**LAPORAN PRAKTIKUM   
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XI**

**PENCARIAN NILAI EKSTRIM PADA HIMPUNAN 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**

Pencarian nilai ekstrim adalah proses untuk menemukan nilai terbesar (maksimum) atau terkecil (minimum) dalam sekumpulan data. Dalam algoritma ini, elemen pertama dari data digunakan sebagai nilai ekstrim awal, yang kemudian divalidasi dengan membandingkan elemen-elemen berikutnya dalam data.

1. Inisialisasi: Nilai ekstrim dimulai dari elemen pertama.

2. Validasi dan Pembaruan: Elemen kedua hingga terakhir dibandingkan dengan nilai ekstrim. Jika ditemukan elemen yang lebih besar (maksimum) atau lebih kecil (minimum), nilai ekstrim diperbarui dengan elemen tersebut.

3. Finalisasi: Setelah semua elemen diperiksa, nilai ekstrim yang tersisa dianggap valid.

Pencarian selesai ketika seluruh elemen data telah diperiksa, menghasilkan nilai ekstrim yang benar sesuai kriteria pencarian.

1. **GUIDED**

**Soal Studi Case**

1. Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // Mendeklarasikan tipe data array arrInt dengan panjang 2023  type arrInt [2023]int  // Fungsi untuk mencari indeks elemen terkecil dalam array  func terkecil(tabInt arrInt, n int) int {  var idx int = 0 // idx menyimpan indeks elemen terkecil  var j int = 1  for j < n {  if tabInt[idx] > tabInt[j] {  idx = j // Simpan indeks j jika elemen di indeks j lebih kecil  }  j = j + 1  }  return idx  }  // Fungsi main untuk menguji fungsi terkecil  func main() {  var n int  var tab arrInt  // Meminta input jumlah elemen array  fmt.Print("Masukkan jumlah elemen (maks 2023): ")  fmt.Scan(&n)  // Validasi input jumlah elemen  if n < 1 || n > 2023 {  fmt.Println("Jumlah elemen harus antara 1 dan 2023.")  return  }  // Memasukkan elemen-elemen array  fmt.Println("Masukkan elemen-elemen array:")  for i := 0; i < n; i++ {  fmt.Print("Elemen ke-", i+1, ": ")  fmt.Scan(&tab[i])  }  // Memanggil fungsi terkecil untuk menemukan indeks elemen terkecil  idxMin := terkecil(tab, n)  // Menampilkan nilai dan indeks terkecil  fmt.Println("Nilai terkecil dalam array adalah:", tab[idxMin], "pada indeks:", idxMin)  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini digunakan untuk mencari dan menampilkan nilai terkecil dalam array serta indeks dari nilai terkecil tersebut. Program ini memanfaatkan array dengan tipe data khusus dan mengimplementasikan fungsi untuk menemukan indeks elemen terkecil dalam array.

* Algoritma:

1. Input jumlah elemen array ( n ) dari pengguna.

* Jika ( n ) kurang dari 1 atau lebih dari 2023, program menampilkan pesan kesalahan dan berhenti.

2. Input elemen-elemen array sebanyak ( n ) dari pengguna, yang disimpan dalam array `tab`.

3. Fungsi `terkecil` digunakan untuk mencari indeks dari elemen terkecil dalam array `tab`.

* Fungsi ini memulai perbandingan dari elemen pertama dan membandingkannya dengan elemen berikutnya.
* Jika ditemukan elemen yang lebih kecil, indeksnya disimpan.

4. Program menampilkan nilai terkecil dan indeksnya di array.

* Cara Kerja:

1. Program meminta pengguna memasukkan jumlah elemen dalam array ( n ).

2. Program kemudian meminta pengguna memasukkan elemen-elemen array satu per satu dan menyimpannya dalam array `tab`.

3. Fungsi `terkecil` digunakan untuk mencari indeks dari elemen terkecil dalam array.

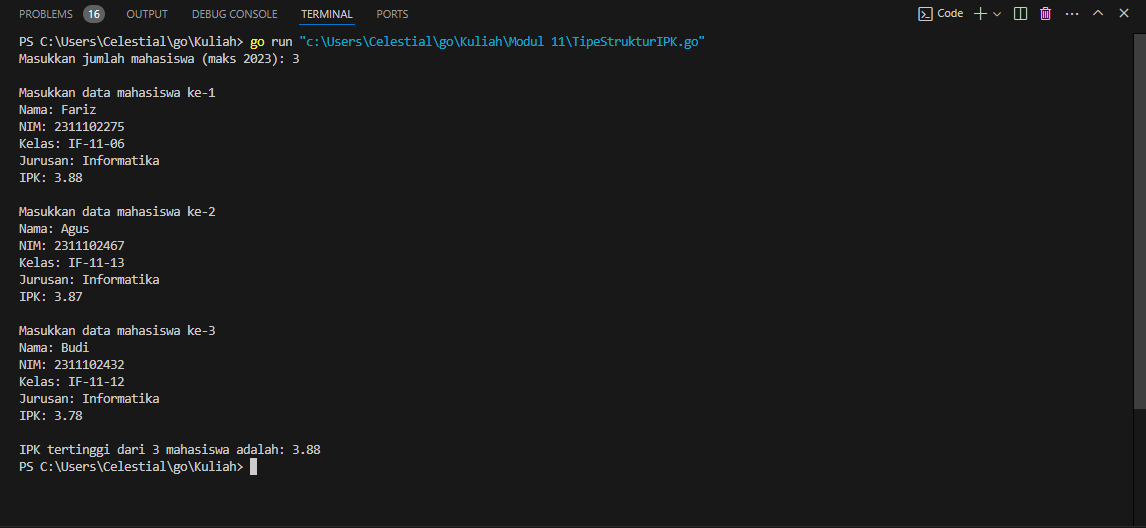
4. Setelah indeks elemen terkecil ditemukan, program menampilkan nilai terkecil tersebut bersama dengan indeksnya.

1. 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.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  // Definisi struct mahasiswa dengan atribut nama, nim, kelas, jurusan, dan ipk  type mahasiswa struct {  nama, nim, kelas, jurusan string  ipk float64  }  // Definisi tipe data array mahasiswa dengan kapasitas maksimal 2023  type arrMhs [2023]mahasiswa  // Fungsi untuk mencari IPK tertinggi dalam array mahasiswa  func ipk(T arrMhs, n int) float64 {  var tertinggi float64 = T[0].ipk  var j int = 1  for j < n {  if tertinggi < T[j].ipk {  tertinggi = T[j].ipk  }  j = j + 1  }  return tertinggi  }  // Fungsi main untuk mengisi data mahasiswa dan mencari IPK tertinggi  func main() {  var n int  var dataMhs arrMhs  // Meminta input jumlah mahasiswa  fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")  fmt.Scan(&n)  // Validasi jumlah mahasiswa yang dimasukkan  if n < 1 || n > 2023 {  fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")  return  }  // Mengisi data mahasiswa  for i := 0; i < n; i++ {  fmt.Printf("\nMasukkan data mahasiswa ke-%d\n", i+1)  fmt.Print("Nama: ")  fmt.Scan(&dataMhs[i].nama)  fmt.Print("NIM: ")  fmt.Scan(&dataMhs[i].nim)  fmt.Print("Kelas: ")  fmt.Scan(&dataMhs[i].kelas)  fmt.Print("Jurusan: ")  fmt.Scan(&dataMhs[i].jurusan)  fmt.Print("IPK: ")  fmt.Scan(&dataMhs[i].ipk)  }  // Mencari dan menampilkan IPK tertinggi  tertinggi := ipk(dataMhs, n)  fmt.Printf("\nIPK tertinggi dari %d mahasiswa adalah: %.2f\n", n, tertinggi)    } |

**Screenshoot Output**



**Deskripsi Program**

Program ini digunakan untuk memasukkan data mahasiswa seperti nama, NIM, kelas, jurusan, dan IPK, lalu mencari dan menampilkan IPK tertinggi dari seluruh mahasiswa yang dimasukkan. Program ini menggunakan array untuk menyimpan data mahasiswa dan mendefinisikan fungsi untuk mencari IPK tertinggi dalam array tersebut.

* Algoritma:

1. Input jumlah mahasiswa ( n ) dari pengguna.

* Jika jumlah mahasiswa yang dimasukkan kurang dari 1 atau lebih dari 2023, program akan menampilkan pesan kesalahan dan berhenti.

2. Input data mahasiswa berupa nama, NIM, kelas, jurusan, dan IPK untuk ( n ) mahasiswa.

3. Fungsi `ipk` digunakan untuk mencari IPK tertinggi di antara mahasiswa:

* Fungsi ini menginisialisasi IPK tertinggi dengan IPK mahasiswa pertama, lalu membandingkannya dengan IPK mahasiswa lainnya.
* Jika ditemukan IPK yang lebih tinggi, maka IPK tertinggi diperbarui.

4. Program kemudian menampilkan IPK tertinggi dari seluruh mahasiswa yang dimasukkan.

* Cara Kerja:

1. Program meminta pengguna untuk memasukkan jumlah mahasiswa ( n ).

2. Program kemudian meminta data mahasiswa (nama, NIM, kelas, jurusan, dan IPK) untuk setiap mahasiswa dan menyimpannya dalam array `dataMhs`.

3. Fungsi `ipk` mencari IPK tertinggi dari mahasiswa yang ada.

4. Setelah mendapatkan IPK tertinggi, program akan menampilkan nilai tersebut.

1. **UNGUIDED**

**Soal Studi Case**

1. Sebuah program digunakan untuk mendata berat anak kelinci yang akan dijual ke pasar. Program ini menggunakan array dengan kapasitas 1000 untuk menampung data berat anak kelinci yang akan dijual.

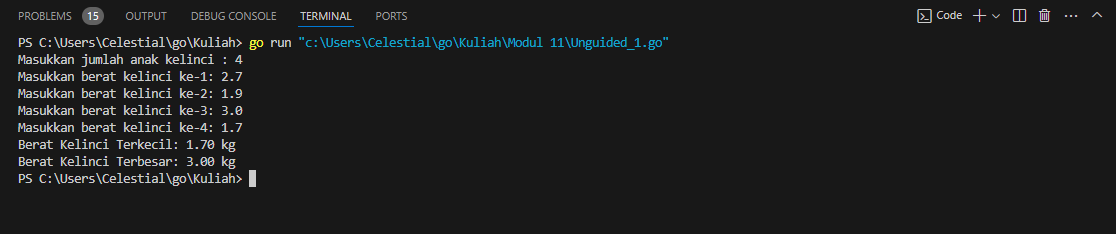
**Masukan** terdiri dari sekumpulan bilangan, yang mana bilangan pertama adalah bilangan bulat N yang menyatakan banyaknya anak kelinci yang akan ditimbang beratnya. Selanjutnya N bilangan riil berikutnya adalah berat dari anak kelinci yang akan dijual.

**Keluaran** terdiri dari dua buah bilangan rill yang menyatakan berat kelinci terkecil dan terbesar

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func main() {  var n int  var minweight, maxweight float64  var weights [1000]float64  fmt.Print("Masukkan jumlah anak kelinci : ")  fmt.Scan(&n)  if n < 1 || n > 1000 {  fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")  return  }  for i := 0; i < n; i++ {  fmt.Printf("Masukkan berat kelinci ke-%d: ", i+1)  fmt.Scan(&weights[i])  if i == 0 {  minweight = weights[i]  maxweight = weights[i]  } else {  if weights[i] < minweight {  minweight = weights[i]  }  if weights[i] > maxweight {  maxweight = weights[i]  }  }  }  fmt.Printf("Berat Kelinci Terkecil: %.2f kg\n", minweight)  fmt.Printf("Berat Kelinci Terbesar: %.2f kg\n", maxweight)  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini menentukan berat badan terkecil dan terbesar dari sejumlah anak kelinci berdasarkan input pengguna. Pengguna memasukkan jumlah anak kelinci dan berat badan masing-masing kelinci. Program kemudian mencetak berat badan terkecil dan terbesar dari data tersebut.

* Algoritma:

1. Input ( n ) jumlah anak kelinci dari pengguna.

2. Validasi nilai ( n ) agar berada dalam rentang 1 hingga 1000. Jika tidak valid, program dihentikan.

3. Input berat badan setiap kelinci sebanyak ( n ) kali:

* Jika berat badan adalah input pertama:

- Tetapkan nilai tersebut sebagai ‘minweight’ dan ‘maxweight’.

* Jika bukan input pertama:

- Bandingkan berat badan dengan ‘minweight’ dan perbarui jika lebih kecil.

- Bandingkan berat badan dengan ‘maxweight’ dan perbarui jika lebih besar.

4. Cetak hasil:

* Berat badan terkecil ( minweight ).
* Berat badan terbesar ( maxweight ).
* Cara Kerja:

1. Program dimulai dengan meminta pengguna memasukkan jumlah anak kelinci ( n ).

2. Jika jumlah anak kelinci ( n ) valid, program meminta pengguna memasukkan berat badan setiap kelinci.

3. Program memproses input berat badan untuk menemukan nilai terkecil dan terbesar:

* Pada input pertama, berat badan kelinci dijadikan acuan untuk ‘minweight’ dan ‘maxweight’.
* Pada input selanjutnya, berat badan dibandingkan dengan nilai terkecil dan terbesar yang tersimpan.

4. Program mencetak berat badan terkecil dan terbesar dalam format dua desimal.

1. Sebuah program digunakan untuk menentukan tarif ikan yang akan dijual ke pasar. Program Ini menggunakan array dengan kapasitas 1000 untuk menampung data berat ikan yang akan dijual.

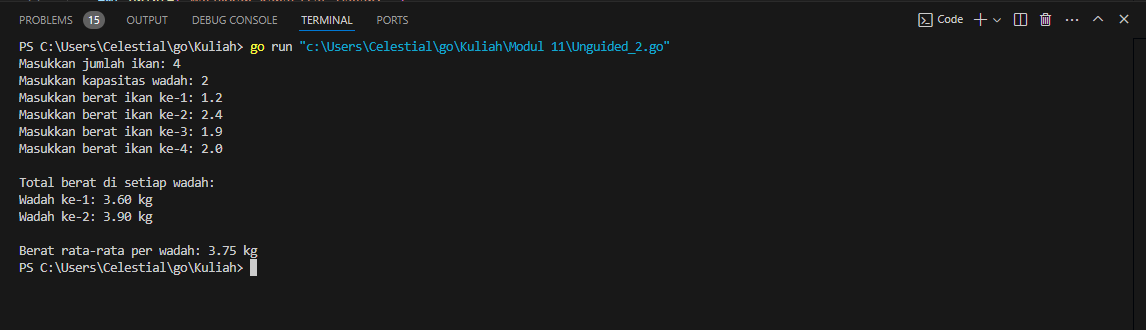
**Masukan** terdiri dari dua baris, yang mana baris pertama terdiri dari dua bilangan bulat x dan y. Bilangan x menyatakan banyaknya ikan yang akan dijual, sedangkan y adalah banyaknya ikan yang akan dimasukan ke dalam wadah. Baris kedua terdiri dari sejumlah x bilangan riil yang menyatakan banyaknya ikan yang akan dijual.

**Keluaran** terdiri dari dua baris. Baris pertama adalah kumpulan bilangan riil yang menyatakan total berat ikan di setiap wadah (jumlah wadah tergantung pada nilai x dan y, urutan ikan yang dimasukan ke dalam wadah sesuai urutan pada masukan baris ke-2). Baris kedua adalah sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func main() {  var x, y int  var weights [1000]float64  var totalWeights []float64  fmt.Printf("Masukkan jumlah ikan: ")  fmt.Scan(&x)  fmt.Printf("Masukkan kapasitas wadah: ")  fmt.Scan(&y)  if x < 1 || x > 1000 || y < 1 {  fmt.Println("Jumlah ikan harus antara 1 dan 1000, kapasitas wadah minimal harus 1.")  return  }  for i := 0; i < x; i++ {  fmt.Printf("Masukkan berat ikan ke-%d: ", i+1)  fmt.Scan(&weights[i])  }  var sum, count, totalWeightsum float64  for i := 0; i < x; i++ {  sum += weights[i]  count++  if int(count) == y || i == x-1 {  totalWeights = append(totalWeights, sum)  totalWeightsum += sum  sum = 0  count = 0  }  }  fmt.Println("\nTotal berat di setiap wadah: ")  for i, weigth := range totalWeights {  fmt.Printf("Wadah ke-%d: %.2f kg\n", i+1, weigth)  }  averangeweight := totalWeightsum / float64(len(totalWeights))  fmt.Printf("\nBerat rata-rata per wadah: %.2f kg\n", averangeweight)  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini digunakan untuk membagi jumlah ikan ke dalam beberapa wadah dengan kapasitas tertentu. Setelah semua ikan ditempatkan, program menghitung total berat ikan di setiap wadah dan rata-rata berat per wadah.

* Algoritma:

1. Input jumlah ikan ( x ) dari pengguna.

2. Input kapasitas wadah ( y ) dari pengguna.

3. Validasi input:

* Pastikan jumlah ikan berada dalam rentang 1 hingga 1000.
* Pastikan kapasitas wadah minimal 1.
* Jika validasi gagal, hentikan program.

4. Input berat ikan satu per satu sebanyak ( x ) kali dan simpan dalam array `weights`.

5. Proses pembagian ikan ke dalam wadah:

* Gunakan variabel `sum` untuk menjumlah berat ikan hingga memenuhi kapasitas ( y ) atau hingga semua ikan selesai diproses.
* Tambahkan nilai `sum` ke dalam slice `totalWeights` untuk mencatat total berat setiap wadah.
* Reset `sum` dan hitung ulang untuk wadah berikutnya.

6. Hitung rata-rata berat per wadah dengan membagi total seluruh berat ikan dengan jumlah wadah.

7. Cetak hasil:

* Total berat ikan di setiap wadah.
* Rata-rata berat ikan per wadah.
* Cara Kerja:

1. Program meminta pengguna memasukkan jumlah ikan (x) dan kapasitas wadah ( y ).

2. Berat setiap ikan dimasukkan satu per satu oleh pengguna.

3. Program memproses berat ikan ke dalam wadah:

* Jika jumlah ikan dalam wadah mencapai kapasitas (y), pindah ke wadah baru.
* Jika ikan habis sebelum kapasitas terpenuhi, simpan total berat ke dalam wadah terakhir.

4. Setelah semua ikan terproses, program mencetak total berat ikan di setiap wadah.

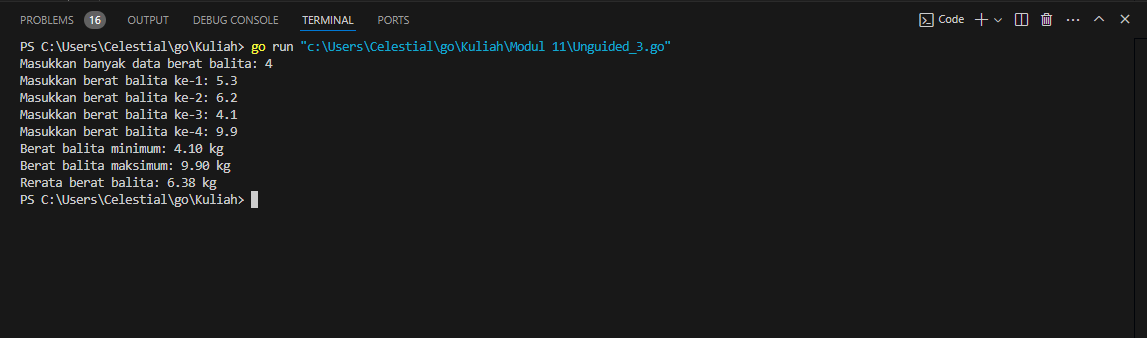
5. Program menghitung dan mencetak rata-rata berat ikan per wadah.

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.

**Sourcecode**

|  |
| --- |
| 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 rerata(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 arrBerat arrBalita  var bMin, bMax float64  fmt.Print("Masukkan banyak data berat balita: ")  fmt.Scan(&n)  if n < 1 || n > 100 {  fmt.Println("Jumlah balita harus antara 1 dan 100.")  return  }  for i := 0; i < n; i++ {  fmt.Printf("Masukkan berat balita ke-%d: ", i+1)  fmt.Scan(&arrBerat[i])  }  hitungMinMax(arrBerat, n, &bMin, &bMax)  rata := rerata(arrBerat, n)  fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)  fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)  fmt.Printf("Rerata berat balita: %.2f kg\n", rata)  } |

**Screenshoot Output**



**Deskripsi Program**

Program ini digunakan untuk menghitung berat balita minimum, maksimum, dan rerata berdasarkan input pengguna. Program ini menggunakan array untuk menyimpan data berat balita dan memanfaatkan fungsi untuk perhitungan.

* Algoritma:

1. Input jumlah data berat balita ( n ) dari pengguna.

- Jika n kurang dari 1 atau lebih dari 100, program menampilkan pesan kesalahan dan berhenti.

2. Input berat setiap balita sebanyak ( n ), yang disimpan dalam array `arrBerat`.

3. Fungsi `hitungMinMax` digunakan untuk mencari berat balita terkecil ( bMin ) dan terbesar ( bMax ).

* Fungsi ini menerima array berat balita dan menghitung nilai minimum dan maksimum.

4. Fungsi `rerata` digunakan untuk menghitung rata-rata berat balita.

* Fungsi ini menjumlahkan seluruh nilai berat dan membaginya dengan jumlah data.

5. Program mencetak hasil perhitungan:

* Berat balita minimum
* Berat balita maksimum
* Rata-rata berat balita
* Cara Kerja:

1. Program meminta pengguna memasukkan jumlah balita ( n ).

2. Program kemudian meminta pengguna memasukkan berat setiap balita satu per satu dan menyimpannya dalam array `arrBerat`.

3. Fungsi `hitungMinMax` menghitung berat minimum dan maksimum balita dalam array.

4. Fungsi `rerata` menghitung rata-rata berat balita.

5. Hasil perhitungan (berat minimum, maksimum, dan rata-rata) ditampilkan kepada pengguna.