

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

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



Disusun Oleh :

Rafi Bintang Maulana / 2311102327

Kelas IF-11-06

Dosen Pengampu :

ABEDNEGO DWI SEPTIADI

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. DASAR TEORI

Pencarian nilai ekstrim merupakan salah satu proses yang umum dilakukan dalam kehidupan sehari-hari, dengan aplikasi yang sangat beragam, seperti pencarian file terbesar di dalam direktori komputer, pencarian teks tertentu di dokumen, atau pencarian buku pada rak buku. Nilai ekstrim yang dimaksud adalah nilai terbesar (maksimum) atau nilai terkecil (minimum) dari suatu himpunan data. Proses ini biasanya dilakukan secara sekuensial, di mana setiap elemen data dibandingkan satu per satu.

Algoritma pencarian nilai ekstrim dirancang dengan konsep sederhana, yaitu membandingkan nilai pada elemen data secara berurutan. Nilai maksimum atau minimum sementara disimpan sebagai referensi dan dibandingkan dengan elemen berikutnya. Nilai yang lebih besar (untuk pencarian maksimum) atau lebih kecil (untuk pencarian minimum) akan menggantikan nilai sementara tersebut. Proses ini berlanjut hingga seluruh elemen data selesai diproses, menghasilkan nilai maksimum atau minimum dari himpunan data tersebut.

II. GUIDED

1. TipeDasarIDX

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
```

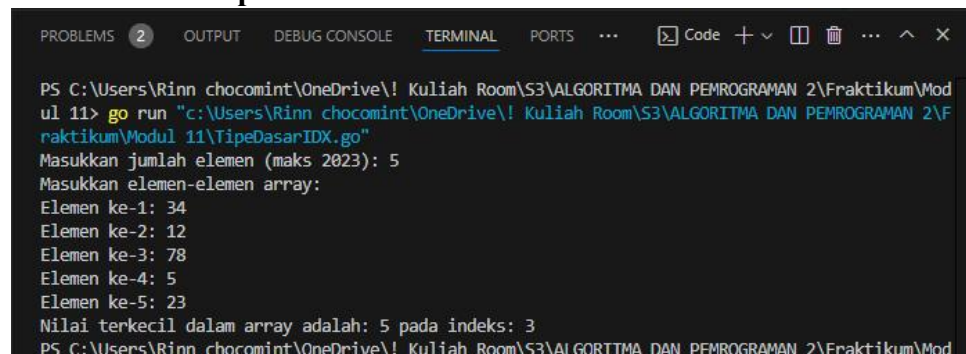
```

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\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11> go run "c:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11\TipeDasarIDX.go"
Masukkan jumlah elemen (maks 2023): 5
Masukkan elemen-elemen array:
Elemen ke-1: 34
Elemen ke-2: 12
Elemen ke-3: 78
Elemen ke-4: 5
Elemen ke-5: 23
Nilai terkecil dalam array adalah: 5 pada indeks: 3
PS C:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Mod

```

Deskripsi Program

Program itu untuk menemukan nilai terkecil dalam sebuah array menggunakan fungsi “**terkecil**”. Fungsi ini menerima array “**arrInt**” dan jumlah elemen “**n**”, kemudian memprosesnya secara iteratif untuk membandingkan setiap elemen, mencari indeks elemen dengan nilai terkecil, dan mengembalikannya. Dimulai dengan meminta pengguna memasukkan jumlah elemen dan nilai-nilai array, kemudian memanggil fungsi “**terkecil**” untuk menentukan indeks nilai terkecil. Hasil akhirnya menampilkan nilai terkecil beserta indeksnya dalam array.

2. TipeStrukturIPK

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)

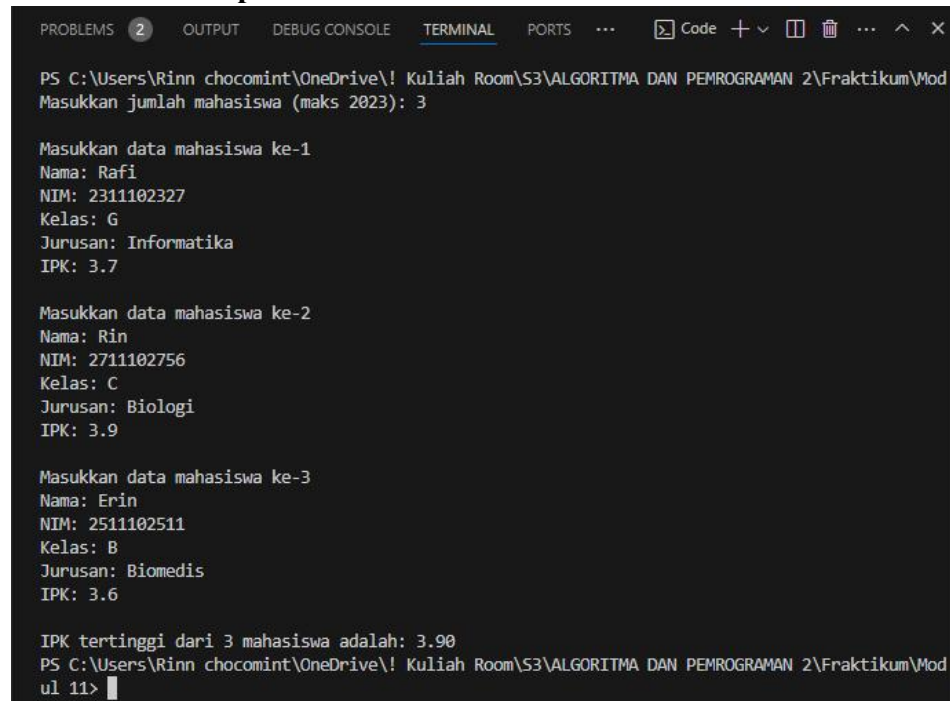
    // Validasi jumlah mahasiswa yang dimasukkan
    if n < 1 || n > 2023 {
        fmt.Println("Jumlah mahasiswa harus antara 1 dan 2023.")
        return
    }

    // 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



```
PS C:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Mod
Masukkan jumlah mahasiswa (maks 2023): 3

Masukkan data mahasiswa ke-1
Nama: Rafi
NIM: 2311102327
Kelas: G
Jurusan: Informatika
IPK: 3.7

Masukkan data mahasiswa ke-2
Nama: Rin
NIM: 2711102756
Kelas: C
Jurusan: Biologi
IPK: 3.9

Masukkan data mahasiswa ke-3
Nama: Erin
NIM: 2511102511
Kelas: B
Jurusan: Biomedis
IPK: 3.6

IPK tertinggi dari 3 mahasiswa adalah: 3.90
PS C:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Mod
ul 11> |
```

Deskripsi Program

Program bertujuan untuk mencari IPK tertinggi dari daftar mahasiswa yang diinputkan. Data mahasiswa disimpan dalam array “**arrMhs**”, yang berisi informasi seperti nama, NIM, kelas, jurusan, dan IPK. Fungsi “**ipk**” digunakan untuk memproses array dan mencari IPK tertinggi melalui perbandingan iteratif antar elemen. Akhirnya, program menampilkan nilai IPK tertinggi dari daftar mahasiswa yang diinputkan.

III. UNGUIDED

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 jumlahAnak int
    var BobotBeratAnak [1000] float64
    var beratPalingKecil, beratPalingGede float64

    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&jumlahAnak)

    if jumlahAnak < 1 || jumlahAnak > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1 dan 1000.")
        return
    }

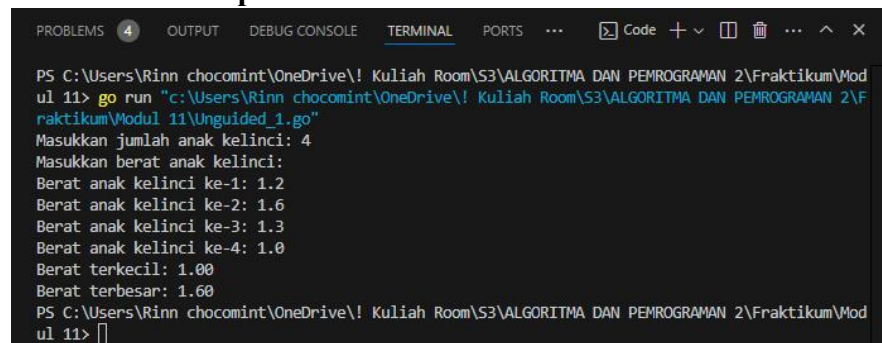
    fmt.Println("Masukkan berat anak kelinci:")
    for i := 0; i < jumlahAnak; i++ {
        fmt.Printf("Berat anak kelinci ke-%d: ", i+1)
        fmt.Scan(&BobotBeratAnak[i])
    }

    beratPalingKecil = BobotBeratAnak[0]
    beratPalingGede = BobotBeratAnak[0]

    for i := 1; i < jumlahAnak; i++ {
        if BobotBeratAnak[i] < beratPalingKecil {
            beratPalingKecil = BobotBeratAnak[i]
        }
        if BobotBeratAnak[i] > beratPalingGede {
            beratPalingGede = BobotBeratAnak[i]
        }
    }

    fmt.Printf("Berat terkecil: %.2f\n", beratPalingKecil)
    fmt.Printf("Berat terbesar: %.2f\n", beratPalingGede)
}
```

Screenshoot Output



```
PS C:\Users\Rinn chocomint\OneDrive\! Kulia Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11> go run "c:\Users\Rinn chocomint\OneDrive\! Kulia Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11\Unguided_1.go"
Masukkan jumlah anak kelinci: 4
Masukkan berat anak kelinci:
Berat anak kelinci ke-1: 1.2
Berat anak kelinci ke-2: 1.6
Berat anak kelinci ke-3: 1.3
Berat anak kelinci ke-4: 1.0
Berat terkecil: 1.00
Berat terbesar: 1.60
PS C:\Users\Rinn chocomint\OneDrive\! Kulia Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11> []
```

Deskripsi Program

Program ini digunakan untuk menentukan berat terkecil dan terbesar dari sejumlah anak kelinci yang beratnya dimasukkan oleh pengguna. Setelah meminta jumlah anak kelinci, program menerima input berat masing-masing kelinci, yang disimpan dalam array "**BobotBeratAnak**". Nilai berat terkecil variabel "**beratPalingKecil**" dan terbesar "**beratPalingGede**" diinisialisasi dengan berat kelinci pertama, kemudian diperbarui melalui iterasi dengan membandingkan setiap elemen dalam array. Hasil akhirnya adalah dua nilai ekstrem yang dicetak sebagai output.

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.

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 jumlahIkan, kapasitasWadah int
    var beratIkan [1000]float64

    fmt.Print("Masukkan jumlah ikan dan kapasitas wadah (x y): ")
    fmt.Scan(&jumlahIkan, &kapasitasWadah)

    if jumlahIkan < 1 || jumlahIkan > 1000 || kapasitasWadah < 1 {
        fmt.Println("Jumlah ikan harus antara 1 dan 1000, kapasitas wadah harus lebih dari 0.")
        return
    }

    fmt.Println("Masukkan berat ikan:")
    for i := 0; i < jumlahIkan; i++ {
        fmt.Printf("Berat ikan ke-%d: ", i+1)
```



```

        fmt.Scan(&beratIkan[i])
    }

    var jumlahWadah = (jumlahIkan + kapasitasWadah - 1) / kapasitasWadah
    var totalBeratPerWadah []float64
    var totalBerat float64

    for i := 0; i < jumlahWadah; i++ {
        var beratWadah float64 = 0
        for j := i * kapasitasWadah; j < (i+1)*kapasitasWadah && j <
jumlahIkan; j++ {
            beratWadah += beratIkan[j]
        }
        totalBeratPerWadah = append(totalBeratPerWadah, beratWadah)
        totalBerat += beratWadah
    }

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

    var rataRataBerat float64 = totalBerat / float64(jumlahWadah)
    fmt.Printf("Berat rata-rata ikan di setiap wadah: %.2f\n", rataRataBerat)
}

```

Screenshoot Output

```

PS C:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11> go run "c:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11\Unguided_2.go"
Masukkan jumlah ikan dan kapasitas wadah (x y): 4 2
Masukkan berat ikan:
Berat ikan ke-1: 3
Berat ikan ke-2: 2
Berat ikan ke-3: 1
Berat ikan ke-4: 1.2
Total berat ikan di setiap wadah:
Wadah ke-1: 5.00
Wadah ke-2: 2.20
Berat rata-rata ikan di setiap wadah: 3.60
PS C:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Mod

```

Deskripsi Program

Program itu berguna untuk menghitung total berat ikan di setiap wadah serta berat rata-rata ikan di setiap wadah berdasarkan masukan berat ikan yang akan dijual. Masukan berupa jumlah ikan dan kapasitas wadah, diikuti oleh berat masing-masing ikan. Array “**beratIkan**” digunakan untuk menyimpan berat ikan, perhitungan total berat di setiap wadah dilakukan dengan iterasi dalam blok ikan yang sesuai kapasitas wadah. Total berat dari semua wadah disimpan dalam “**totalBeratPerWadah**”,

Jadi fungsi utama dari program ini adalah menghitung rata-rata berat ikan dengan membagi total berat seluruh ikan dengan jumlah wadah.

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 */
  ...
}
```

Sourcecode

```
package main
import (
    "fmt"
)

type arrBalita [100]float64
func hitungMinMax(arrBerat arrBalita, jumlah int, bMin, bMax *float64) {
    *bMin = arrBerat[0]
    *bMax = arrBerat[0]

    for i := 1; i < jumlah; i++ {
        if arrBerat[i] < *bMin {
            *bMin = arrBerat[i]
        }
        if arrBerat[i] > *bMax {
            *bMax = arrBerat[i]
        }
    }
}

func rerata(arrBerat arrBalita, jumlah int) float64 {
    var total float64
    for i := 0; i < jumlah; i++ {
        total += arrBerat[i]
    }
    return total / float64(jumlah)
}

func main() {
```

```

var jumlahBalita int
var beratBalita arrBalita
var beratTerkecil, beratTerbesar, rataRata float64

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

if jumlahBalita < 1 || jumlahBalita > 100 {
    fmt.Println("Jumlah balita harus antara 1 dan 100.")
    return
}

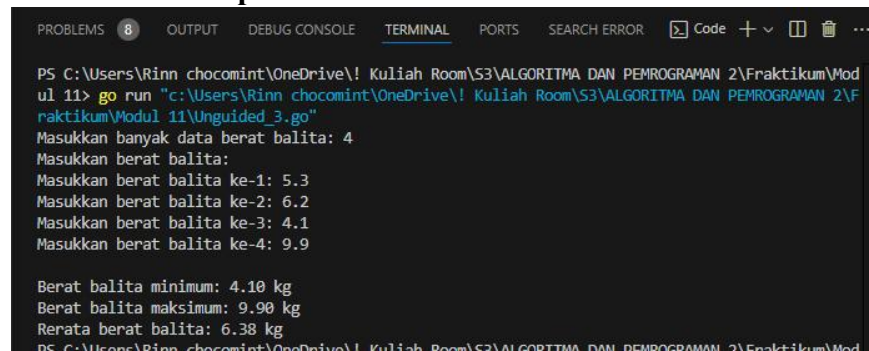
fmt.Println("Masukkan berat balita:")
for i := 0; i < jumlahBalita; i++ {
    fmt.Printf("Masukkan berat balita ke-%d: ", i+1)
    fmt.Scan(&beratBalita[i])
}

hitungMinMax(beratBalita, jumlahBalita, &beratTerkecil, &beratTerbesar)
rataRata = rerata(beratBalita, jumlahBalita)

fmt.Printf("\nBerat balita minimum: %.2f kg\n", beratTerkecil)
fmt.Printf("Berat balita maksimum: %.2f kg\n", beratTerbesar)
fmt.Printf("Rerata berat balita: %.2f kg\n", rataRata)
}

```

Screenshoot Output



```

PS C:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11> go run "c:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Modul 11\Unguided 3.go"
Masukkan banyak data berat balita: 4
Masukkan berat balita:
Masukkan berat balita ke-1: 5.3
Masukkan berat balita ke-2: 6.2
Masukkan berat balita ke-3: 4.1
Masukkan berat balita ke-4: 9.9

Berat balita minimum: 4.10 kg
Berat balita maksimum: 9.90 kg
Rerata berat balita: 6.38 kg
PS C:\Users\Rinn chocomint\OneDrive\! Kuliah Room\S3\ALGORITMA DAN PEMROGRAMAN 2\Fraktikum\Mod

```

Deskripsi Program

Program ini bisa untuk membantu petugas Posyandu mencatat berat balita dan menghitung statistik seperti berat terkecil, terbesar, dan rata-rata. Data berat balita disimpan dalam array “**arrBalita**”. Fungsi “**hitungMinMax**” digunakan untuk menentukan berat minimum dan maksimum dengan membandingkan setiap elemen array. Fungsi “**rerata**” menghitung rata-rata berat balita dengan menjumlahkan semua elemen dan membaginya dengan jumlah balita. Program ini menghasilkan keluaran berupa berat terkecil, terbesar, dan rata-rata balita dalam format yang jelas dan terstruktur.