

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

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



**Disusun Oleh:**  
Bayu Kuncoro Adi / 2311102031  
S1 IF 11 05

**Dosen Pengampu:**  
Arif Amrulloh, S.Kom., M.Kom.

**PROGRAM STUDI S1 TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
TELKOM UNIVERSITY PURWOKERTO  
2024**

## I. DASAR TEORI

### A. 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 lainnya. 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.
3. Apabila semua data telah dicek, maka nilai ekstrim yang dimiliki adalah valid.

Berikut ini adalah notasi dalam pseudocode dan bahasa Go. misalnvu untuk pencarian nilai terbesar atau maksimum:

	Notasi Algoritma	Notasi dalam bahasa Go
1	<code>max ← 1</code>	<code>max = 0</code>
2	<code>i ← 2</code>	<code>i = 1</code>
3	<code>while i &lt;= n do</code>	<code>for i &lt; n {</code>
4	<code>    if a[i] &gt; a[max] then</code>	<code>    if a[i] &gt; a[max] {</code>
5	<code>        max ← i</code>	<code>        max = i</code>
6	<code>    endif</code>	<code>    }</code>
7	<code>    i ← i + 1</code>	<code>    i = i + 1</code>
8	<code>endwhile</code>	<code>}</code>

## B. Pencarian Nilai Ekstrim pada Array bertipe Data Dasar

Misalnya terdefinisi sebuah array of integer dengan kapasitas 2023, dan array terisi sejumlah N bilangan bulat, kemudian pencarian nilai terkecil dilakukan pada array tersebut. Perhatikan potongan program dalam bahasa Go berikut ini!

```
5  type arrInt [2023]int
..  ...
15
16 func terkecil_1(tabInt arrInt, n int) int {
17  /* mengembalikan nilai terkecil yang terdapat di dalam tabInt yang berisi n
18  bilangan bulat */
19      var min int = tabInt[0]          // min berisi data pertama
20      var j int = 1                    // pencarian dimulai dari data berikutnya
21      for j < n {
22          if min > tabInt[j] {          // pengecekan apakah nilai minimum valid
23              min = tabInt[j]          // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return min                       // returnkan nilai minimumnya
28 }
```

Potongan program di atas sedikit berbeda dengan sebelumnya karena penggunaan indeks array pada bahasa Go di mulai dari nol atau 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!

```
..  ...
5  type arrInt [2023]int
..  ...
15
16 func terkecil_2(tabInt arrInt, n int) int {
17  /* mengembalikan indeks nilai terkecil yang terdapat di dalam tabInt yang berisi
18  n bilangan bulat */
19      var idx int = 0                  // idx berisi indeks data pertama
20      var j int = 1                    // pencarian dimulai dari data berikutnya
21      for j < n {
22          if tabInt[idx] > tabInt[j] { // pengecekan apakah nilai minimum valid
23              idx = j                  // update nilai minimum dengan yang valid
24          }
25          j = j + 1
26      }
27      return idx                       // returnkan indeks nilai minimumnya
28 }
```

### C. 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.

```
..   ...
5   type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
..   }
..   type arrMhs [2023]mahasiswa
..   ...
15
16 func IPK_1(T arrMhs, n int) float64 {
17   /* mengembalikan ipk terkecil yang dimiliki mahasiswa pada array T yang berisi
18   n mahasiswa */
19   var tertinggi float64 = T[0].ipk
20   var j int = 1
21   for j < n {
22     if tertinggi < T[j].ipk {
23       tertinggi = T[j].ipk
24     }
25     j = j + 1
26   }
27   return tertinggi
28 }
```

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!

```

.. ...
5  type mahasiswa struct {
..     nama, nim, kelas, jurusan string
..     ipk float64
.. }
.. type arrMhs [2023]mahasiswa
.. ...
15
16 func IPK_2(T arrMhs, n int) int {
17     /* mengembalikan indeks mahasiswa yang memiliki ipk tertinggi pada array T yang
18     berisi n mahasiswa */
19     var idx int = 0
20     var j int = 1
21     for j < n {
22         if T[idx].ipk < T[j].ipk {
23             idx = j
24         }
25         j = j + 1
26     }
27     return idx
28 }

```

Sehingga melalui algoritma di atas, identitas mahasiswa dapat diperoleh, misalnya T[idx].nama, T[idx].nim, T[idx].kelas, hingga T[idx].jurusan.

## A. UNGUIDED

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 ril berikutnya adalah berat dari anak kelinci yang akan dijual.

Keluaran terdiri dari dua buah bilangan ri yang menyatakan berat kelinci terkecil dan terbesar.

```
package main

import (
    "fmt"
)

func main() {
    // Deklarasi variabel
    var n_2311102031 int
    var weights_2311102031 []float64

    // Input jumlah anak kelinci
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&n_2311102031)

    // Validasi jumlah anak kelinci
    if n_2311102031 <= 0 || n_2311102031 > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")
        return
    }

    // Input berat anak kelinci
    weights_2311102031 = make([]float64, n_2311102031)
    fmt.Println("Masukkan berat masing-masing anak kelinci:")
    for i_2311102031 := 0; i_2311102031 < n_2311102031; i_2311102031++
    {
        fmt.Scan(&weights_2311102031[i_2311102031])
    }

    // Inisialisasi min dan max
    minWeight_2311102031 := weights_2311102031[0]
    maxWeight_2311102031 := weights_2311102031[0]

    // Cari berat terkecil dan terbesar
    for _, weight_2311102031 := range weights_2311102031 {
        if weight_2311102031 < minWeight_2311102031 {
            minWeight_2311102031 = weight_2311102031
        }
        if weight_2311102031 > maxWeight_2311102031 {
            maxWeight_2311102031 = weight_2311102031
        }
    }
```

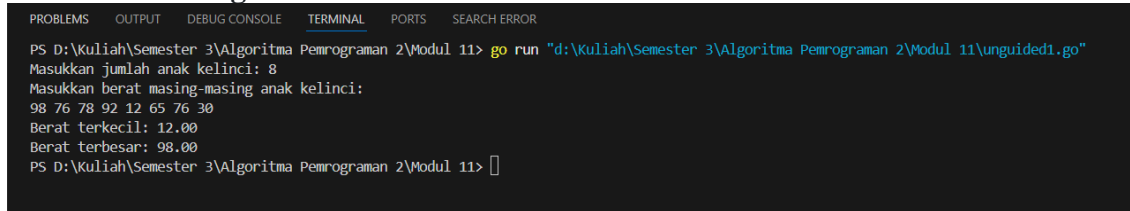
```

    }

    // Output hasil
    fmt.Printf("Berat terkecil: %.2f\n", minWeight_2311102031)
    fmt.Printf("Berat terbesar: %.2f\n", maxWeight_2311102031)
}

```

## Screenshoot Program



```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  SEARCH ERROR
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11\unguided1.go"
Masukkan jumlah anak kelinci: 8
Masukkan berat masing-masing anak kelinci:
98 76 78 92 12 65 76 30
Berat terkecil: 12.00
Berat terbesar: 98.00
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11> 

```

## Deskripsi dan Cara Kerja Program

Program di atas adalah program Go yang dibuat untuk mencari berat terkecil dan terbesar dari sekumpulan anak kelinci. Program dimulai dengan mengimpor package "fmt" untuk input/output dan mendeklarasikan variabel `n_2311102031` bertipe integer untuk menyimpan jumlah anak kelinci, serta slice `weights_2311102031` bertipe float64 untuk menyimpan berat masing-masing kelinci. Program meminta input jumlah anak kelinci dari pengguna dan melakukan validasi bahwa jumlah tersebut harus berada antara 1 dan 1000. Jika validasi gagal, program akan menampilkan pesan error dan berhenti.

Setelah validasi berhasil, program membuat slice dengan ukuran sesuai jumlah kelinci yang diinput dan meminta pengguna memasukkan berat untuk setiap anak kelinci. Program kemudian menginisialisasi variabel `minWeight_2311102031` dan `maxWeight_2311102031` dengan nilai berat kelinci pertama, lalu melakukan iterasi melalui seluruh berat kelinci menggunakan range loop untuk mencari nilai terkecil dan terbesar. Setiap berat dibandingkan dengan nilai minimum dan maksimum yang ada, dan nilai-nilai tersebut diperbarui jika ditemukan berat yang lebih kecil atau lebih besar. Akhirnya, program menampilkan hasil berat terkecil dan terbesar dengan format dua angka desimal menggunakan `fmt.Printf`.

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 dimasukan ke dalam wadah. Baris kedua terdiri dari sejumlah  $x$  bilangan riil yang menyatakan banyaknya ikan yang akan dijual. InFormatict 1a6

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 ialah: sebuah bilangan riil yang menyatakan berat rata-rata ikan di setiap wadah.

```
package main

import (
    "fmt"
)

func main() {
    // Deklarasi variabel
    var x_2311102031, y_2311102031 int
    var weights_2311102031 []float64

    // Input jumlah ikan dan kapasitas wadah
    fmt.Print("Masukkan jumlah ikan (x) dan kapasitas wadah (y): ")
    fmt.Scan(&x_2311102031, &y_2311102031)

    // Validasi jumlah ikan dan kapasitas wadah
    if x_2311102031 <= 0 || y_2311102031 <= 0 || x_2311102031 > 1000
{
        fmt.Println("Jumlah ikan (x) harus antara 1 dan 1000, dan
kapasitas wadah (y) harus lebih dari 0.")
        return
    }

    // Input berat ikan
    weights_2311102031 = make([]float64, x_2311102031)
    fmt.Println("Masukkan berat masing-masing ikan:")
    for i := 0; i < x_2311102031; i++ {
        fmt.Scan(&weights_2311102031[i])
    }

    // Hitung total berat per wadah
    var totalWeights_2311102031 []float64
    currentWeight_2311102031 := 0.0

    for i, weight := range weights_2311102031 {
        currentWeight_2311102031 += weight
    }
}
```



```

        // Jika wadah penuh atau ikan terakhir
        if (i+1)%y_2311102031 == 0 || i == x_2311102031-1 {
            totalWeights_2311102031 =
append(totalWeights_2311102031, currentWeight_2311102031)
            currentWeight_2311102031 = 0.0
        }
    }

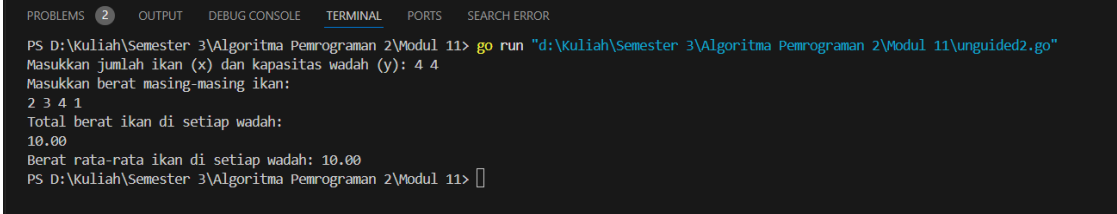
    // Hitung rata-rata berat per wadah
    totalSum_2311102031 := 0.0
    for _, total := range totalWeights_2311102031 {
        totalSum_2311102031 += total
    }
    averageWeight_2311102031 := totalSum_2311102031 /
float64(len(totalWeights_2311102031))

    // Output hasil
    fmt.Println("Total berat ikan di setiap wadah:")
    for _, total := range totalWeights_2311102031 {
        fmt.Printf("%.2f ", total)
    }
    fmt.Println()

    fmt.Printf("Berat rata-rata ikan di setiap wadah: %.2f\n",
averageWeight_2311102031)
}

```

## Screenshoot Program



```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11\unguided2.go"
Masukkan jumlah ikan (x) dan kapasitas wadah (y): 4 4
Masukkan berat masing-masing ikan:
2 3 4 1
Total berat ikan di setiap wadah:
10.00
Berat rata-rata ikan di setiap wadah: 10.00
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11> 

```

### **Deskripsi dan Cara Kerja Program**

Program di atas adalah sebuah implementasi untuk menentukan total berat ikan di setiap wadah dan menghitung rata-rata berat ikan per wadah. Pengguna diminta memasukkan jumlah ikan yang akan dijual (`x_2311102031`) dan kapasitas maksimal ikan per wadah (`y_2311102031`), diikuti dengan berat masing-masing ikan. Program memproses berat ikan sesuai urutan input untuk dimasukkan ke dalam wadah. Jika jumlah ikan dalam wadah mencapai kapasitas maksimal atau ikan terakhir telah dimasukkan, berat total untuk wadah tersebut dihitung dan disimpan. Hasilnya berupa daftar total berat setiap wadah dan rata-rata berat ikan per wadah.

Cara kerja program dimulai dengan membaca input dari pengguna, melakukan validasi pada jumlah ikan dan kapasitas wadah, lalu memproses berat ikan untuk dimasukkan ke dalam wadah. Berat ikan ditambahkan secara bertahap ke variabel `currentWeight_2311102031` hingga wadah penuh atau semua ikan telah diproses. Setelah itu, berat total setiap wadah disimpan dalam array `totalWeights_2311102031`. Program kemudian menghitung rata-rata berat per wadah dengan membagi jumlah total semua berat wadah dengan jumlah wadah yang digunakan. Hasil akhir berupa total berat setiap wadah dan rata-rata berat ditampilkan dalam format dua desimal.

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) {
  /* I.S. Terdefinisi array dinamis arrBerat
```

```
    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 bergaris 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
```

```
package main

import (
    "fmt"
    "strings"
)

// Definisi tipe data untuk array berat balita
type arrBalita [100]float64

// Fungsi untuk menghitung berat minimum dan maksimum
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]
    }
}

// Fungsi untuk menghitung rata-rata berat balita
func rataRata(arrBerat arrBalita, n int) float64 {
    total := 0.0
    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

    // Tampilan header
    fmt.Println(strings.Repeat("=", 40))
    fmt.Println("  Program Pengolahan Data Berat Balita")
    fmt.Println(strings.Repeat("=", 40))

    // Meminta pengguna untuk memasukkan jumlah data balita
    fmt.Print("\nMasukan banyak data berat balita: ")
    fmt.Scanln(&n)

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

    // Memanggil fungsi untuk menghitung minimum dan maksimum
    hitungMinMax(berat, n, &bMin, &bMax)

    // Menghitung rata-rata berat balita
    rata := rataRata(berat, n)

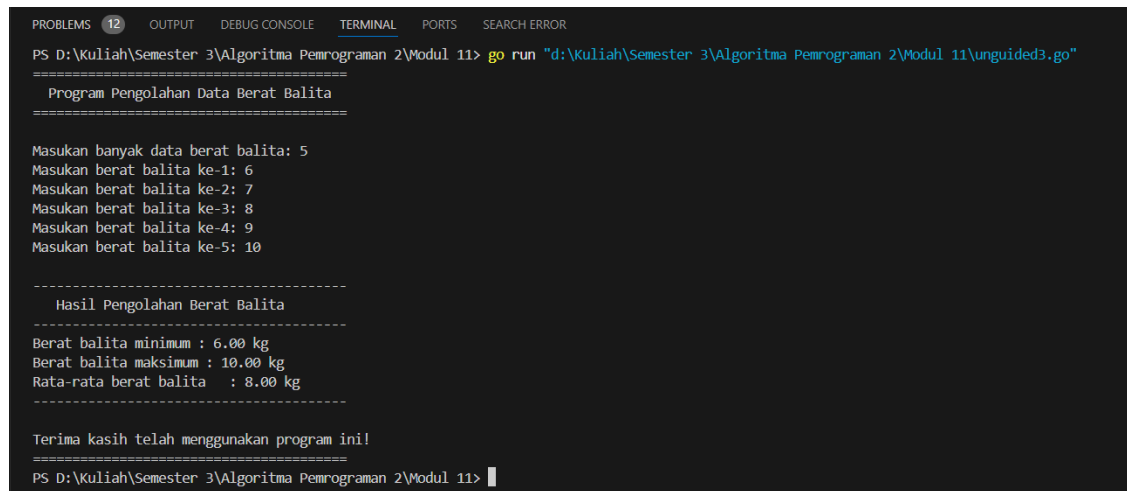
    // Menampilkan hasil
    fmt.Println("\n" + strings.Repeat("-", 40))
    fmt.Println("  Hasil Pengolahan Berat Balita")
    fmt.Println(strings.Repeat("-", 40))
    fmt.Printf("Berat balita minimum : %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum : %.2f kg\n", bMax)
    fmt.Printf("Rata-rata berat balita   : %.2f kg\n", rata)
    fmt.Println(strings.Repeat("-", 40))

    // Menutup program dengan ucapan terima kasih
    fmt.Println("\nTerima kasih telah menggunakan program ini!")
    fmt.Println(strings.Repeat("=", 40))
}

```

```
}
```

## Screenshoot Program



```
PROBLEMS (12) OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11> go run "d:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11\unguided3.go"

Program Pengolahan Data Berat Balita
=====

Masukan banyak data berat balita: 5
Masukan berat balita ke-1: 6
Masukan berat balita ke-2: 7
Masukan berat balita ke-3: 8
Masukan berat balita ke-4: 9
Masukan berat balita ke-5: 10

-----

Hasil Pengolahan Berat Balita
-----

Berat balita minimum : 6.00 kg
Berat balita maksimum : 10.00 kg
Rata-rata berat balita : 8.00 kg
-----

Terima kasih telah menggunakan program ini!
=====
PS D:\Kuliah\Semester 3\Algoritma Pemrograman 2\Modul 11> |
```

## Deskripsi dan Cara Kerja Program

Program ini bertujuan untuk mengolah data berat balita dengan menghitung berat minimum, maksimum, dan rata-rata dari data yang dimasukkan. Pengguna diminta untuk menentukan jumlah balita terlebih dahulu (n) dan memasukkan berat masing-masing balita. Data berat balita disimpan dalam array `arrBalita` dengan kapasitas maksimum 100 elemen. Program menggunakan fungsi `hitungMinMax` untuk mencari nilai berat minimum dan maksimum, serta fungsi `rataRata` untuk menghitung rata-rata berat dari semua data yang dimasukkan. Hasil perhitungan ini kemudian ditampilkan dengan format yang rapi dan menarik.

Cara kerja program dimulai dengan membaca input dari pengguna dan menyimpan data ke dalam array. Fungsi `hitungMinMax` membandingkan setiap elemen array untuk memperbarui nilai berat minimum (`bMin`) dan maksimum (`bMax`). Sementara itu, fungsi `rataRata` menjumlahkan seluruh berat dalam array dan membaginya dengan jumlah data balita (n). Program kemudian mencetak hasil berupa berat minimum, maksimum, dan rata-rata dengan format dua angka desimal, diikuti ucapan terima kasih. Program ini dirancang untuk mempermudah pengguna dalam mengolah data berat balita dengan tampilan yang informatif.

## KESIMPULAN

Program-program yang dibuat dalam laporan ini bertujuan untuk mengolah data dalam berbagai skenario, seperti pencarian nilai ekstrem, pengelompokan data, dan perhitungan rata-rata. Dalam kasus pertama, program dirancang untuk mencari berat terkecil dan terbesar dari sekumpulan data berat anak kelinci. Dengan pendekatan iteratif, program dapat secara efisien menentukan nilai maksimum dan minimum dengan membandingkan setiap elemen dalam array. Pendekatan ini memperlihatkan pentingnya algoritma sederhana untuk memproses data dengan cepat dan akurat.

Pada kasus kedua, program dikembangkan untuk menghitung total berat ikan per wadah dan rata-rata berat ikan di setiap wadah. Program ini mengimplementasikan logika pengelompokan data menggunakan konsep array dan iterasi. Setiap berat ikan ditambahkan ke wadah secara berurutan hingga mencapai kapasitas maksimum, memastikan bahwa data terorganisir dengan baik. Metode ini menunjukkan bagaimana array dapat digunakan untuk mengelompokkan data sesuai dengan kriteria tertentu, sambil memberikan informasi yang berguna, seperti rata-rata berat per kelompok.

Kasus ketiga menyoroti penggunaan program untuk pencatatan data berat balita di posyandu. Program ini memanfaatkan subprogram untuk menghitung nilai ekstrem dan rata-rata, mencerminkan pentingnya modularitas dalam pemrograman. Dengan memisahkan fungsi-fungsi tertentu seperti pencarian nilai ekstrem dan perhitungan rata-rata, program menjadi lebih mudah dibaca, dipelihara, dan diadaptasi untuk skenario serupa. Secara keseluruhan, ketiga program ini menunjukkan efisiensi penggunaan algoritma dasar dalam pemrosesan data, memberikan solusi sederhana untuk berbagai kebutuhan pengolahan data di kehidupan sehari-hari.

## **DAFTAR PUSTAKA**

Modul 11 Praktikum Algoritma dan Pemrograman 2 Modul Nilai Ekstrim