

**LAPORAN PRAKTIKUM  
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 11**

**PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA**



**Disusun Oleh :**

**Rory Refaldo Sinulingga 2311102323**

**Kelas IF-11-06**

**Dosen Pengampu :**

-----

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

## **I. DASAR TEORI**

### Dasar Teori

Pencarian nilai ekstrem, yaitu nilai maksimum dan minimum, pada suatu himpunan data merupakan salah satu konsep dasar dalam algoritma dan pemrograman (alpro). Hal ini sering menjadi bagian awal dalam pembelajaran pemrograman karena melibatkan penggunaan logika iterasi dan seleksi, yang merupakan inti dari banyak algoritma. Pencarian nilai ekstrem digunakan untuk menentukan batas atas dan bawah dalam data, yang berguna dalam analisis statistik, optimasi, dan pengambilan keputusan.

Secara sederhana, pencarian nilai ekstrem dilakukan dengan membandingkan elemen-elemen dalam himpunan data satu per satu. Dalam sebuah algoritma dasar, nilai awal (seed) untuk maksimum dan minimum biasanya diinisialisasi dengan elemen pertama dari data. Kemudian, melalui proses iterasi, setiap elemen dibandingkan dengan nilai maksimum dan minimum sementara, dan pembaruan dilakukan jika elemen tersebut lebih besar atau lebih kecil dari nilai ekstrem sementara. Algoritma ini memiliki kompleksitas waktu  $O(n)$ , di mana  $n$  adalah jumlah elemen dalam himpunan data.

## II. GUIDED

### 1. Sourcecode

```
package main

import "fmt"

// Mendeklarasikan tipe data array arrInt_aldo_323 dengan panjang 2023
type arrInt_aldo_323 [2023]int

// Fungsi untuk mencari elemen terkecil dalam array
func terkecil_aldo_323(tabInt_aldo_323 arrInt_aldo_323, n_aldo_323 int) int {
    var min_aldo_323 int = tabInt_aldo_323[0]
    for j_aldo_323 := 1; j_aldo_323 < n_aldo_323; j_aldo_323++ {
        if min_aldo_323 > tabInt_aldo_323[j_aldo_323] {
            min_aldo_323 = tabInt_aldo_323[j_aldo_323]
        }
    }
    return min_aldo_323
}

// Fungsi main
func main() {
    var n_aldo_323 int
    var tab_aldo_323 arrInt_aldo_323

    // Meminta input jumlah elemen array
    fmt.Println("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n_aldo_323)

    // Validasi input jumlah elemen
    if n_aldo_323 < 1 || n_aldo_323 > 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_aldo_323; i++ {
        fmt.Printf("Elemen ke-%d: ", i+1)
        fmt.Scan(&tab_aldo_323[i])
    }

    // Memanggil fungsi terkecil untuk menemukan nilai terkecil
    minVal_aldo_323 := terkecil_aldo_323(tab_aldo_323, n_aldo_323)

    // Menampilkan nilai terkecil
    fmt.Println("Nilai terkecil dalam array adalah:", minVal_aldo_323)
}
```

## Screenshoot Output

```
PS C:\Users\aldop\OneDrive\Documents\Folder Baru> go run "c:\Users\aldop\OneDrive\Documents\Folder Baru\tempCodeRunnerFile.go"
Masukkan jumlah elemen (maks 2023): 2
Masukkan elemen-elemen array:
Elemen ke-1: 3
Elemen ke-2: 1
Nilai terkecil dalam array adalah: 1
PS C:\Users\aldop\OneDrive\Documents\Folder Baru> |
```

## Deskripsi Program

Program ini bertujuan untuk menemukan elemen terkecil dalam sebuah array. Program akan meminta pengguna untuk memasukkan jumlah elemen array (maksimal 2023), kemudian memasukkan nilai-nilai elemen tersebut satu per satu. Setelah itu, program akan memproses data menggunakan fungsi `terkecil_aldo_323` untuk menentukan nilai terkecil dari array yang diinputkan. Hasilnya ditampilkan sebagai output. Program juga memastikan bahwa jumlah elemen yang dimasukkan valid, yaitu antara 1 dan 2023.

## 2. Sourcecode

```
package main

import "fmt"

// Mendeklarasikan tipe data array arrInt_aldo_323 dengan panjang 2023
type arrInt_aldo_323 [2023]int

// Fungsi untuk mencari indeks elemen terkecil dalam array
func terkecil_aldo_323(tabInt_aldo_323 arrInt_aldo_323, n_aldo_323 int) int {
    var idx_aldo_323 int = 0 // idx_aldo_323 menyimpan indeks elemen terkecil
    for j_aldo_323 := 1; j_aldo_323 < n_aldo_323; j_aldo_323++ {
        if tabInt_aldo_323[idx_aldo_323] > tabInt_aldo_323[j_aldo_323] {
            idx_aldo_323 = j_aldo_323 // Simpan indeks j_aldo_323 jika elemen di indeks j_aldo_323 lebih kecil
        }
    }
    return idx_aldo_323
}

// Fungsi main untuk menguji fungsi terkecil_aldo_323
func main() {
    var n_aldo_323 int
    var tab_aldo_323 arrInt_aldo_323

    // Meminta input jumlah elemen array
    fmt.Print("Masukkan jumlah elemen (maks 2023): ")
    fmt.Scan(&n_aldo_323)

    // Validasi input jumlah elemen
    if n_aldo_323 < 1 || n_aldo_323 > 2023 {
        fmt.Println("Jumlah elemen harus antara 1 dan 2023.")
        return
    }

    // Memasukkan elemen-elemen array
    fmt.Println("Masukkan elemen-elemen array:")
    for i_aldo_323 := 0; i_aldo_323 < n_aldo_323; i_aldo_323++ {
        fmt.Printf("Elemen ke-%d: ", i_aldo_323+1)
        fmt.Scan(&tab_aldo_323[i_aldo_323])
    }

    // Memanggil fungsi terkecil_aldo_323 untuk menemukan indeks elemen terkecil
    idxMin_aldo_323 := terkecil_aldo_323(tab_aldo_323, n_aldo_323)
```

```
// Menampilkan nilai dan indeks terkecil  
fmt.Println("Nilai terkecil dalam array adalah:", tab_aldo_323[idxMin_aldo_323], "pada indeks:", idxMin_aldo_323)  
}
```

## Screenshoot output

```
PS C:\Users\aldop\OneDrive\Documents\Folder Baru> go run "c:\Users\aldop\OneDrive\Documents\Folder Baru\tempCodeRunnerFile.go"
Masukkan jumlah elemen (maks 2023): 3
Masukkan elemen-elemen array:
Elemen ke-1: 1
Elemen ke-2: 4
Elemen ke-3: 2
Nilai terkecil dalam array adalah: 1 pada indeks: 0
PS C:\Users\aldop\OneDrive\Documents\Folder Baru>
```

## Deskripsi

Program ini berfungsi untuk menemukan indeks elemen terkecil dalam sebuah array. Pengguna diminta untuk memasukkan jumlah elemen array (maksimal 2023), kemudian memasukkan nilai elemen satu per satu. Program memproses data dengan fungsi `terkecil_aldo_323`, yang mencari indeks dari elemen terkecil dalam array. Setelah proses selesai, program akan menampilkan nilai terkecil beserta indeksnya. Validasi dilakukan untuk memastikan jumlah elemen yang dimasukkan sesuai dengan batas yang ditentukan.

### 3. Sourcecode

```
package main

import "fmt"

// Definisi struct mahasiswa_aldo_323 dengan atribut nama, nim, kelas,
// jurusan, dan ipk
type mahasiswa_aldo_323 struct {
    nama, nim, kelas, jurusan string
    ipk          float64
}

// Definisi tipe data array mahasiswa_aldo_323 dengan kapasitas maksimal
// 2023
type arrMhs_aldo_323 [2023]mahasiswa_aldo_323

// Fungsi untuk mencari indeks IPK tertinggi dalam array
// mahasiswa_aldo_323
func indeksIPKTertinggi_aldo_323(T_aldo_323 arrMhs_aldo_323,
n_aldo_323 int) int {
    var idx_aldo_323 int = 0 // Inisialisasi indeks IPK tertinggi pada indeks
    pertama
    for j_aldo_323 := 1; j_aldo_323 < n_aldo_323; j_aldo_323++ {
        if T_aldo_323[idx_aldo_323].ipk < T_aldo_323[j_aldo_323].ipk {
            idx_aldo_323 = j_aldo_323 // Update indeks jika ditemukan IPK yang
            lebih tinggi
        }
    }
    return idx_aldo_323
}

// Fungsi main untuk menguji fungsi indeksIPKTertinggi_aldo_323
func main() {
    var n_aldo_323 int
    var dataMhs_aldo_323 arrMhs_aldo_323

    // Meminta input jumlah data mahasiswa
    fmt.Print("Masukkan jumlah mahasiswa (maks 2023): ")
    fmt.Scan(&n_aldo_323)

    // Validasi jumlah mahasiswa
    if n_aldo_323 < 1 || n_aldo_323 > 2023 {
```

```

        fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")
        return
    }

    // Memasukkan data mahasiswa
    fmt.Println("Masukkan data mahasiswa:")
    for i_aldo_323 := 0; i_aldo_323 < n_aldo_323; i_aldo_323++ {
        fmt.Printf("Mahasiswa ke-%d:\n", i_aldo_323+1)
        fmt.Print("Nama: ")
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].nama)
        fmt.Print("NIM: ")
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].nim)
        fmt.Print("Kelas: ")
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].kelas)
        fmt.Print("Jurusan: ")
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].jurusan)
        fmt.Print("IPK: ")
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].ipk)
    }

    // Mencari indeks mahasiswa dengan IPK tertinggi
    idxTertinggi_aldo_323 :=
    indeksIPKTertinggi_aldo_323(dataMhs_aldo_323, n_aldo_323)

    // Menampilkan data mahasiswa dengan IPK tertinggi
    fmt.Printf("\nMahasiswa dengan IPK tertinggi:\n")
    fmt.Printf("Nama: %s\n",
    dataMhs_aldo_323[idxTertinggi_aldo_323].nama)
    fmt.Printf("NIM: %s\n", dataMhs_aldo_323[idxTertinggi_aldo_323].nim)
    fmt.Printf("Kelas: %s\n",
    dataMhs_aldo_323[idxTertinggi_aldo_323].kelas)
    fmt.Printf("Jurusan: %s\n",
    dataMhs_aldo_323[idxTertinggi_aldo_323].jurusan)
    fmt.Printf("IPK: %.2f\n", dataMhs_aldo_323[idxTertinggi_aldo_323].ipk)
}

```



## Screenshoot output

```
PS C:\Users\aldop\OneDrive\Documents\Folder Baru> go run "c:\Users\aldop\OneDrive\Documents\Folder Baru\tempCodeRunnerFile.go"
Masukkan jumlah mahasiswa (maks 2023): 3
Masukkan data mahasiswa:
Mahasiswa ke-1:
Nama: aldo
NIM: 32
Kelas: 1
Jurusan: if
IPK: 3
Mahasiswa ke-2:
Nama: aldi
NIM: 23
Kelas: 2
Jurusan: if
IPK: 4
Mahasiswa ke-3:
Nama: aldowww
NIM: 34
Kelas: 3
Jurusan: if
IPK: 5

Mahasiswa dengan IPK tertinggi:
Nama: aldowww
NIM: 34
Kelas: 3
Jurusan: if
IPK: 5.00
```

## Deskripsi Program

Program ini bertujuan untuk mencari mahasiswa dengan Indeks Prestasi Kumulatif (IPK) tertinggi dari sekumpulan data mahasiswa yang diinputkan pengguna. Pengguna diminta memasukkan jumlah mahasiswa (maksimal 2023) dan data lengkap tiap mahasiswa, termasuk nama, NIM, kelas, jurusan, dan IPK. Program kemudian menggunakan fungsi `indeksIPKTertinggi_aldo_323` untuk menentukan indeks mahasiswa dengan IPK tertinggi, dan hasilnya ditampilkan dalam bentuk data lengkap mahasiswa tersebut. Program ini juga memastikan validitas jumlah mahasiswa yang diinputkan.



## 4. Sourcecode

```
package main

import "fmt"

// Definisi struct mahasiswa_aldo_323 dengan atribut nama, nim, kelas, jurusan, dan ipk
type mahasiswa_aldo_323 struct {

    nama, nim, kelas, jurusan string

    ipk          float64
}

// Definisi tipe data array mahasiswa_aldo_323 dengan kapasitas maksimal 2023
type arrMhs_aldo_323 [2023]mahasiswa_aldo_323

// Fungsi untuk mencari indeks IPK tertinggi dalam array mahasiswa_aldo_323
func indeksIPKTertinggi_aldo_323(T_aldo_323 arrMhs_aldo_323, n_aldo_323 int) int {

    var idx_aldo_323 int = 0 // Inisialisasi indeks IPK tertinggi pada indeks pertama

    for j_aldo_323 := 1; j_aldo_323 < n_aldo_323; j_aldo_323++ {

        if T_aldo_323[idx_aldo_323].ipk < T_aldo_323[j_aldo_323].ipk {

            idx_aldo_323 = j_aldo_323 // Update indeks jika ditemukan IPK yang lebih tinggi

        }

    }

    return idx_aldo_323

}
```

```
// Fungsi main untuk mengisi data mahasiswa dan mencari IPK tertinggi beserta indeks nya
```

```
func main() {
```

```
    var n_aldo_323 int
```

```
    var dataMhs_aldo_323 arrMhs_aldo_323
```

```
    // Meminta input jumlah mahasiswa
```

```
    fmt.Println("Masukkan jumlah mahasiswa (maks 2023): ")
```

```
    fmt.Scan(&n_aldo_323)
```

```
    // Validasi jumlah mahasiswa yang dimasukkan
```

```
    if n_aldo_323 < 1 || n_aldo_323 > 2023 {
```

```
        fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")
```

```
        return
```

```
    }
```

```
    // Mengisi data mahasiswa
```

```
    fmt.Println("Masukkan data mahasiswa:")
```

```
    for i_aldo_323 := 0; i_aldo_323 < n_aldo_323; i_aldo_323++ {
```

```
        fmt.Printf("Mahasiswa ke-%d:\n", i_aldo_323+1)
```

```
        fmt.Print("Nama: ")
```

```
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].nama)
```

```
        fmt.Print("NIM: ")
```

```
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].nim)
```

```
        fmt.Print("Kelas: ")
```

```
        fmt.Scan(&dataMhs_aldo_323[i_aldo_323].kelas)
```

```
    fmt.Print("Jurusan: ")

    fmt.Scan(&dataMhs_aldo_323[i_aldo_323].jurusan)

    fmt.Print("IPK: ")

    fmt.Scan(&dataMhs_aldo_323[i_aldo_323].ipk)

}

// Mencari indeks mahasiswa dengan IPK tertinggi

idxTertinggi_aldo_323 := indeksIPKTertinggi_aldo_323(dataMhs_aldo_323, n_aldo_323)

// Menampilkan data mahasiswa dengan IPK tertinggi

fmt.Printf("\nMahasiswa dengan IPK tertinggi:\n")

fmt.Printf("Nama: %s\n", dataMhs_aldo_323[idxTertinggi_aldo_323].nama)

fmt.Printf("NIM: %s\n", dataMhs_aldo_323[idxTertinggi_aldo_323].nim)

fmt.Printf("Kelas: %s\n", dataMhs_aldo_323[idxTertinggi_aldo_323].kelas)

fmt.Printf("Jurusan: %s\n", dataMhs_aldo_323[idxTertinggi_aldo_323].jurusan)

fmt.Printf("IPK: %.2f\n", dataMhs_aldo_323[idxTertinggi_aldo_323].ipk)

}
```

## Scrennshot output

```
PS C:\Users\aldop\OneDrive\Documents\Folder Baru> go run "c:\Users\aldop\OneDrive\Documents\Folder Baru\TempCodeRunnerFile.go"
Masukkan jumlah mahasiswa (maks 2023): 2
Masukkan data mahasiswa:
Mahasiswa ke-1:
Nama: aldo
NIM: 23
Kelas: 1
Jurusan: if
IPK: 4
Mahasiswa ke-2:
Nama: aldi
NIM: 32
Kelas: 2
Jurusan: if
IPK: 5

Mahasiswa dengan IPK tertinggi:
Nama: aldi
NIM: 32
Kelas: 2
Jurusan: if
IPK: 5.00
```

## Deskripsi output

Program ini digunakan untuk mencatat data mahasiswa dan menentukan mahasiswa dengan Indeks Prestasi Kumulatif (IPK) tertinggi. Pengguna akan diminta untuk memasukkan jumlah mahasiswa (maksimal 2023) dan data lengkap tiap mahasiswa, termasuk nama, NIM, kelas, jurusan, dan IPK. Program memproses data tersebut menggunakan fungsi `indeksiIPKTertinggi_aldo_323` untuk menentukan indeks mahasiswa dengan IPK tertinggi, dan hasilnya ditampilkan dengan detail lengkap. Validasi jumlah mahasiswa dilakukan untuk memastikan data yang diinput sesuai







## UNGUIDED

### 1. Sourcecode

```
2. package main
3.
4. import (
5.     "fmt"
6. )
7.
8. func main() {
9.     var N_aldo_323 int
10.    fmt.Print("Masukkan jumlah anak kelinci: ")
11.    fmt.Scan(&N_aldo_323)
12.
13.    if N_aldo_323 > 1000 {
14.        fmt.Println("Jumlah anak kelinci tidak boleh lebih dari 1000!")
15.        return
16.    }
17.
18.    berat_aldo_323 := make([]float64, N_aldo_323)
19.
20.    fmt.Printf("Masukkan berat %d anak kelinci:\n", N_aldo_323)
21.    for i_aldo_323 := 0; i_aldo_323 < N_aldo_323; i_aldo_323++ {
22.        fmt.Printf("Berat anak kelinci ke-%d: ", i_aldo_323+1)
23.        fmt.Scan(&berat_aldo_323[i_aldo_323])
24.    }
25.
26.    minBerat_aldo_323 := berat_aldo_323[0]
27.    maxBerat_aldo_323 := berat_aldo_323[0]
28.
29.    for i_aldo_323 := 1; i_aldo_323 < N_aldo_323; i_aldo_323++ {
30.        if berat_aldo_323[i_aldo_323] > maxBerat_aldo_323 {
31.            maxBerat_aldo_323 = berat_aldo_323[i_aldo_323]
32.        }
33.        if berat_aldo_323[i_aldo_323] < minBerat_aldo_323 {
34.            minBerat_aldo_323 = berat_aldo_323[i_aldo_323]
35.        }
36.    }
37.
38.    fmt.Printf("Berat terkecil: %.2f\n", minBerat_aldo_323)
39.    fmt.Printf("Berat terbesar: %.2f\n", maxBerat_aldo_323)
40. }
41.
42.
```

## Screenshoot Output

```
PS C:\Users\aldop\OneDrive\Documents\Folder Baru> go run "c:\Users\aldop\OneDrive\Documents\Folder Baru\tempCodeRunnerFile.go"
Masukkan jumlah anak kelinci: 2
Masukkan berat 2 anak kelinci:
Berat anak kelinci ke-1: 3
Berat anak kelinci ke-2: 4
Berat terkecil: 3.00
Berat terbesar: 4.00
PS C:\Users\aldop\OneDrive\Documents\Folder Baru>
```

## Deskripsi Program

Program ini digunakan untuk menentukan berat anak kelinci dengan nilai terkecil dan terbesar berdasarkan input pengguna. Pertama, pengguna memasukkan jumlah anak kelinci (maksimal 1000), kemudian program meminta berat masing-masing anak kelinci. Program menggunakan perulangan untuk mencari berat terkecil dan terbesar dari data yang dimasukkan. Hasil akhir berupa berat terkecil dan terbesar ditampilkan dengan format dua desimal. Program juga memvalidasi agar jumlah anak kelinci tidak melebihi 1000.

#### 43. Sourcecode

```
package main

import (
    "fmt"
)

func main() {

    var x, y int
    fmt.Print("Masukkan jumlah ikan (x) dan jumlah wadah (y): ")
    fmt.Scan(&x, &y)

    if x > 1000 || y > 1000 {
        fmt.Println("Jumlah ikan atau wadah tidak boleh lebih dari 1000!")
        return
    }

    ikan := make([]float64, x)
    fmt.Printf("Masukkan berat %d ikan:\n", x)
    for i := 0; i < x; i++ {
        fmt.Printf("Berat ikan ke-%d: ", i+1)
        fmt.Scan(&ikan[i])
    }

    totalPerWadah := make([]float64, y)
    for i := 0; i < x; i++ {

        indexWadah := i % y
        totalPerWadah[indexWadah] += ikan[i]
    }

    rataRataPerWadah := make([]float64, y)
    for i := 0; i < y; i++ {

        jumlahIkanDiWadah := (x + i) / y
        if jumlahIkanDiWadah > 0 {
            rataRataPerWadah[i] = totalPerWadah[i] /
float64(jumlahIkanDiWadah)
        } else {
            rataRataPerWadah[i] = 0
        }
    }
}
```

```

    }
}

fmt.Println("Total berat di setiap wadah:")
for i := 0; i < y; i++ {
    fmt.Printf("Wadah ke-%d: %.2f\n", i+1, totalPerWadah[i])
}

fmt.Println("Rata-rata berat di setiap wadah:")
for i := 0; i < y; i++ {
    fmt.Printf("Wadah ke-%d: %.2f\n", i+1, rataRataPerWadah[i])
}
}

```

## Screenshoot output

```

PROBLEMS 2 OUTPUT DEBUG CONSOLE TERMINAL SEARCH ERROR
Masukkan berat 2 ikan:
Berat ikan ke-1: 1
Berat ikan ke-2: 2
Total berat di setiap wadah:
Wadah ke-1: 1.00
Wadah ke-2: 2.00
Wadah ke-3: 0.00
Rata-rata berat di setiap wadah:
Wadah ke-1: 0.00
Wadah ke-2: 2.00
Wadah ke-3: 0.00

```

## Deskripsi program

Program di atas dirancang untuk menghitung total berat ikan dan rata-rata berat ikan di setiap wadah berdasarkan jumlah ikan (xxx) dan jumlah wadah (yyy). Program meminta pengguna untuk memasukkan data jumlah ikan dan wadah, diikuti dengan berat masing-masing ikan, yang kemudian didistribusikan secara berurutan ke wadah menggunakan logika modulus ( $i\%y$  \  $y$ ). Total berat untuk setiap wadah dihitung dengan menjumlahkan berat ikan yang masuk ke wadah tersebut, sedangkan rata-rata berat dihitung dengan membagi total berat wadah dengan jumlah ikan yang didistribusikan ke dalamnya. Hasil akhirnya berupa daftar total berat dan rata-rata berat untuk masing-masing wadah, yang dicetak dengan format desimal dua angka. Program ini dirancang untuk memastikan distribusi ikan ke wadah berjalan merata dan efisien, dengan kapasitas hingga 1000 ikan dan 1000 wadah.

**44.** 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 {
    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

    fmt.Print("Masukkan banyak data berat balita: ")
    fmt.Scan(&n)

    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
        fmt.Scan(&berat[i])
    }
}
```

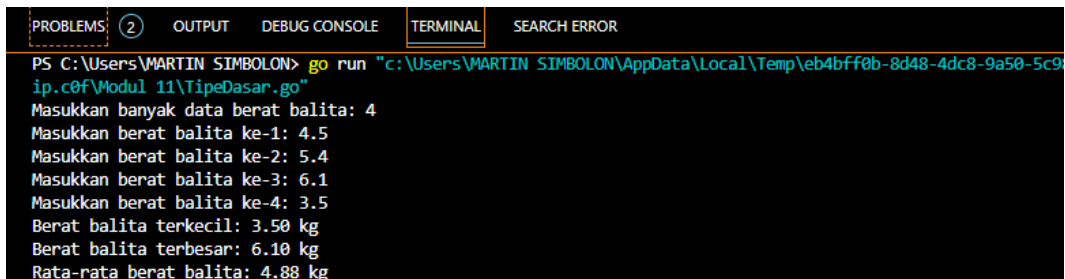
```

hitungMinMax(berat, n, &bMin, &bMax)
rataRata := rerata(berat, n)

fmt.Printf("Berat balita terkecil: %.2f kg\n", bMin)
fmt.Printf("Berat balita terbesar: %.2f kg\n", bMax)
fmt.Printf("Rata-rata berat balita: %.2f kg\n", rataRata)
}

```

### Scrennshoot output



```

PS C:\Users\MARTIN SIMBOLON> go run "c:\Users\MARTIN SIMBOLON\AppData\Local\Temp\eb4bff0b-8d48-4dc8-9a50-5c9
ip.c0f\Modul 11\TipeDasar.go"
Masukkan banyak data berat balita: 4
Masukkan berat balita ke-1: 4.5
Masukkan berat balita ke-2: 5.4
Masukkan berat balita ke-3: 6.1
Masukkan berat balita ke-4: 3.5
Berat balita terkecil: 3.50 kg
Berat balita terbesar: 6.10 kg
Rata-rata berat balita: 4.88 kg

```

### Deskripsi Program

Program ini dirancang untuk membantu petugas Pos Pelayanan Terpadu (Posyandu) mencatat dan menganalisis data berat balita dalam satuan kilogram. Program memungkinkan pengguna memasukkan jumlah balita dan berat masing-masing balita, yang kemudian disimpan dalam array statis dengan kapasitas maksimum 100 elemen. Program menggunakan dua fungsi utama: `hitungMinMax` untuk menghitung berat balita terkecil dan terbesar dalam array, serta `rerata` untuk menghitung rata-rata berat balita. Dengan pendekatan iterasi, program memproses data untuk menampilkan hasil berupa nilai minimum, maksimum, dan rata-rata berat balita, sehingga memudahkan petugas dalam melakukan evaluasi data kesehatan secara efisien dan akurat.