LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2
MODUL 7
PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

DISUSUN OLEH:

ANASTASIA ADINDA NARENDRA INDRIANTO

103112400085

S1 IF-12-01

DOSEN:

Dimas Fanny Hebrasianto Permadi, S.ST., M.KOM

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

1. Pengertian Bahasa Pemorgraman Golang

Golang adalah bahasa pemrograman yang memiliki sejumlah kelebihan yang tidak dimiliki bahasa pemrograman yang lainnya. Hadirnya bahasa pemrograman Go Language (Golang) semakin dirasakan oleh para pengembang. Tidak heran jika banyak orang yang mulai belajar bahasa pemrograman yang satu ini.
Golang merupakan bahasa pemrograman yang dibuat Google dan tujuannya untuk menyempurnakan bahasa pemrograman yang ada, seperti C, Phyton dan yang lainnya. Golang bisa jadi pilihan yang tepat saat membuat aplikasi baru.

2. Pengertian Input dan Output

i. *Input* atau masukan adalah data yang diberikan ke dalam program. Masukan ini bisa berasal dari berbagai sumber, seperti pengguna melalui keyboard, mouse, atau suara, dan juga bisa berasal dari sensor atau perangkat lain yang terhubung ke komputer. Dalam pemrograman, input diperlakukan sebagai bahan baku yang akan diproses oleh program.

ii. *Output* atau keluaran adalah hasil yang diproduksi oleh program setelah mengolah input. Output bisa berbentuk tampilan pada layar, cetakan pada printer, suara, ataupun penyimpanan data ke file. Inti dari program yang kita tulis sebenarnya adalah menghasilkan output yang bermakna dari input yang diberikan.

3. Pengertian Tipe Data

Tipe data adalah kategori data yang akan diproses oleh sebuah program komputer. Dengan tipe data, program dapat mengetahui cara menangani data dengan benar, seperti melakukan operasi matematika pada angka atau memanipulasi string untuk tujuan tertentu.Secara sederhana, pengertian tipe data adalah kategori data yang akan diproses oleh sebuah program komputer. Dengan tipe data, program dapat mengetahui cara menangani data dengan benar, seperti melakukan operasi matematika pada angka atau memanipulasi string untuk tujuan tertentu. Contoh paling sederhana dari tipe data adalah tipe data **integer** yang digunakan untuk menyimpan angka bulat atau tipe data **string** yang digunakan untuk menyimpan rangkaian karakter. Penggunaan tipe data yang benar dalam suatu program memastikan bahwa data diolah secara tepat dan mengurangi risiko kesalahan atau bug dalam program tersebut.

4. Fungsi Tipe Data

i. **Menentukan Jenis Nilai**: Tipe data memberi tahu program jenis nilai yang akan disimpan dalam variabel. Misalnya, jika kita ingin menyimpan angka, kita menggunakan tipe data integer atau float, sedangkan untuk menyimpan teks, kita menggunakan tipe data string. Ini membantu program memahami bagaimana cara menangani data tersebut.

ii. **Efisiensi Penggunaan Memori**: Setiap tipe data memerlukan jumlah memori yang berbeda. Jika kita memilih tipe data yang tepat, program bisa menggunakan memori lebih efisien. Misalnya, menggunakan tipe data yang lebih kecil untuk angka yang tidak terlalu besar akan menghemat ruang di memori.

iii. **Menjamin Konsistensi Data**: Dengan menentukan tipe data, kita memastikan bahwa variabel hanya bisa menyimpan jenis nilai yang sesuai. Misalnya, jika kita mendeklarasikan variabel sebagai tipe integer, program tidak akan bisa memasukkan teks atau jenis data lainnya ke dalamnya. Ini membantu menjaga agar data tetap konsisten dan sesuai dengan yang diharapkan.

iv. **Memudahkan Operasi pada Data**: Tipe data juga menentukan operasi apa saja yang bisa dilakukan pada data. Misalnya, kita bisa melakukan perhitungan

matematika pada angka, tetapi kita tidak bisa melakukan hal yang sama pada teks. Jadi, dengan menentukan tipe data yang tepat, kita bisa melakukan operasi yang sesuai dengan jenis data yang kita miliki.

5. **Pengertian If-Else dan Fungsinya**

   If-else adalah struktur percabangan dalam pemrograman yang digunakan untuk mengeksekusi kode berdasarkan suatu kondisi. Jika kondisi dalam if bernilai true, maka blok kode di dalamnya akan dijalankan. Jika tidak, program akan memeriksa kondisi dalam else if sebagai alternatif. Jika semua kondisi false, maka else akan dijalankan sebagai pilihan terakhir. Dengan menggunakan if-else, program dapat mengambil keputusan dan menjalankan instruksi yang sesuai berdasarkan kondisi yang diberikan.

   If-else berfungsi untuk mengontrol alur program dengan mengeksekusi kode berdasarkan suatu kondisi. If digunakan untuk memeriksa apakah suatu kondisi bernilai true, jika ya, maka blok kode di dalamnya akan dijalankan. Jika tidak, program dapat menggunakan else-if untuk mengevaluasi beberapa kondisi tambahan secara berurutan. Jika semua kondisi false, maka else akan dieksekusi sebagai pilihan terakhir. Selain itu, terdapat if-else bersarang, yang memungkinkan pengecekan kondisi di dalam kondisi lain untuk menangani keputusan yang lebih kompleks.

6. **Pengertian While Loop dan Fungsinya**

   While loop adalah metode perulangan yang mengeksekusi blok kode selama kondisi yang diberikan bernilai true dan akan berhenti ketika kondisi berubah menjadi false. Perulangan ini sangat berguna dalam kasus di mana jumlah iterasi belum diketahui secara pasti, seperti membaca input hingga valid atau menjalankan suatu proses hingga syarat tertentu terpenuhi. Fungsinya adalah;

   i. **Menjalankan Perulangan Berdasarkan Kondisi** – While loop memastikan suatu proses terus berjalan selama kondisi bernilai **true**.
   ii. **Mengatasi Iterasi yang Tidak Diketahui Jumlahnya** – Digunakan ketika jumlah perulangan tidak bisa ditentukan sejak awal, seperti menunggu input yang valid.
   iii. **Mengoptimalkan Kontrol Program** – Membantu mengelola eksekusi kode agar hanya berjalan saat kondisi tertentu terpenuhi, meningkatkan efisiensi program.

7. **Pengertian Fungsi dan Manfaatnya**

   Fungsi merupakan rangkaian instruksi yang menghasilkan suatu nilai dengan memetakkan input ke output tertentu. Dalam pemrograman, suatu subprogram dikategorikan sebagai fungsi apabila memiliki deklarasi tipe nilai yang dikembalikan dan menggunakan kata kunci return dalam tubuhnya. Fungsi digunakan dalam berbagai situasi, seperti menetapkan nilai ke suatu variabel melalui assignment, menjadi bagian dari suatu ekspresi, atau digunakan sebagai argumen dalam subprogram lain. Karena fungsi selalu menghasilkan nilai, penamaannya sebaiknya bersifat deskriptif dan mencerminkan hasil yang diberikan, seperti *median*, *rerata*, *nilaiTerbesar*, atau *selesai*.

   Function memungkinkan programmer membagi kode menjadi segmen-segmen kecil yang lebih terkelola, masing-masing melakukan bagian tertentu dari tugas yang lebih besar. Tidak hanya membantu dalam mengorganisasi kode secara lebih efisien, hal ini juga memudahkan pemeliharaan dan pengujian kode.

8. **Pengertian Prosedur dan Manfaatnya**
   Prosedur adalah kumpulan instruksi yang dikemas menjadi satu kesatuan untuk menyederhanakan kode dalam program besar. Prosedur tidak mengembalikan nilai karena tidak memiliki deklarasi tipe nilai kembalian dan tidak menggunakan kata kunci *return*. Ketika dipanggil, prosedur langsung memberikan efek pada program utama, seperti halnya instruksi dasar atau fungsi bawaan (*built-in*). Nama prosedur sebaiknya menggunakan kata kerja atau kata yang menggambarkan proses, seperti *cetak*, *hitungRerata*, atau *cariNilai*. Hal ini bertujuan agar lebih mudah dipahami dan sesuai dengan fungsinya dalam program. Dengan menggunakan prosedur, kode program menjadi lebih terstruktur, mudah dibaca, dan dikelola.
   Manfaat utama prosedur adalah meningkatkan keterbacaan dan keteraturan kode, sehingga lebih mudah dikelola dan diperbaiki. Dengan memecah program menjadi bagian-bagian kecil, prosedur juga membantu dalam mengurangi pengulangan kode (code redundancy), meningkatkan efisiensi, serta mempermudah debugging dan pengembangan program. Hal ini membuat program lebih modular dan fleksibel untuk diperbarui di masa mendatang.
9. **Pengertian Rekursif dan Fungsinya**
   Rekursi adalah teknik dalam pemrograman di mana sebuah fungsi memanggil dirinya sendiri untuk menyelesaikan masalah yang lebih kecil dari masalah utama. Pendekatan ini sering digunakan untuk menyelesaikan permasalahan yang dapat dipecah menjadi submasalah serupa, seperti pencarian dalam struktur data pohon atau perhitungan faktorial. Agar tidak berjalan tanpa henti, rekursi memerlukan kondisi dasar yang menentukan kapan fungsi harus berhenti memanggil dirinya sendiri. Fungsi rekursif memiliki keunggulan dalam menyederhanakan kode untuk masalah yang memiliki pola berulang, seperti algoritma Divide and Conquer atau pemrosesan data yang bersifat hierarkis. Namun, jika tidak dirancang dengan baik, rekursi dapat menyebabkan penggunaan memori yang berlebihan atau bahkan error karena stack overflow. Oleh karena itu, memahami kapan dan bagaimana menggunakan rekursi dengan efisien sangat penting dalam pemrograman.
10. **Pengertian Struck dan Array**
    Struck adalah kumpulan yang terdiri dari elemen-elemen dengan tipe data yang heterogen atau tidak sama. Struktur dapat dideklarasikan menggunakan kata kunci 'struct', dan menggunakan '.' (operator titik) untuk mengakses elemen-elemen tersebut. Array mengacu pada koleksi yang terdiri dari elemen-elemen yang homogen, yaitu bertipe data yang sama. Array dideklarasikan menggunakan '[]'. Array menggunakan subskrip/'[ ]' (kurung siku) untuk mengakses elemen-elemennya. Pada dasarnya, array adalah pointer yang menunjuk ke elemen pertama dari sebuah koleksi.
11. **Pencarian Nilai Max dan Min**
    Pencarian adalah proses umum dalam kehidupan sehari-hari maupun dalam pemrograman, seperti mencari nilai maksimum atau minimum dalam array. Algoritma pencarian ini dimulai dengan menjadikan elemen pertama sebagai nilai awal, lalu dibandingkan satu per satu dengan elemen lainnya dalam array. Jika ditemukan nilai yang lebih ekstrim, maka nilai tersebut akan diperbarui. Setelah semua elemen diperiksa, nilai maksimum atau minimum yang valid akan diperoleh sebagai hasil akhir.

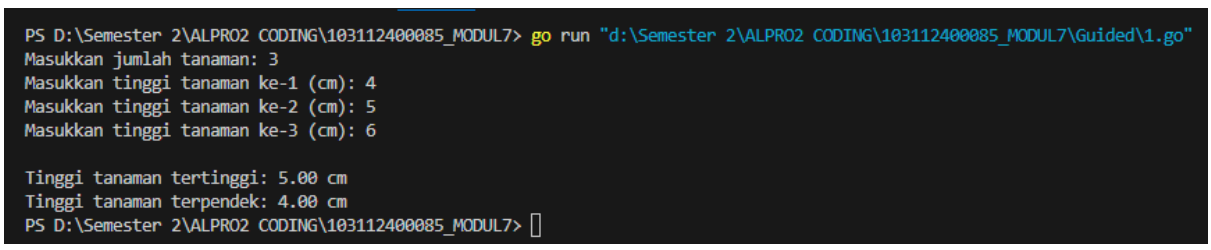
II. GUIDED

1. Guide 1

Source Code:

```
//Anastasia Adinda Narendra Indrianto
//103112400085
package main

import (
    "fmt"
)

func main() {
    var n int
    fmt.Print("Masukkan jumlah tanaman: ")
    fmt.Scan(&n)

    var tinggiTanaman [500]float64
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan tinggi tanaman ke-%d (cm): ", i+1)
        fmt.Scan(&tinggiTanaman[i])
    }

    min, max := tinggiTanaman[0], tinggiTanaman[0]
    for i := 1; i < n; i++ {
        if tinggiTanaman[1] < min {
            min = tinggiTanaman[i]
        }
        if tinggiTanaman[1] > max {
            max = tinggiTanaman[i]
        }
    }

    fmt.Printf("\nTinggi tanaman tertinggi: %.2f cm\n", max)
    fmt.Printf("Tinggi tanaman terpendek: %.2f cm\n", min)
}
```

Output:



Deskripsi Program:

Program ini meminta pengguna untuk memasukkan jumlah tanaman dan tinggi masing-masing tanaman, kemudian menghitung dan menampilkan tinggi tanaman tertinggi dan terpendek. Setelah menerima input tinggi tanaman, program memproses data untuk mencari nilai maksimum dan minimum dari seluruh tinggi tanaman yang dimasukkan, dan menampilkan hasilnya dalam satuan sentimeter.

2. Guide 2

Source Code:

```
//Anastasia Adinda Narendra Indrianto
//103112400085
package main

import (
    "fmt"
)

func main() {
    var x, y int
    fmt.Print("Masukkan jumlah buku dan jumlah buku per rak: ")
    fmt.Scan(&x, &y)

    var hargaBuku [500]float64
    fmt.Println("\nMasukkan harga setiap buku (dalam ribuan Rp): ")
    for i := 0; i < x; i++ {
        fmt.Scan(&hargaBuku[i])
    }

    var hargaRataRata []float64
    for i := 0; i < x; i += y {
        total := 0.0
        for j := i; j < i+y && j < x; j++ {
            total += hargaBuku[j]
        }
        hargaRataRata = append(hargaRataRata, total/float64(y))
    }

    min, max := hargaBuku[0], hargaBuku[0]
    for _, harga := range hargaBuku[:x] {
        if harga < min {
            min = harga
        }
        if harga > max {
```

```
            max = harga
        }
    }

    fmt.Printf("\nRata-rata harga per rak: ")
    for _, avg := range hargaRataRata {
        fmt.Printf("%.2f ", avg)
    }
    fmt.Printf("\nHarga termahal: %.2f Rp\n", max)
    fmt.Printf("Harga termurah: %.2f Rp\n", min)
}
```

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL7\Guided\2.go"
Masukkan jumlah buku dan jumlah buku per rak: 5 10

Masukkan harga setiap buku (dalam ribuan Rp):
10000
20000
30000
40000
50000

Rata-rata harga per rak: 15000.00
Harga termahal: 50000.00 Rp
Harga termurah: 10000.00 Rp
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7>
```

Deskripsi Program:

Program ini menghitung rata-rata harga buku per rak dan mencari harga buku termahal serta termurah berdasarkan input yang diberikan pengguna. Pengguna diminta untuk memasukkan jumlah buku dan jumlah buku per rak, kemudian harga setiap buku dimasukkan satu per satu. Program kemudian menghitung rata-rata harga untuk setiap rak, serta menemukan harga buku yang paling mahal dan paling murah, dan menampilkan hasilnya.

3. Guide 3

Source Code:

```
// Anastasia Adinda Narendra Indrianto
// 103112400085
package main

import (
    "fmt"
)

func main() {
    var n int
    fmt.Print("Masukkan jumlah siswa: ")
    fmt.Scan(&n)
```

```
    var nilaiSiswa [200]float64
    var totalNilai float64 = 0

    fmt.Println("\nMasukkan nilai ujian masing-masing siswa: ")
    for i := 0; i < n; i++ {
        fmt.Scan(&nilaiSiswa[i])
        totalNilai += nilaiSiswa[i]
    }

    rataRata := totalNilai / float64(n)

    min, max := nilaiSiswa[0], nilaiSiswa[0]
    var diAtasRataRata int = 0
    for _, nilai := range nilaiSiswa[:n] {
        if nilai < min {
            min = nilai
        }
        if nilai > max {
            max = nilai
        }
        if nilai > rataRata {
            diAtasRataRata++
        }
    }
    fmt.Printf("\nNilai terendah: %.0f\n", min)
    fmt.Printf("Nilai tertinggi: %.0f\n", max)
    fmt.Printf("Rata-rata kelas: %.2f\n", rataRata)
    fmt.Printf("Jumlah siswa di atas rata-rata: %d\n", diAtasRataRata)

}
```

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL7\Guided\3.go"
Masukkan jumlah siswa: 5

Masukkan nilai ujian masing-masing siswa:
70
80
90
100
60

Nilai terendah: 60
Nilai tertinggi: 100
Rata-rata kelas: 80.00
Jumlah siswa di atas rata-rata: 2
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7>
```

Deskripsi Program:

Program ini digunakan untuk mengolah data nilai ujian siswa dengan meminta pengguna memasukkan jumlah siswa dan nilai ujian masing-masing. Program akan menghitung total nilai, mencari nilai tertinggi dan terendah, menghitung rata-rata kelas, serta menentukan jumlah siswa yang memiliki nilai di atas rata-rata. Hasil perhitungan tersebut kemudian ditampilkan ke layar sebagai ringkasan statistik nilai kelas.

III. UNGUIDED

1. Unguided 1

Source Code:

```
//Anastasia Adinda Narendra Indrianto
//103112400085
package main

import (
    "fmt"
)

func main() {
    var anastasiaN int
    fmt.Print("Masukkan jumlah anak kelinci: ")
    fmt.Scan(&anastasiaN)

    if anastasiaN <= 0 || anastasiaN > 1000 {
        fmt.Println("Jumlah anak kelinci harus antara 1 hingga 1000.")
        return
    }

    var berat [1000]float64

    fmt.Println("Masukkan berat anak kelinci satu per satu:")
    for i := 0; i < anastasiaN; i++ {
        fmt.Scan(&berat[i])
    }

    min := berat[0]
    max := berat[0]

    for i := 1; i < anastasiaN; i++ {
        if berat[i] < min {
            min = berat[i]
        }
        if berat[i] > max {
            max = berat[i]
        }
```

```
    }

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

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL7\Unguided\1.go"
Masukkan jumlah anak kelinci: 5
Masukkan berat anak kelinci satu per satu:
4
5
6
7
8
Berat terkecil: 4.00
Berat terbesar: 8.00
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7>
```

Deskripsi Program:

Program ini berfungsi untuk menentukan berat terkecil dan terbesar dari sejumlah anak kelinci berdasarkan input pengguna. Pengguna diminta memasukkan jumlah anak kelinci (dengan batas antara 1 hingga 1000), kemudian berat masing-masing kelinci diinputkan satu per satu. Program akan memproses data tersebut dan menampilkan hasil berupa berat minimum dan maksimum dari seluruh anak kelinci yang dimasukkan.

2. Unguided 2

Source Code:

```
// Anastasia Adinda Narendra Indrianto
//103112400085
package main

import (
    "fmt"
)

func main() {
    var anastasiaX, y int
    fmt.Print("Masukkan jumlah ikan (anastasiaX) dan kapasitas wadah (y): ")
    fmt.Scan(&anastasiaX, &y)

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

    var ikan [1000]float64
```

```
    fmt.Println("Masukkan berat ikan satu per-satu:")
    for i := 0; i < anastasiaX; i++ {
        fmt.Scan(&ikan[i])
    }

    jumlahWadah := (anastasiaX + y - 1) / y // pembulatan ke atas untuk wadah terakhir
    var totalWadah [1000]float64
    var totalIkanInWadah [1000]int // untuk menghitung jumlah ikan dalam setiap
wadah

    for i := 0; i < anastasiaX; i++ {
        idx := i / y
        totalWadah[idx] += ikan[i]
        totalIkanInWadah[idx]++
    }

    var totalBerat float64
    for i := 0; i < jumlahWadah; i++ {
        if totalIkanInWadah[i] > 0 {
            // Menghitung berat rata-rata ikan dalam wadah
            rataRataIkan := totalWadah[i] / float64(totalIkanInWadah[i])
            fmt.Printf("Berat rata-rata ikan dalam wadah %d: %.2f\n", i+1, rataRataIkan)
        } else {
            fmt.Println("Tidak ada ikan dalam wadah ini.")
        }
        totalBerat += totalWadah[i]
    }

    fmt.Printf("\nTotal berat semua ikan: %.2f\n", totalBerat)
}
```

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL7\Unguided\2.go"
Masukkan jumlah ikan (anastasiaX) dan kapasitas wadah (y): 5 5
Masukkan berat ikan satu per-satu:
1
1
2
2
2
Berat rata-rata ikan dalam wadah 1: 1.60

Total berat semua ikan: 8.00
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7>
```

Deskripsi Program:

Program ini menghitung berat rata-rata ikan dalam beberapa wadah berdasarkan jumlah ikan dan kapasitas wadah yang dimasukkan oleh pengguna. Setelah menerima input jumlah ikan dan kapasitas wadah, program meminta pengguna untuk memasukkan berat ikan satu per satu. Berdasarkan kapasitas wadah, ikan kemudian dikelompokkan dalam wadah dan dihitung total berat serta jumlah ikan dalam setiap wadah. Program menghitung dan menampilkan berat rata-rata ikan dalam setiap wadah, serta total berat semua ikan secara keseluruhan

3. **Unguided 3**

Source Code:

```
//Anastasia Adinda Narendra Indrianto
//103112400085
package main

import (
    "fmt"
)

type arrBalita [100]float64

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

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

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

func main() {
    var jumlahData int
    var berat arrBalita

    fmt.Print("Masukan banyak data berat balita: ")
    fmt.Scan(&jumlahData)

    if jumlahData > 100 {
        fmt.Println("Jumlah data tidak boleh lebih dari 100.")
        return
    }

    for i := 0; i < jumlahData; i++ {
        fmt.Printf("Masukan berat balita ke-%d: ", i+1)
```

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

    var bMin, bMax float64
    hitungMinMax(berat, jumlahData, &bMin, &bMax)

    fmt.Printf("Berat balita minimum: %.2f kg\n", bMin)
    fmt.Printf("Berat balita maksimum: %.2f kg\n", bMax)

    rata := rerata(berat, jumlahData)
    fmt.Printf("Rerata berat balita: %.2f kg\n", rata)
}
```

Output:

```
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7> go run "d:\Semester 2\ALPRO2 CODING\103112400085_MODUL7\Unguided\3.go"
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
PS D:\Semester 2\ALPRO2 CODING\103112400085_MODUL7>
```

Deskripsi Program:

Program di atas berfungsi untuk mencatat dan menghitung data berat balita yang dimasukkan oleh pengguna. Pengguna diminta untuk memasukkan jumlah data berat balita, kemudian berat setiap balita dimasukkan satu per satu. Program ini kemudian menghitung berat balita terkecil (minimum), terbesar (maksimum), serta menghitung rata-rata berat balita yang ada. Fungsi hitungMinMax digunakan untuk menentukan nilai berat terkecil dan terbesar, sementara fungsi rerata digunakan untuk menghitung rata-rata berat balita berdasarkan jumlah data yang dimasukkan. Program ini membatasi jumlah data yang dimasukkan hingga 100 balita dan menampilkan hasil dengan format yang jelas.

IV. KESIMPULAN

Kesimpulan Program Guided

1. Program Pengolahan Data Tinggi Tanaman
   Program ini meminta input jumlah tanaman dan tinggi masing-masing tanaman, lalu menghitung dan menampilkan tinggi tanaman tertinggi dan terpendek dalam satuan sentimeter. Program memanfaatkan logika perbandingan sederhana untuk mencari nilai maksimum dan minimum.

2. Program Pengolahan Harga Buku per Rak
   Program menerima input jumlah buku dan jumlah buku per rak, kemudian meminta harga setiap buku satu per satu. Program menghitung rata-rata harga setiap rak serta menentukan harga buku tertinggi dan terendah, sehingga hasilnya dapat digunakan untuk menganalisis distribusi harga dalam rak buku.

3. Program Statistik Nilai Siswa
   Program ini digunakan untuk mengelola data nilai ujian siswa. Program menghitung total nilai, rata-rata, nilai tertinggi, nilai terendah, serta menghitung jumlah siswa yang memiliki nilai di atas rata-rata. Hasil yang ditampilkan memberikan ringkasan statistik nilai kelas secara menyeluruh.

Kesimpulan Program Unguided

1. Program Berat Anak Kelinci
   Program ini meminta input jumlah anak kelinci dan berat masing-masing kelinci, lalu menghitung dan menampilkan berat terkecil dan terbesar. Program ini menerapkan algoritma pencarian nilai ekstrem secara berurutan dari data input.

2. Program Rata-rata Berat Ikan per Wadah
   Program menerima jumlah ikan dan kapasitas wadah, lalu meminta input berat tiap ikan. Berdasarkan kapasitas, ikan dikelompokkan ke dalam beberapa wadah, dan program menghitung rata-rata berat ikan per wadah serta total berat semua ikan secara keseluruhan.

3. Program Statistik Berat Balita
   Program ini mencatat data berat balita, kemudian menghitung nilai berat minimum, maksimum, dan rata-rata. Fungsi-fungsi terpisah digunakan untuk menghitung nilai-nilai tersebut, membuat program lebih terstruktur dan modular. Program membatasi input hingga 100 data dan menyajikan hasil akhir secara jelas.

Seluruh program baik guided maupun unguided menunjukkan bahwa teknik dasar pemrograman seperti penggunaan array, perulangan, logika pencarian nilai ekstrem, dan perhitungan rata-rata sangat berguna dalam mengolah data numerik. Struktur kode yang rapi dan logika yang terorganisir memungkinkan hasil yang efisien, akurat, dan mudah dipahami. Hal ini memperlihatkan bahwa penguasaan dasar-dasar logika pemrograman sangat penting untuk menyelesaikan permasalahan nyata dalam berbagai konteks secara sistematis.

V. REFRENSI

https://sko.dev/wiki/input-dan-output

https://codingstudio.id/blog/golang-adalah/

https://dif.telkomuniversity.ac.id/tipe-data-pemrograman/

https://rpubs.com/maulidyarahmah/829044

https://repository.unikom.ac.id/62967/1/Materi%20Pertemuan%204_Labview%201%2BWhile%20Loop%20%2B%20Shift%20Register.pdf

https://www.revou.co/kosakata/function

https://sko.dev/wiki/fungsi

https://drive.google.com/file/d/1tMnxOLAYoMqLB9cKyeJHwyFjmyVtKqMV/view?usp=sharing

https://janabadra.ac.id/2023/algoritma-rekursi/#:~:text=Rekursi%20adalah%20sebuah%20metode%20pengulangan,seperti%20definisi%20fungsi%20pada%20umumnya.

https://www.tutorialspoint.com/difference-between-array-and-structure

file:///C:/Users/Laptopku/Downloads/Modul%207%20-%20Praktikum%20Alpro%202.pdf