**LAPORAN PRAKTIKUM**

**ALGORITMA DAN PEMROGRAMAN 2**

**MODUL X**

**PENCARIAN NILAI EKSTRIM DAN HIMPUNAN DATA**

****

**DISUSUN OLEH :**

**WILDAN MAULANA ZIDAN**

**2311102162**

**IF 11 02**

**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

1. **DASAR TEORI**
2. **Ide Pencarian Nilai Max/Min**

Pencarian adalah suatu proses yang lazim dilakukan di dalam kehidupan sehari-hari. Contoh penggunaannya dalam kehidupan nyata sangat beragam, misalnya pencarian file di dalam directory komputer, pencarian suatu teks di dalam sebuah dokumen, pencarian buku pada rak buku, dan contoh Iainnya. Pertama pada modul ini akan dipelajari salah satu algoritma pencarian nilai terkecil atau terbesar pada sekumpulan data, atau biasa disebut pencarian nilai ekstrim.

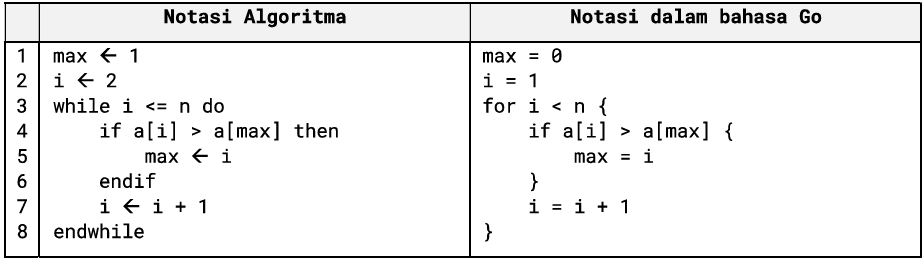
Ide algoritma sederhana sekali. Karena data harus diproses secara sekuensial, maka nilai atau indeks ke nilai maksimum dari data yang telah diproses disimpan untuk dibandingkan dengan data berikutnya. Nilai yang berhasil disimpan sampai algoritma tersebut berakhir adalah nilai maksimum yang dicari. Adapun algoritmanya secara umum adalah sebagai berikut:

1. Jadikan data pertama sebagai nilai ekstrim.
2. Lakukan validasi nilai ekstrim dari data kedua hingga data terakhir.

* Apabila nilai ekstrim tidak valid, maka update nilai ekstrims tersebut dengan data yang dicek.

1. Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid.

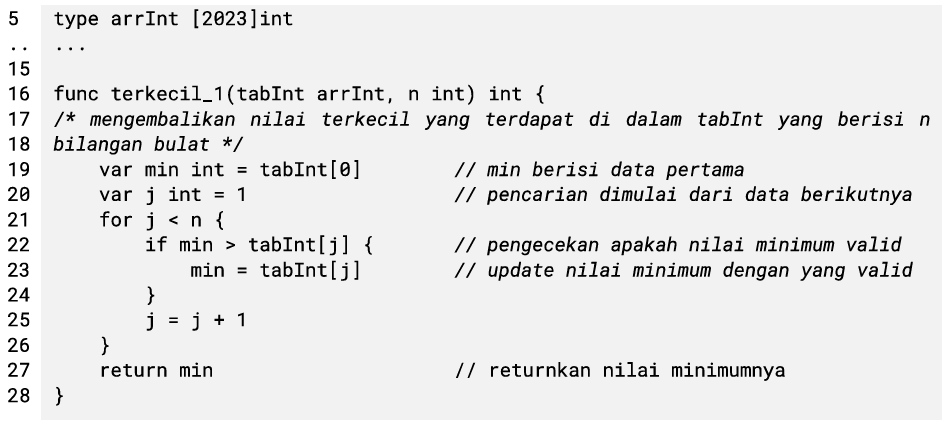
Berikut ini adalah notasi dalam pseudecode dan bahasa Go, misalnya untuk pencarian nilai terbesar atau maksimum:



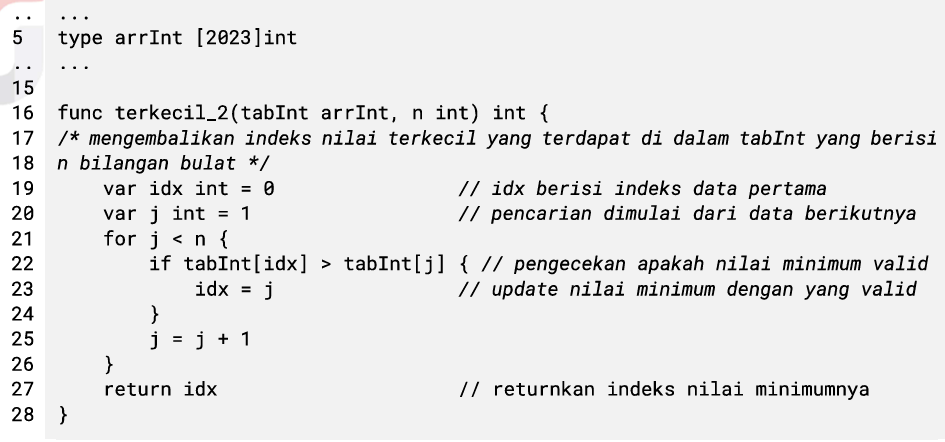
1. **Pencarian Nilai Ekstrim Pada Array Bertipe Data Dasar**

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array

terisisejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut.

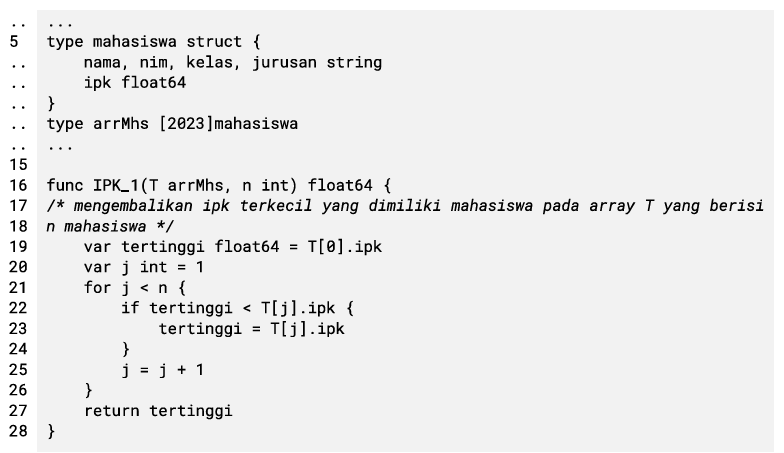
Perhatikan potongan program dalam bahasa Go berikut ini!

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada bahasa Go di mulai dari nol atau "0" seperti penjelasan pada modul 9. Selanjutnya, pada penjelasan di awal bab 3 telah disampaikan bahwa pada pencarian yang terpenting adalah posisi atau indeks dari nilai yang dicari dalam kumpulan data atau array. Oleh karena itu modifikasi pada program di atas dapat dilihat pada potongan program berikut ini!

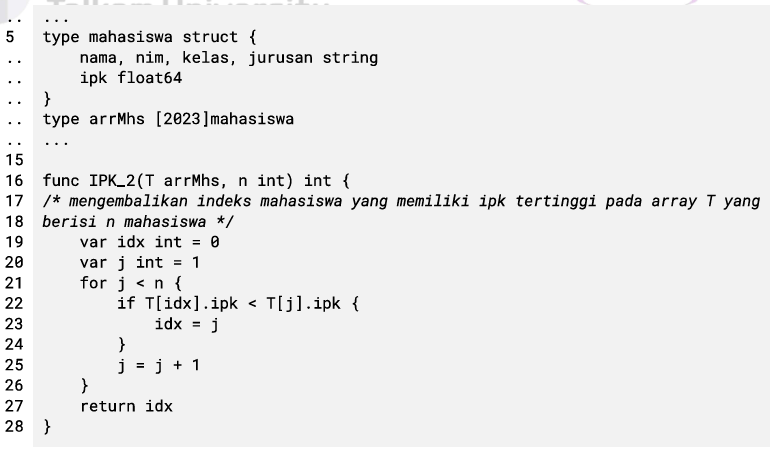


1. **Pencarian Nilai Ekstrim pada Array Bertipe Data Terstruktur**

Pada kasus yang lebih kompleks pencarian ekstrim dapat juga dilakukan, misalnya mencari data mahasiswa dengan nilai terbesar, mencari lagu dengan durasi terlama, mencari pembalap yang memiliki catatan waktu balap tercepat, dan sebagainya. Sebagai contoh misalnya terdapat array yang digunakan untuk menyimpan data mahasiswa, kemudian terdapat fungsi IPK yang digunakan untuk mencari data mahasiswa dengan IPK tertinggi.



Apabila diperhatikan potongan program di atas, maka kita akan memperoleh ipk tertinggi, tetapi kita tidak memperoleh identitas mahasiswa dengan ipk tertinggi tersebut. Maka seperti penjelasan yang sudah diberikan sebelumnya, maka pencarian yang dilakukan bisa mengembalikan indeks mahasiswa dengan ipk tertinggi tersebut. Berikut ini adalah modifikasinya!



Sehingga melalui algoritma di atas, identitas mahasiswa dapat diperoleh, misalnya T[idx].nama, T[idx]. Nim, T[idx]. Kelas, sehingga T[idx].jurusan.

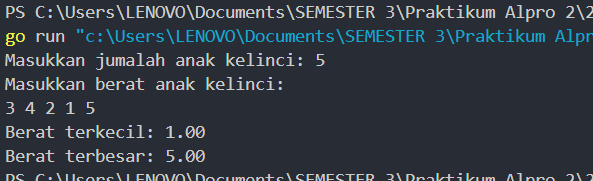
1. **GUIDED**

Guided 1

Source Code

|  |
| --- |
| package main  import (  "fmt"  )  func main() {  var N int  var berat [1000]float64  fmt.Print("Masukkan jumalah anak kelinci: ")  fmt.Scan(&N)  fmt.Println("Masukkan berat anak kelinci: ")  for i := 0; i < N; i++ {  fmt.Scan(&berat[i])  }  // Inisialisasi nilai min dan max dengan elemen pertama  min := berat[0]  max := berat[0]  for i := 1; i < N; i++ {  if berat[i] < min {  min = berat[i]  }  if berat[i] > max {  max = berat[i]  }  }  fmt.Printf("Berat terkecil: %.2f\n", min)  fmt.Printf("Berat terbesar: %.2f\n", max)  } |

Output

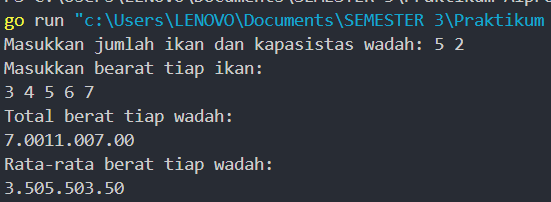


Kode di atas adalah menghitung jumlah anak kelinci (N) dan berat setiap anak kelinci, kemudian menemukan nilai terkecil (min) dan terbesar (max) dari data tersebut. Program ini menggunakan array "berat" untuk menyimpan berat setiap anak kelinci, dengan input dimasukkan oleh pengguna melalui "fmt.Scan". Nilai "min" dan "max" dimulai dengan elemen pertama array, dan kemudian diperbarui dengan perulangan " Di belakang koma, hasil akhir menunjukkan berat terkecil dan terbesar.

Guided 2

Source Code

|  |
| --- |
| package main  import (  "fmt"  )  func main() {  var x, y int  fmt.Print("Masukkan jumlah ikan dan kapasistas wadah: ")  fmt.Scan(&x, &y)  berat := make([]float64, x)  fmt.Println("Masukkan bearat tiap ikan:")  for i := 0; i < x; i++ {  fmt.Scan(&berat[i])  }  jumlahWadah := (x + y - 1) / y // Pembulatan keatas jika x tidak dibagi y  totalBeratWadah := make([]float64, jumlahWadah)  for i := 0; i < x; i++ {  indeksWadah := i / y  totalBeratWadah[indeksWadah] += berat[i]  }  // Output total berat tiap wadah  fmt.Println("Total berat tiap wadah: ")  for \_, total := range totalBeratWadah {  fmt.Printf("%.2f", total)  }  fmt.Println()  // Output rata-rata berat tiap wadah  fmt.Println("Rata-rata berat tiap wadah: ")  for \_, total := range totalBeratWadah {  rataRata := total / float64(y)  fmt.Printf("%.2f", rataRata)  }  fmt.Println()  } |

Output  
 

Kode di atas yang dapat digunakan untuk menghitung berat total dan rata-rata ikan dalam wadah dengan kapasitas tertentu. Program dimulai dengan meminta jumlah ikan (x) dan kapasitas wadah (y). Kemudian, masing-masing ikan dibaca ke dalam slice "berat" dan menggunakan rumus pembulatan ke atas "(x + y - 1) / y" untuk menghitung jumlah wadah yang diperlukan. Selanjutnya, slice "totalBeratWadah" digunakan untuk menyimpan berat total ikan dalam masing-masing wadah. Dengan menggunakan pembagian integer "i/y", berat ikan dialokasikan ke wadah berdasarkan indeksnya. Program kemudian mencetak berat total dan rata-rata setiap wadah. Hasil ditampilkan dalam bentuk dua angka desimal di belakang koma.

1. **UNGUIDED**

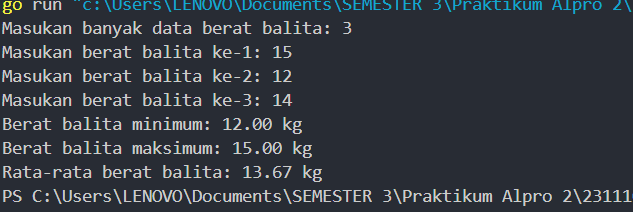
Unguided 1

Pos Pelayanan Terpadu (posyandu) sebagai tempat pelayanan kesehatan perlu mencatat data berat balita (dalam kg). Petugas akan memasukkan data tersebut ke dalam array. Dari data yang diperoleh akan dicari berat balita terkecil, terbesar, dan reratanya.

Source Code

|  |
| --- |
| package main  import (  "fmt"  )  type arrBalita [100]float64  func hitungMinMax(arrBerat arrBalita, n int, bMin, bMax \*float64) {  \*bMin = arrBerat[0]  \*bMax = arrBerat[0]  for i := 1; i < n; i++ {  if arrBerat[i] < \*bMin {  \*bMin = arrBerat[i]  }  if arrBerat[i] > \*bMax {  \*bMax = arrBerat[i]  }  }  }  func rataRata(arrBerat arrBalita, n int) float64 {  var total float64  for i := 0; i < n; i++ {  total += arrBerat[i]  }  return total / float64(n)  }  func main() {  var n int  var berat arrBalita  var bMin, bMax float64  fmt.Print("Masukan banyak data berat balita: ")  fmt.Scan(&n)  for i := 0; i < n; i++ {  fmt.Printf("Masukan berat balita ke-%d: ", i+1)  fmt.Scan(&berat[i])  }  hitungMinMax(berat, n, &bMin, &bMax)  rata := rataRata(berat, n)  fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)  fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)  fmt.Printf("Rata-rata berat balita: %.2f kg\n", rata)  } |

Output



Kode di atas yang digunakan untuk menghitung berat balita minimum, maksimum, dan rata-rata. Untuk menyimpan berat balita hingga 100 elemen, program menggunakan tipe data array "arrBalita". Fungsi "hitungMinMax" menerima array berat, jumlah data ("n"), dan pointer ke variabel "bMin" dan "bMax" untuk menghitung berat minimum dan maksimum melalui iterasi. Fungsi "rataRata" menghitung berat rata-rata dengan menjumlahkan semua elemen dalam array lalu membagi totalnya dengan jumla Pengguna dapat memanggil fungsi "main" setelah memasukkan jumlah balita dan beratnya. Akhirnya, hasil mencetak berat minimum, maksimum, dan rata-rata dalam bentuk dua angka di belakang koma.