LAPORAN PRAKTIKUM
ALGORITME DAN PEMROGRAMAN 2

MODUL 11
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

Oleh:

MUHAMMAD AMIR SALEH

2311102233

IF - 11 - 06

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. Dasar Teori

Pencarian adalah proses mencari elemen atau nilai tertentu dalam sebuah himpunan data, seperti array atau list. Tujuannya adalah untuk menemukan apakah nilai yang dicari ada dalam himpunan tersebut atau untuk mengetahui posisi elemen tersebut. Pada pencarian nilai ekstrem, seperti nilai maksimum atau minimum, pada sebuah himpunan data, algoritmanya bisa dilakukan dengan cara iterasi melalui seluruh data, membandingkan setiap elemen untuk menemukan nilai yang paling besar atau paling kecil. Proses ini biasanya dilakukan dalam satu kali iterasi, namun, dalam beberapa kasus, jika data sudah terurut, kita dapat langsung mengambil elemen pertama atau terakhir sebagai nilai minimum atau maksimum tanpa harus melakukan iterasi penuh.

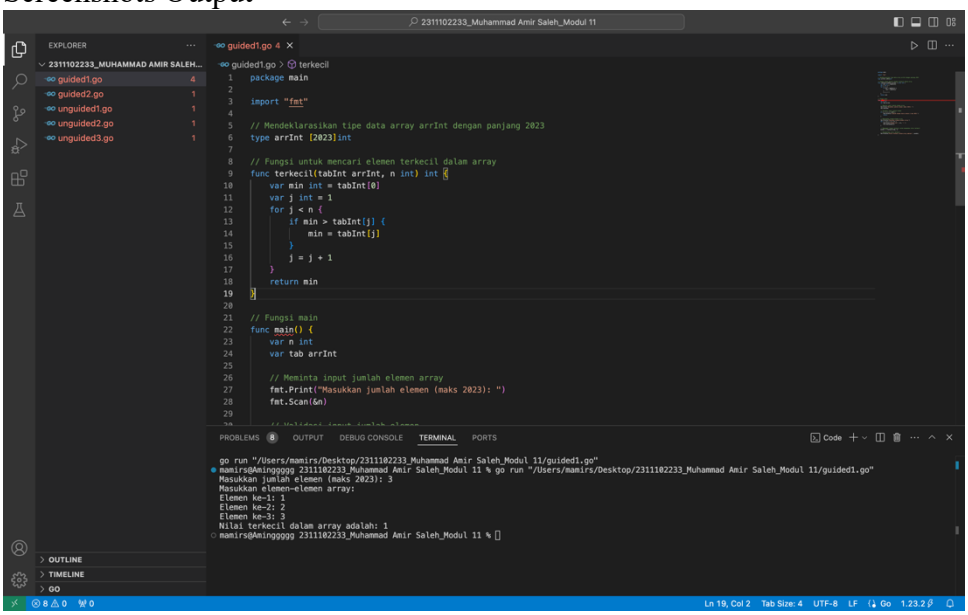
II. Guided

Guided 1

```
package main

import "fmt"

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

// Fungsi untuk mencari elemen terkecil dalam array
func terkecil(tabInt arrInt, n int) int {
    var min int = tabInt[0]
    var j int = 1
    for j < n {
        if min > tabInt[j] {
            min = tabInt[j]
        }
        j = j + 1
    }
    return min
}

// Fungsi main
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 nilai terkecil
    minVal := terkecil(tab, n)
```

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

Screenshots Output



Deskripsi:

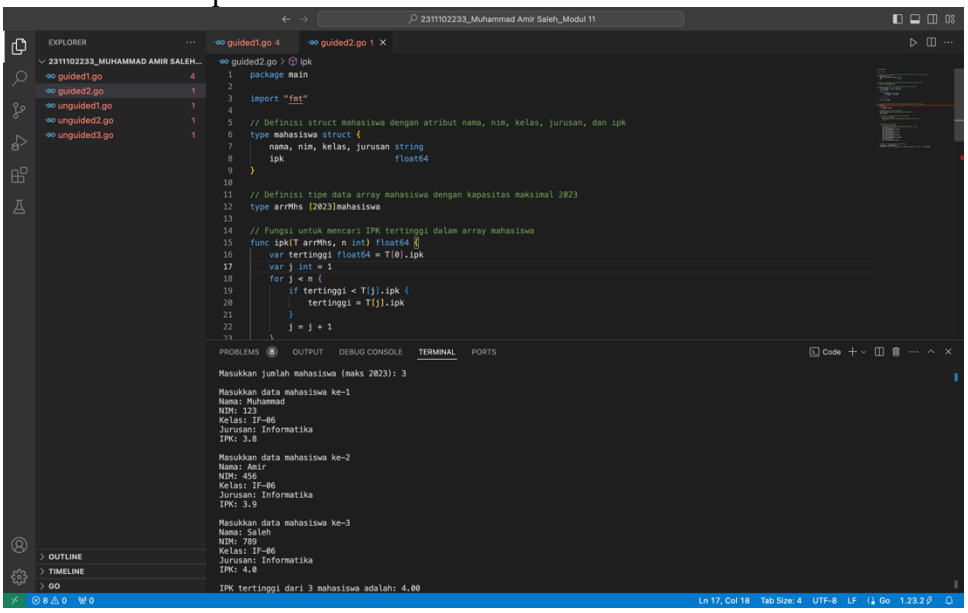
Program ini adalah program yang mencari nilai terkecil dalam sebuah array yang berisi sejumlah elemen yang dimasukkan oleh pengguna. Array yang digunakan memiliki tipe arrInt dengan panjang maksimum 2023 elemen. Fungsi terkecil digunakan untuk mencari nilai terkecil dengan cara memulai pencarian dari elemen pertama dan kemudian membandingkan setiap elemen berikutnya. Jika ditemukan nilai yang lebih kecil, nilai tersebut akan menjadi nilai terkecil sementara. Di bagian utama program, pengguna diminta untuk memasukkan jumlah elemen yang ingin digunakan dalam array, dengan validasi bahwa jumlah elemen harus antara 1 dan 2023. Setelah jumlah elemen divalidasi, pengguna diminta untuk memasukkan nilai-nilai elemen array. Setelah semua elemen dimasukkan, fungsi terkecil dipanggil untuk menemukan nilai terkecil dalam array, dan hasilnya ditampilkan ke layar.

Guided 2

```
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)

}
```

Screenshots Output



Deskripsi:

PProgram ini adalah program yang mencari IPK tertinggi dari sejumlah data mahasiswa yang dimasukkan oleh pengguna. Program ini menggunakan struktur mahasiswa yang berisi informasi seperti nama, NIM, kelas, jurusan, dan IPK, dan data mahasiswa disimpan dalam array bertipe

arrMhs dengan kapasitas hingga 2023 mahasiswa. Fungsi ipk digunakan untuk mencari IPK tertinggi dengan cara membandingkan IPK mahasiswa satu per satu, dan jika ditemukan IPK yang lebih tinggi, nilai tertinggi diperbarui. Di bagian utama program, pengguna diminta untuk memasukkan jumlah mahasiswa dengan validasi jumlah antara 1 dan 2023, lalu memasukkan data mahasiswa, termasuk nama, NIM, kelas, jurusan, dan IPK. Setelah itu, fungsi ipk dipanggil untuk mencari dan menampilkan IPK tertinggi dari mahasiswa yang ada.

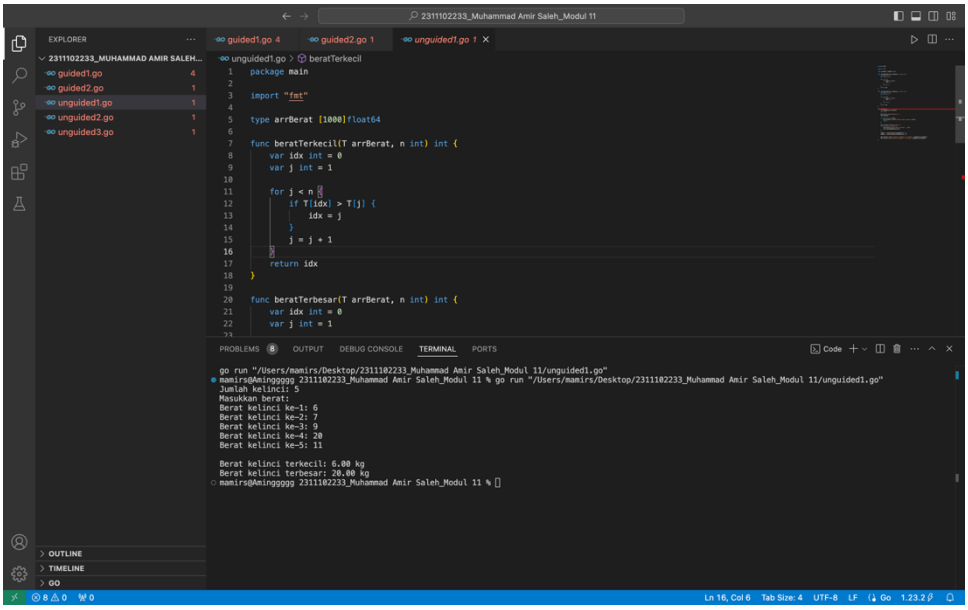
III. Unguided

Unguided 1

```
package main

import "fmt"

type arrBerat [1000]float64

func beratTerkecil(T arrBerat, n int) int {
    var idx int = 0
    var j int = 1

    for j < n {
        if T[idx] > T[j] {
            idx = j
        }
        j = j + 1
    }
    return idx
}

func beratTerbesar(T arrBerat, n int) int {
    var idx int = 0
    var j int = 1

    for j < n {
        if T[idx] < T[j] {
            idx = j
        }
        j = j + 1
    }
    return idx
}

func main() {
    var dataKelinci arrBerat
    var n int

    fmt.Print("Jumlah kelinci: ")
    fmt.Scan(&n)

    if n <= 0 || n > 1000 {
        fmt.Println("Jumlah kelinci harus antara 1–1000")
        return
    }
```

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

    idxMin := beratTerkecil(dataKelinci, n)
    idxMax := beratTerbesar(dataKelinci, n)

    fmt.Printf("\nBerat kelinci terkecil: %.2f kg\n",
dataKelinci[idxMin])
    fmt.Printf("Berat kelinci terbesar: %.2f kg\n",
dataKelinci[idxMax])
}
```

Screenshots Output



Deskripsi:

Program ini adalah program yang menghitung dan menampilkan berat terkecil serta berat terbesar dari sejumlah data berat kelinci yang dimasukkan pengguna. Data berat kelinci disimpan dalam sebuah array bertipe arrBerat dengan kapasitas maksimum 1000 elemen. Fungsi beratTerkecil mencari indeks kelinci dengan berat paling kecil, sementara fungsi beratTerbesar mencari indeks kelinci dengan berat terbesar. Kedua fungsi ini membandingkan elemen-elemen dalam array untuk menentukan indeks yang sesuai. Di bagian utama program, pengguna diminta untuk memasukkan jumlah kelinci dan berat masing-masing kelinci. Jika jumlah kelinci tidak valid (di luar rentang 1 hingga 1000), program akan memberi

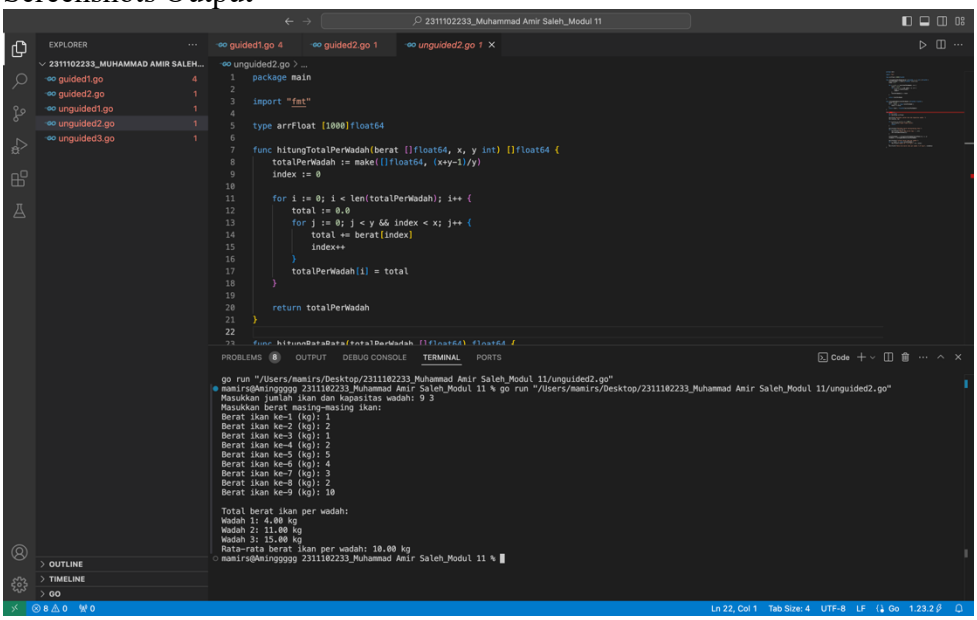
pesan kesalahan dan berhenti. Setelah data dimasukkan, program menggunakan kedua fungsi tersebut untuk menemukan indeks kelinci dengan berat terkecil dan terbesar, kemudian mencetak berat kelinci terkecil dan terbesar yang ditemukan.

Unguided 2

```
package main

import "fmt"

type arrFloat [1000]float64

func hitungTotalPerWadah(berat []float64, x, y int) []float64 {
    totalPerWadah := make([]float64, (x+y-1)/y)
    index := 0

    for i := 0; i < len(totalPerWadah); i++ {
        total := 0.0
        for j := 0; j < y && index < x; j++ {
            total += berat[index]
            index++
        }
        totalPerWadah[i] = total
    }

    return totalPerWadah
}

func hitungRataRata(totalPerWadah []float64) float64 {
    total := 0.0
    for _, nilai := range totalPerWadah {
        total += nilai
    }
    return total / float64(len(totalPerWadah))
}

func main() {
    var x, y int
    var beratIkan arrFloat

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

    if x <= 0 || y <= 0 || x > 1000 {
        fmt.Println("Input tidak valid")
        return
    }

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

```
        fmt.Scan(&beratIkan[i])
    }

    totalPerWadah := hitungTotalPerWadah(beratIkan[:x], x, y)
    rataRata := hitungRataRata(totalPerWadah)

    fmt.Println("\nTotal berat ikan per wadah:")
    for i, total := range totalPerWadah {
        fmt.Printf("Wadah %d: %.2f kg\n", i+1, total)
    }
    fmt.Printf("Rata-rata berat ikan per wadah: %.2f kg\n",
rataRata)
}
```

Screenshots Output



Deskripsi:

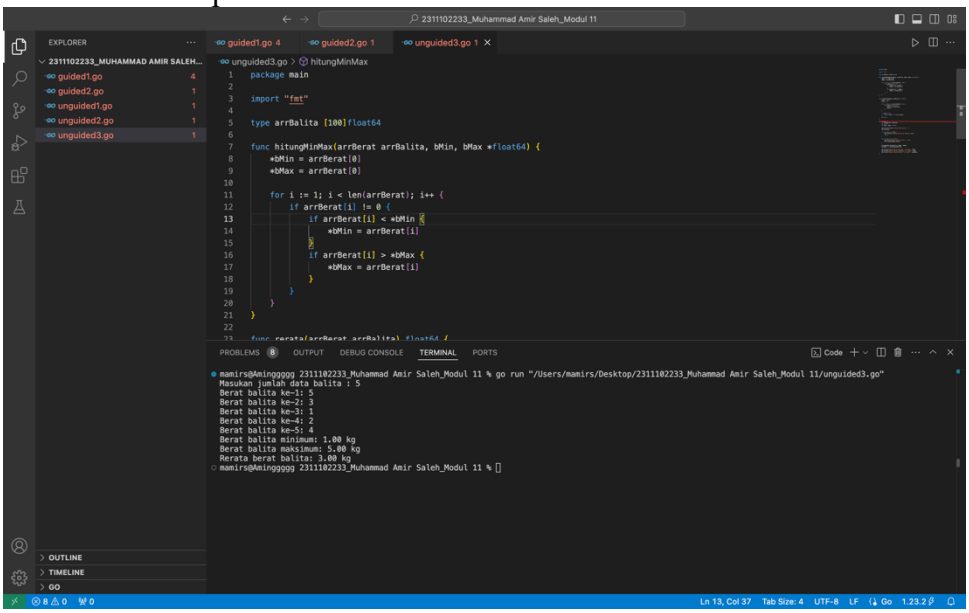
Program ini adalah program yang menghitung total berat ikan di setiap wadah dan rata-rata berat ikan per wadah berdasarkan jumlah ikan, berat masing-masing ikan, dan kapasitas wadah yang dimasukkan oleh pengguna. Program ini menggunakan array arrFloat untuk menyimpan berat ikan dan fungsi hitungTotalPerWadah untuk mengelompokkan ikan ke dalam wadah sesuai kapasitas, lalu menghitung total berat ikan di setiap wadah. Fungsi hitungRataRata digunakan untuk menghitung rata-rata berat ikan per wadah dengan menjumlahkan total berat dari semua wadah dan membaginya dengan jumlah wadah. Setelah pengguna memasukkan jumlah ikan, kapasitas wadah, dan berat masing-masing ikan, program memvalidasi input dan menghitung total berat per wadah serta rata-rata berat per wadah, kemudian mencetak hasilnya.

Unguided 3

```
package main

import "fmt"

type arrBalita [100]float64

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

    for i := 1; i < len(arrBerat); i++ {
        if arrBerat[i] != 0 {
            if arrBerat[i] < *bMin {
                *bMin = arrBerat[i]
            }
            if arrBerat[i] > *bMax {
                *bMax = arrBerat[i]
            }
        }
    }
}

func rerata(arrBerat arrBalita) float64 {
    total := 0.0
    count := 0

    for i := 0; i < len(arrBerat); i++ {
        if arrBerat[i] != 0 {
            total += arrBerat[i]
            count++
        }
    }

    if count > 0 {
        return total / float64(count)
    }
    return 0
}

func main() {
    var dataBalita arrBalita
    var n int
    var bMin, bMax float64

    fmt.Print("Masukan jumlah data balita : ")
```

```
    fmt.Scan(&n)

    if n <= 0 || n > 100 {
        fmt.Println("Jumlah data harus antara 1-100")
        return
    }

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

    hitungMinMax(dataBalita, &bMin, &bMax)
    rataRata := rerata(dataBalita)

    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", rataRata)
}
```

Screenshots Output



Deskripsi:

Program ini adalah program yang menghitung berat balita minimum, maksimum, dan rata-rata berdasarkan data berat balita yang dimasukkan pengguna. Data berat balita disimpan dalam array dengan kapasitas maksimum 100 elemen. Fungsi hitungMinMax digunakan untuk mencari berat balita terkecil dan terbesar dengan membandingkan setiap elemen dalam array dan memperbarui nilai minimum atau maksimum.

Fungsi rerata menghitung rata-rata berat balita dengan menjumlahkan berat balita yang ada dan membaginya dengan jumlah balita yang memiliki data berat. Di bagian utama program, pengguna diminta untuk memasukkan jumlah data balita dan berat masing-masing balita, kemudian program memeriksa validitas input dan menghitung berat balita minimum, maksimum, dan rata-rata dengan menggunakan fungsi yang telah dibuat. Program kemudian mencetak hasil perhitungan tersebut.