

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL XI
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



Disusun Oleh :

Rendi Widya Anggita/2311102278

S1IF-11-06

Dosen Pengampu :

ABEDNEGO DWI SEPTIADI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Secara umum, pencarian nilai ekstrem bertujuan untuk menemukan elemen dengan nilai terbesar (maksimum) atau terkecil (minimum) dari suatu himpunan data. Proses ini melibatkan iterasi melalui elemen-elemen dalam himpunan dan melakukan perbandingan berulang kali untuk menentukan elemen yang memenuhi kriteria maksimum atau minimum. Algoritma ini dapat diterapkan pada berbagai jenis struktur data, seperti array, list, atau bahkan himpunan data yang lebih kompleks. Pencarian nilai ekstrem merupakan bagian dari operasi dasar yang sering digunakan dalam berbagai aplikasi pemrograman, seperti analisis data, pemrosesan sinyal, hingga pengelolaan basis data.

1. Algoritma Dasar Pencarian Nilai Ekstrem

- Algoritma untuk mencari nilai ekstrem umumnya bersifat sederhana dan efisien. Prosesnya dapat dijelaskan sebagai berikut:
- Inisialisasi sebuah variabel untuk menyimpan nilai ekstrem sementara, biasanya dimulai dengan nilai elemen pertama.
- Lakukan iterasi pada setiap elemen dalam himpunan data.
- Bandingkan setiap elemen dengan nilai ekstrem sementara. Jika elemen tersebut lebih besar (atau lebih kecil), perbarui nilai ekstrem sementara.
- Setelah iterasi selesai, variabel yang menyimpan nilai ekstrem sementara akan berisi nilai maksimum atau minimum yang dicari.

II. GUIDED

1. Soal Studi Case

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])
    }
}
```

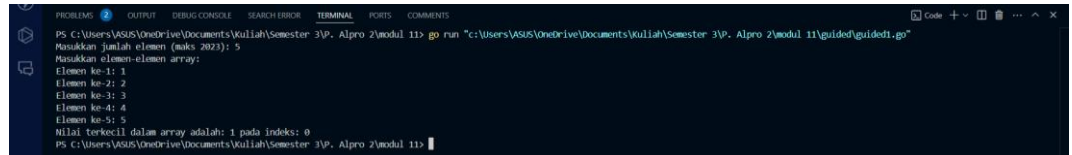
```

    // 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



```

PS C:\Users\VASUS\OneDrive\Documents\Ualliah\Semester 3\P. Alpro 2\modul 11> go run "c:\Users\VASUS\OneDrive\Documents\Ualliah\Semester 3\P. Alpro 2\modul 11\guided\guided.go"
Masukkan jumlah elemen (maks 2023): 5
Masukkan elemen-elemen array:
Elemen ke-1: 1
Elemen ke-2: 2
Elemen ke-3: 3
Elemen ke-4: 4
Elemen ke-5: 5
Nilai terkecil dalam array adalah: 1 pada indeks: 0
PS C:\Users\VASUS\OneDrive\Documents\Ualliah\Semester 3\P. Alpro 2\modul 11>

```

Deskripsi Program

Program tersebut adalah program untuk mencari elemen terkecil dalam sebuah array beserta indeksnya. Program ini menggunakan tipe data array statis dengan Panjang maksimum 2023 elemen, yang telah dideklarasikan sebagai `arrInt`.

- Tipe data `arrInt` didefinisikan sebagai array dengan panjang tetap 2023.
- Fungsi `terkecil` bertugas untuk mencari indeks elemen terkecil dari array yang diberikan. Algoritma ini menggunakan iterasi untuk membandingkan elemen-elemen array, dimulai dari indeks ke-0 sebagai deklarasi awal elemen terkecil. Jika ditemukan elemen lain yang lebih kecil, indeksnya disimpan dalam variabel `idx`. Fungsi ini mengembalikan indeks elemen terkecil setelah selesai memeriksa seluruh elemen.

Awalnya program akan meminta user untuk memasukkan jumlah elemen array dengan batas maksimum 2023, kemudian program akan memvalidasi untuk memastikan apakah nilai `n` berada dalam rentang 1 sampai 2023.

Kemudian program akan meminta user untuk memasukkan nilai dari setiap elemen array, elemen ini akan disimpan dalam variabel `tab` bertipe `arrInt`.

Program kemudian memanggil fungsi “`terkecil`” untuk menentukan indeks elemen terkecil dalam array. Kemudian hasilnya akan ditampilkan ke layar.

2. Soal Studi Case

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

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE SEARCH ERROR TERMINAL PORTS COMMENTS
Name: Rendi
NIM: 2311102278
Kelas: G
Jurusan: Informatika
IPK: 3.9
Masukkan data mahasiswa ke-3
Name: Egi
NIM: 2311102277
Kelas: G
Jurusan: Informatika
IPK: 3.95
Masukkan data mahasiswa ke-4
Name: Hanif
NIM: 2311102266
Kelas: F
Jurusan: Informatika
IPK: 3.85
Masukkan data mahasiswa ke-5
Name: Deffa
NIM: 2311102272
Kelas: A
Jurusan: Informatika
IPK: 4
IPK tertinggi dari 5 mahasiswa adalah: 4.00
PS C:\Users\VASUS\OneDrive\Documents\Kuliah\Semester 3\P. Alpro 2\modul 11>

```

Deskripsi Program

Program ini dibuat untuk mengolah data mahasiswa serta menentukan nilai IPK (Indeks Prestasi Kumulatif) tertinggi di antara sekumpulan data mahasiswa yang dimasukkan oleh pengguna. Program memanfaatkan tipe data struct untuk merepresentasikan data mahasiswa, dan array statis untuk menyimpan data banyak mahasiswa.

Struktur Data dan Tipe Data :

- Struct mahasiswa :
 - Struct mahasiswa digunakan untuk menyimpan atribut terkait mahasiswa, meliputi:
 - Nama : Nama mahasiswa (string).
 - Nim : Nomor Induk Mahasiswa (string).
 - Kelas : Kelas mahasiswa (string).
 - Jurusan: Jurusan mahasiswa (string).
 - Ipk : IPK mahasiswa (float64).
- Array arrMhs

Tipe data arrMhs didefinisikan sebagai array statis dengan kapasitas maksimum 2023 elemen. Array ini digunakan untuk menyimpan data seluruh mahasiswa.

Pertama program akan meminta pengguna untuk memasukkan jumlah mahasiswa (n) dengan batas maks 2023, kemudian program memvalidasi input jumlah mahasiswa apakah berada dalam rentang 1 hingga 2023, jika tidak valid program akan langsung berhenti.

Setelah itu program akan meminta pengguna memasukkan data setiap mahasiswa, seperti nama, nim, kelas, jurusan, dan ipk. Setelah itu program akan memanggil fungsi ipk untuk mencari nilai IPK tertinggi di antara data mahasiswa yang telah dimasukkan.

III. UNGUIDED

1. Soal Studi Case

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 riil yang menyatakan berat kelinci terkecil dan terbesar.

Sourcecode

```
package main

import "fmt"

func cariBeratKelinci(beratKelinci []float64) (float64, float64) {
    // Penanganan kasus array kosong
    if len(beratKelinci) == 0 {
        return 0, 0
    }

    // Inisialisasi nilai awal
    beratTerkecil := beratKelinci[0]
    beratTerbesar := beratKelinci[0]

    // Iterasi untuk mencari nilai ekstrim
    for i := 1; i < len(beratKelinci); i++ {
        if beratKelinci[i] < beratTerkecil {
            beratTerkecil = beratKelinci[i]
        }
        if beratKelinci[i] > beratTerbesar {
            beratTerbesar = beratKelinci[i]
        }
    }

    return beratTerkecil, beratTerbesar
}

func main() {
    // Deklarasi array untuk menyimpan data
    var jumlahData int
    fmt.Print("Masukkan jumlah kelinci: ")
    fmt.Scan(&jumlahData)

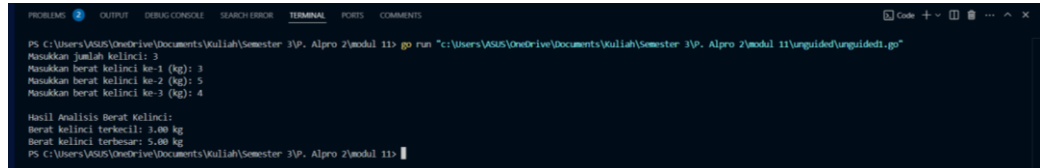
    // Membuat slice dengan kapasitas sesuai input
    beratKelinci := make([]float64, jumlahData)

    // Input data berat kelinci
    for i := 0; i < jumlahData; i++ {
        fmt.Printf("Masukkan berat kelinci ke-%d (kg): ",
i+1)
        fmt.Scan(&beratKelinci[i])
    }

    // Mencari berat terkecil dan terbesar
    min, max := cariBeratKelinci(beratKelinci)
```

```
// Menampilkan hasil
fmt.Printf("\nHasil Analisis Berat Kelinci:")
fmt.Printf("\nBerat kelinci terkecil: %.2f kg", min)
fmt.Printf("\nBerat kelinci terbesar: %.2f kg\n", max)
}
```

Screenshoot Output



```
PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\Pro. Alpro 2\modul 11> go run "c:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\Pro. Alpro 2\modul 11\unguided\unguided.go"
Masukkan jumlah kelinci: 3
Masukkan berat kelinci ke-1 (kg): 3
Masukkan berat kelinci ke-2 (kg): 5
Masukkan berat kelinci ke-3 (kg): 4

Hasil Analisis Berat Kelinci:
Berat kelinci terkecil: 3.00 kg
Berat kelinci terbesar: 5.00 kg
PS C:\Users\ASUS\OneDrive\Documents\Kuliah\Semester 3\Pro. Alpro 2\modul 11>
```

Deskripsi Program

Program ini adalah program untuk mencari nilai ekstrim pada data berat kelinci, dimana program ini menggunakan array untuk menyimpan data dan teknik iterasi sederhana untuk mencari nilai minimum dan maksimum.

Program ini dibuat untuk membantu pengguna menentukan berat terkecil dan terbesar dari sejumlah kelinci berdasarkan data berat yang diinputkan. Program ini menggunakan array untuk menyimpan data dan teknik iterasi sederhana untuk mencari nilai minimum dan maksimum.

Pertama pengguna akan diminta untuk memasukkan jumlah kelinci terlebih dahulu. Kemudian, program meminta berat masing-masing kelinci satu per satu dalam satuan kilogram. Data berat ini disimpan dalam sebuah slice bertipe float64 untuk memastikan akurasi pengolahan angka desimal.

Kemudian program menggunakan fungsi bernama cariBeratKelinci yang menerima slice berisi data berat kelinci. Fungsi ini bertugas untuk. Lalu program mencari berat terbesar. Fungsi ini melakukan iterasi terhadap slice dan membandingkan setiap elemen untuk menemukan nilai minimum dan maksimum. Jika jumlah kelinci yang dimasukkan adalah nol (array kosong), fungsi cariBeratKelinci akan mengembalikan nilai default, yaitu 0 untuk berat terkecil dan terbesar. Ini memastikan program tetap berjalan tanpa error meskipun input tidak valid.

2. Soal Studi Case

- 2) 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 dimasukkan 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 dimasukkan 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() {
    // Step 1: Penjelasan Program
    fmt.Println("Program ini mendistribusikan berat ikan ke dalam wadah.")
    fmt.Println("Masukkan jumlah ikan (x) dan jumlah wadah (y), diikuti berat ikan masing-masing.")

    // Step 2: Input
    var x, y int
    fmt.Print("Masukkan jumlah ikan (x) : ")
    fmt.Print("Masukkan jumlah wadah (y): ")
    fmt.Scan(&x)
    fmt.Scan(&y)

    // Membuat slice untuk menyimpan berat ikan
    fishWeights := make([]float64, x)
    fmt.Printf("Masukkan %d berat ikan (dipisahkan dengan spasi): ", x)
    for i := 0; i < x; i++ {
        fmt.Scan(&fishWeights[i])
    }

    // Step 3: Distribusi ikan ke dalam wadah
    buckets := make([][]float64, y)
    for i, weight := range fishWeights {
        bucketIndex := i % y // Distribusi secara bergilir (round-robin)
        buckets[bucketIndex] = append(buckets[bucketIndex], weight)
    }

    // Step 4: Perhitungan total dan rata-rata berat di setiap wadah
    totalWeights := make([]float64, y)
    averageWeights := make([]float64, y)

    for i := 0; i < y; i++ {
        var totalWeight float64
        for _, weight := range buckets[i] {
            totalWeight += weight
        }
    }
}
```

```

        totalWeights[i] = totalWeight
        if len(buckets[i]) > 0 {
            averageWeights[i] = totalWeight /
float64(len(buckets[i]))
        }
    }

    // Step 5: Output Hasil
    fmt.Println("\nHasil :")
    fmt.Println("Total berat di setiap wadah:")
    for i, total := range totalWeights {
        fmt.Printf("Wadah %d: %.2f\n", i+1, total)
    }

    fmt.Println("\nRata-rata berat ikan di setiap wadah:")
    for i, avg := range averageWeights {
        fmt.Printf("Wadah %d: %.2f\n", i+1, avg)
    }
}

```

Screenshoot Output

```

PS C:\Users\VASUS\OneDrive\Documents\Kuliah\Semester 3\p. Alpro 2\modul 11> go run "c:\Users\VASUS\OneDrive\Documents\Kuliah\Semester 3\p. Alpro 2\modul 11\unguided\unguided2.go"
Program ini mendistribusikan berat ikan ke dalam wadah.
Masukkan jumlah ikan (x) dan jumlah wadah (y), diikuti berat ikan masing-masing.
Masukkan jumlah ikan (x) : 5
Masukkan jumlah wadah (y): 3
Masukkan 5 berat ikan (dipisahkan dengan spasi): 2 3 4 5 6

Hasil :
Total berat di setiap wadah:
Wadah 1: 7.00
Wadah 2: 9.00
Wadah 3: 4.00

Rata-rata berat ikan di setiap wadah:
Wadah 1: 2.50
Wadah 2: 4.50
Wadah 3: 4.00
PS C:\Users\VASUS\OneDrive\Documents\Kuliah\Semester 3\p. Alpro 2\modul 11>

```

Deskripsi Program

Program ini dirancang untuk mendistribusikan berat ikan ke sejumlah wadah menggunakan metode round-robin. Program ini memberikan solusi untuk pembagian beban secara seimbang dengan menampilkan total berat dan rata-rata berat ikan di setiap wadah.

Pertama pengguna diminta untuk memasukkan jumlah ikan (x) dan jumlah wadah (y). Kemudian program akan memberikan daftar berat ikan sesuai dengan jumlah ikan yang dimasukkan.

Program menggunakan metode round-robin untuk mendistribusikan ikan ke wadah. Ini berarti ikan pertama akan masuk ke wadah pertama, ikan kedua ke wadah kedua, dan seterusnya. Ketika semua wadah sudah terisi setidaknya sekali, distribusi akan kembali ke wadah pertama, mengulang siklus hingga semua ikan terdistribusi.

3. Soal Studi Case

- 3) 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.

Buatlah program dengan spesifikasi subprogram sebagai berikut:

```
type arrBalita [100]float64
func hitungMinMax(arrBerat arrBalita; bMin, bMax *float64) {
    /* F.S. Terdefinisi array dinamis arrBerat
```

alaman 73 | Modul Praktikum Algoritma dan Pemrograman 2

```
Proses: Menghitung berat minimum dan maksimum dalam array
F.S. Menampilkan berat minimum dan maksimum balita */
...
}

function rerata(arrBerat arrBalita) real {
    /* menghitung dan mengembalikan rerata berat balita dalam array */
    ...
}
```

Perhatikan sesi interaksi pada contoh berikut ini (teks berwarna bawah adalah input/read)

```
Masukan banyak data berat balita : 4
Masukan berat balita ke-1: 5.3
Masukan berat balita ke-2: 6.2
Masukan berat balita ke-3: 4.1
Masukan berat balita ke-4: 9.9
Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
```

Sourcecode

```
package main

import (
    "fmt"
)

// Mendefinisikan tipe data arrBalita sebagai array float64
dengan ukuran maksimum 100
type arrBalita [100]float64

// Fungsi untuk menghitung berat minimum dan maksimum
func hitungMinMax(arrBerat []float64, bMin, bMax *float64) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]

    for _, berat := range arrBerat {
        if berat < *bMin {
            *bMin = berat
        }
        if berat > *bMax {
            *bMax = berat
        }
    }
}

// Fungsi untuk menghitung rata-rata berat
func hitungRerata(arrBerat []float64) float64 {
    var total float64
    for _, berat := range arrBerat {
        total += berat
    }
}
```

```

    return total / float64(len(arrBerat))
}

func main() {
    var n int
    var berat arrBalita

    // Input: jumlah data berat balita
    fmt.Print("Masukan banyak data berat balita: ")
    fmt.Scanln(&n)

    // Input: berat masing-masing balita
    for i := 0; i < n; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ", i+1)
        fmt.Scanln(&berat[i])
    }

    arrBerat := berat[:n]

    // Menghitung berat minimum, maksimum, dan rata-rata
    var bMin, bMax float64
    hitungMinMax(arrBerat, &bMin, &bMax)
    rerata := hitungRerata(arrBerat)

    // Output hasil
    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", rerata)
}

```

Screenshoot Output

```

PS C:\Users\VASUS\OneDrive\Documents\Kuliah\Semester 3\Pro. Alpro 2\modul 11> go run "c:\Users\VASUS\OneDrive\Documents\Kuliah\Semester 3\Pro. Alpro 2\modul 11\unguided\unguided3.go"
Masukan banyak data berat balita: 5
Masukan berat balita ke-1: 2
Masukan berat balita ke-2: 3
Masukan berat balita ke-3: 4
Masukan berat balita ke-4: 5
Masukan berat balita ke-5: 6
Berat balita minimum: 2.00 kg
Berat balita maksimum: 6.00 kg
Rata-rata berat balita: 4.00 kg
PS C:\Users\VASUS\OneDrive\Documents\Kuliah\Semester 3\Pro. Alpro 2\modul 11>

```

Deskripsi Program

Program ini dibuat untuk menganalisis data berat balita. Program ini menghitung berat minimum, berat maksimum, dan rata-rata dari sejumlah data berat yang dimasukkan oleh pengguna.

Pertama pengguna diminta untuk memasukkan jumlah data berat balita yang akan dianalisis. Kemudian berat masing-masing balita dimasukkan satu per satu, dan program menyimpannya dalam array statis dengan kapasitas maksimum 100 elemen.

Setelah penghitungan selesai, program menampilkan hasil berat balita minimum, berat balita maksimum, rata-rata berat balita. Semua hasil ditampilkan dengan dua angka desimal untuk memberikan informasi yang presisi.

Program menggunakan fungsi `hitungMinMax`, yang menerima array berat balita serta pointer untuk menyimpan nilai berat minimum dan maksimum. Fungsi ini membandingkan setiap elemen dalam array untuk menentukan nilai minimum dan maksimum secara efisien.

Fungsi `hitungRerata` digunakan untuk menghitung rata-rata berat balita. Fungsi ini menjumlahkan semua elemen dalam array dan membaginya dengan jumlah data yang tersedia.