuk menghitung dan mencetak median dari sekumpulan angka yang dimasukkan secara dinamis. Sampai input \*sentinel\* "-5313" diberikan, pengguna terus memasukkan angka. Setiap kali angka "0" dimasukkan, program menghitung median dari semua angka yang telah dimasukkan sebelumnya, termasuk angka "0", dengan menggunakan "insertion sort" dan menggunakan fungsi "median" untuk menemukan nilai median. Setelah menemukan median, proses pengumpulan angka dilanjutkan, sementara angka "0" hanya digunakan untuk memulai perhitungan tanpa disimpan. Perhitungan median subset angka dapat dilakukan oleh program dalam satu sesi input.

Unguided 3 Source code

|  |
| --- |
| 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)  } |



Kode di atas digunakan untuk mengelola data buku di perpustakaan. Pengguna diminta untuk memasukkan jumlah buku (n) dan detail setiap buku, termasuk judul, penulis, penerbit, eksemplar, tahun, dan rating, melalui program. Data buku disimpan dalam slice. Program dapat menambahkan data buku, mencetak buku dengan rating tertinggi, menggunakan "urutkan masuk" untuk mengurutkan buku berdasarkan rating secara descending, mencetak lima buku terbaik berdasarkan rating, dan mencari buku dengan rating tertentu. Setelah data diolah, program menghasilkan daftar buku favorit, daftar lima buku terbaik, dan hasil pencarian buku berdasarkan rating yang dimasukkan. Selain itu, program memverifikasi bahwa jumlah buku berada dalam rentang 1 hingga 7919.