LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2

MODUL 7

PENCARIAN NILAI EKSTRIM PADA HIMPUNAN DATA

DISUSUN OLEH:

Keishin Naufa Alfaridzhi

103112400061

S1 IF-12-01

S1 TEKNIK INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

I. DASAR TEORI

A. Bahasa Yang Digunakan

Pada praktikum ini bahasa pemrograman yang digunakan adalah bahasa pemrograman Go, sesuai dengan modul yang menjadi acuan praktikum. Golang (atau) Go adalah bahasa pemrograman baru, yang mulai dilirik oleh para developer karena kelebihan-kelebihan yang dimilikinya. Sudah banyak Perusahaan besar yang menggunakan bahasa ini untuk produk-produk mereka hingga di level production.

B. Komentar

Komentar biasa dimanfaatkan untuk menyisipkan catatan pada kode program, menulis penjelasan atau deskripsi mengenai suatu blok kode, atau bisa juga digunakan untuk me-remark kode (men-non-aktifkan kode yang tidak digunakan). Komentar akan diabaikan Ketika kompilasi maupun eksekusi program.

Ada 2 jenis komentar di Golang, yaitu inline dan multiline.

1. Komentar Inline

Penulisan komentar jenis ini diawali dengan tanda *double slash* (//) lalu diikuti pesan komentarnya. Komentar inline hanya berlaku untuk satu baris pesan saja. Jika pesan komentar lebih dari satu baris, maka tanda *double slash* harus ditulis lagi di baris selanjutnya.

2. Komentar Multiline

Komentar yang cukup panjang akan lebih rapi jika ditulis menggunakan teknik komentar multiline. Ciri dari komentar jenis ini adalah penulisannya diawali dengan tanda (/*) dan diakhiri (*/).

C. Variabel

Golang mengadopsi 2 jenis penulisan variabel, yang dituliskan tipe data-nya dan yang tidak. Kedua cara tersebut intinya adalah sama, pembedanya hanyalah cara penulisannya saja. Untuk penulisan variabel dengan tipe data, keyword *var* digunakan untuk deklarasi variabel kemudian diakhiri dengan tipe data misalnya *string*. Kemudian untuk penulisan variabel tanpa tipe data, variabel dideklarasikan dengan menggunakan metode type inference. Penandanya tipe data tidak dituliskan pada saat deklarasi. Pada penggunaan metode ini, operand (=) harus diganti dengan (:=) dan keyword *var* dihilangkan.

Golang memiliki aturan unik yang tidak dimiliki bahasa lain, yaitu tidak boleh ada satupun variabel yang menganggur. Artinya, semua variabel yang dideklarasikan harus digunakan. Jika terdapat variabel yang tidak digunakan tapi dideklarasikan, program akan gagal dikompilasi. Untuk mengatasi itu, golang memiliki variabel yaitu underscore. Underscore (_) adalah predefined variabel yang bisa dimanfaatkan untuk menampung nilai yang tidak dipakai.

D. Tipe Data

Golang mengenal beberapa jenis tipe data, diantaranya adalah tipe data numerik (decimal dan non-desimal), string, dan boolean.

1. Tipe Data Numerik Non-Desimal (uint, int)
2. Tipe Data Numerik Desimal (float64, float32)
3. Tipe Data Bool (true, false)
4. Tipe Data String (string, " ")

E. Operator Aritmatika

Operator aritmatika merupakan operator yang digunakan untuk operasi yang sifatnya perhitungan. Golang mendukung beberapa operator aritmatika standar, yaitu:

1. Penjumlahan (+)
2. Pengurangan (-)
3. Perkalian (*)
4. Pembagian (/)
5. Modulus atau sisa hasil pembagian (%)

F. Seleksi Kondisi

Seleksi kondisi pada program berguna untuk mengontrol sebuah blok kode yang akan dieksekusi. Yang dijadikan acuan oleh selksi kondisi adalah nilai bertipe bool, bisa berasal dari variabel, ataupun hasil operasi perbandingan. Nilai tersebut menentukan blok kode mana yang akan dieksekusi. Go memiliki 2 macam keyword untuk selesksi kondisi, yaitu if else dan switch.

1. **If Expression**

   If adalah salah satu kata kunci yang digunakan dalam percabangan. Percabangan artinya kitabisa mengeksekusi kode program tertentu ketika suatu kondisi terpenuhi. Hampir semua bahasa pemrograman mendukung if expression.

2. **Else if expression**

   Terkadang kita butuh membuat beberapa kondisi. Kasus seperti ini dapat menggunakan else if expression. If mendukung short statement sebelum kondisi.

Hal ini sangat cocok untuk membuat statement yang sederhana sebelum melakukan pengecekan terhadap kondisi.

3. **Switch-Case**

   Switch merupakan seleksi kondisi yang sifatnya fokus pada satu variabel, lalu kemudian di-cek nilainya. Contoh sederhananya seperti penentuan apakah nilai variabel x adalah: 1, 2, 3, atau lainnya. Perlu diketahui, switch pada pemrograman Go memiliki perbedaan dibanding bahasa lain. Di Go, ketika sebuah case terpenuhi, tidak akan dilanjutkan ke pengecekan case selanjutnya, meskipun tidak ada keyword "break" di situ. Konsep ini berkebalikan dengan switch pada umumnya pemrograman lain (yang ketika sebuah case terpenuhi, maka akan tetap dilanjut mengecek case selanjutnya kecuali ada keyword "break").

G. Perulangan

Perulangan merupakan proses mengulang dan mengeksekusi blok kode tanpa henti sesuai dengan kondisi yang dijadikan acuan. Biasanya disiapkan variabel untuk iterasi atau penanda kapan perulangan akan dihentikan.

a. For Loop

For loop merupakan statement perulangan dasar dan cukup sering ditemui. Format for loop yaitu sebagai berikut.

- *Init Statement*: bagian ini akan dieksekusi sebelum perulangan dimulai. Biasanya diisi dengan mendeklarasi variabel iterasi.
- *Condition Expression*: bagian ini akan dicek dan dieksekusi setiap perulangan yang dilakukan, jika true maka perulangan akan terus berjalan hingga kondisi bernilai false.
- *Post Statement*: statement ini akan dieksekusi pada akhir iterasi. Jika terdapat range, maka perulangan akan dieksekusi untuk setiap item pada range.

b. While Loop

While loop merupakan perulangan yang akan terus berjalan hingga suatu kondisi terpenuhi. Penulisan while loop adalah dengan menuliskan kondisi setelah keyword for (hanya kondisi). Deklarasi dan iterasi variabel counter tidak dituliskan setelah keyword, hanya kondisi perulangan saja. Konsepnya mirip seperti while milik bahasa pemrograman lain.

c. Repeat Until

Untuk Repeat Until ini mirip seperti for loop biasa namun hanya menggunakan inisiasi dan kondisi saja.

H. Fungsi

Dalam konteks pemrograman, fungsi adalah sekumpulan blok kode yang dibungkus dengan nama tertentu. Penerapan fungsi yang tepat akan menjadikan kode lebih modular dan juga *dry* (singkatan dari *don't repeat yourself*) yang artinya kita tidak perlu menuliskan banyak kode untuk kegunaan yang sama berulang kali. Cukup deklarasikan sekali saja blok kode sebagai suatu fungsi, lalu panggil sesuai kebutuhan.

1. Penerapan Fungsi

Sebenarnya kita sudah mengimplementasikan fungsi pada banyak praktek sebelumnya, yaitu fungsi `main()`. Fungsi `main()` sendiri merupakan fungsi utama pada program Go, yang akan dieksekusi ketika program dijalankan.

Selain fungsi `main()`, kita juga bisa membuat fungsi lainnya. Dan caranya cukup mudah, yaitu dengan menuliskan keyword `func` kemudian diikuti nama fungsi, lalu kurung `()` (yang bisa diisi parameter), dan diakhiri dengan kurung kurawal untuk membungkus blok kode.

Parameter merupakan variabel yang menempel di fungsi yang nilainya ditentukan saat pemanggilan fungsi tersebut. Parameter sifatnya opsional, suatu fungsi bisa tidak memiliki parameter, atau bisa saja memeliki satu atau banyak parameter (tergantung kebutuhan).

2. Fungsi dengan Nilai Balik / Return Value

Selain parameter, fungsi bisa memiliki attribute **return value** atau nilai balik. Fungsi yang memiliki return value, saat deklarasinya harus ditentukan terlebih dahulu tipe data dari nilai baliknya.

3. Fungsi tanpa Nilai Balik / Return Value

Fungsi juga dapat tidak memiliki nilai balik yang dapat disebut juga sebagai Prosedur. Prosedur dapat dianggap sebagai potongan beberapa instruksi program menjadi suatu instruksi baru yang dibuat untuk mengurangi kerumitan dari kode program yang kompleks pada suatu program yang besar.

I. Array

Array adalah kumpulan data bertipe sama, yang disimpan dalam sebuah variabel. Array memiliki kapasitas yang nilainya ditentukan pada saat pembuatan, menjadikan elemen/data yang disimpan di array tersebut jumlahnya tidak boleh melebihi yang sudah dialokasikan.

Default nilai tiap elemen array pada awalnya tergantung dari tipe datanya. Jika `int` maka tiap element zero value-nya adalah `0`, jika `bool` maka `false`, dan seterusnya. Setiap elemen array memiliki indeks berupa angka yang merepresentasikan posisi urutan elemen tersebut. Indeks array dimulai dari 0.

II. GUIDED

1. **Source Code:**

```
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[i] < min {
            min = tinggiTanaman[i]
        }
        if tinggiTanaman[i] > max {
            max = tinggiTanaman[i]
        }
    }

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

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul10\103112400061_
Masukkan jumlah tanaman: 5
Masukkan tinggi tanaman ke-1 (cm): 12
Masukkan tinggi tanaman ke-2 (cm): 20.3
Masukkan tinggi tanaman ke-3 (cm): 30.78
Masukkan tinggi tanaman ke-4 (cm): 32
Masukkan tinggi tanaman ke-5 (cm): 25

Tinggi tanaman tertinggi: 32.00 cm
Tinggi tanaman terpendek: 12.00 cm
```

Penjelasan:

Program untuk mencari nilai min dan max dari set data tanaman. Menentukan jumlah tanaman yang akan diinput melalui variabel n lalu menginputkan tinggi masing-masing tanaman dari index 0 hingga index n. Kemudian dicari nilai min dan max nya

jika index i < min maka min = tanaman index i. Jika index i > max maka max = tanaman index i.

2. **Source Code:**

```
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:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul7\103112400061_
Masukkan jumlah buku dan jumlah buku per rak: 8 3

Masukkan harga setiap buku (dalam ribuan Rp):
2000 3000 4000 2500 3200 5000 8000 4300

Rata-rata harga per rak: 3000.00 3566.67 4100.00
Harga termahal: 8000.00 Rp
Harga termurah: 2000.00 Rp
```

Penjelasan:

Program mencari harga termurah, termahal, dan rata-rata harga buku tiap rak-nya. Pertama masukkan nilai x dan y. x untuk jumlah buku dan y untuk jumlah buku per-rak. Setelah itu masukkan harga setiap buku nya (x). Mencari total dengan menjumlahkan harga buku per-rak untuk setiap rak kemudian dibagi dengan jumlah buku per-rak. Kemudian dicari nilai min dan max nya jika index i < min maka min = harga index i. Jika index i > max maka max = harga index i.

3. **Source Code:**

```
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 siswa terendah: %.0f\n", min)
    fmt.Printf("Nilai siswa tertinggi: %.0f\n", max)
    fmt.Printf("\nRata-rata nilai siswa: %.2f\n", rataRata)
    fmt.Printf("Jumlah siswa di atas rata-rata: %d\n", diAtasRataRata)
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul7\103112400061
Masukkan jumlah siswa: 6

Masukkan nilai ujian masing-masing siswa:
80
90
91
86
72
89

Nilai siswa terendah: 72
Nilai siswa tertinggi: 91

Rata-rata nilai siswa: 84.67
Jumlah siswa di atas rata-rata: 4
```

Penjelasan:

Program mencari nilai siswa terendah, tertinggi, rata-rata nilai dan nilai di atas rata-rata. Menentukan jumlah siswa yang akan dihitung nilainya dengan menginputkan variabel n. Kemudian inputkan nilai n siswa. Dicari nilai min dan max nya jika index i < min maka min = nilai index i. Jika index i > max maka max = nilai index i. Lalu dicari nilai rata-ratanya dengan cara total nilai siswa dibagi dengan jumlah siswa. Untuk cek nilai di atas rata-rata menggunakan kondisi nilai > rataRata.

III. UNGUIDED

1. Latihan no. 1

Source Code:

```
// KEISHIN NAUFA ALFARIDZHI
// 103112400061
package main
import "fmt"
```

```
func main() {
    var (
        n int
    )

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

    var kandangKelinci [1000]float64
    for i := 0; i < n; i++ {
        fmt.Printf("Masukkan berat kelinci ke-%d (kg): ",i+1)
        fmt.Scan(&kandangKelinci[i])
    }

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

    fmt.Printf("\nBerat anak kelinci terberat: %.2f kg\n", max)
    fmt.Printf("Berat anak kelinci teringan: %.2f kg\n", min)
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul7\103112400061
Masukkan jumlah anak kelinci: 5
Masukkan berat kelinci ke-1 (kg): 2.7
Masukkan berat kelinci ke-2 (kg): 1.8
Masukkan berat kelinci ke-3 (kg): 1.89
Masukkan berat kelinci ke-4 (kg): 2.3
Masukkan berat kelinci ke-5 (kg): 3

Berat anak kelinci terberat: 3.00 kg
Berat anak kelinci teringan: 1.80 kg
```

Deskripsi Program:

Menentukan anak kelenci terberat dan teringan. Menentukan jumlah kelinci yang akan dihitung beratnya dengan menginputkan variabel n. Kemudian input berat tiap kelinci. Setelah itu dicari nilai min dan max nya jika index i < min maka min = berat index i. Jika index i > max maka max = berat index i.

2. Latihan no. 2

Source Code:

```
// KEISHIN NAUFA ALFARIDZHI
// 103112400061
package main
import "fmt"

func main() {
    var (
        x, y int
        totalWadah int
        totalBeratIwak, avg float64
    )
    fmt.Print("Masukkan jumlah ikan yang akan dijual dan ikan yang dimasukkan ke
dalam wadah: ")
    fmt.Scan(&x, &y)

    var iwak [1000]float64
    fmt.Println("\nMasukkan berat setiap ikan (kg):")
    for i := 0; i < x; i++ { // 103112400061
        fmt.Scan(&iwak[i])
    }

    totalWadah = (x+y-1) / y
    totalBeratTiapWadah := make([]float64, totalWadah)

    for nope := 0; nope < x; nope++ {
        iWadah := nope / y
        totalBeratTiapWadah[iWadah] += iwak[nope]
    }

    fmt.Print("\nTotal berat ikan tiap wadah: ")
    for _, wadah := range totalBeratTiapWadah {
        totalBeratIwak += wadah
        fmt.Printf("%.2f ", wadah)
    }
    if len(totalBeratTiapWadah) < y {
        fmt.Print(y-y)
    }
    avg = totalBeratIwak / float64(totalWadah)
    fmt.Printf("\nRata-rata berat ikan tiap wadah: %.2f\n", avg)
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul7\103112400061
Masukkan jumlah ikan yang akan dijual dan ikan yang dimasukkan ke dalam wadah: 8 3

Masukkan berat setiap ikan (kg):
1.2 2.64 2 3.12 0.9 1.46 1.99 1.4

Total berat ikan tiap wadah: 5.84 5.48 3.39
Rata-rata berat ikan tiap wadah: 4.90
```

Deskripsi Program:

Program untuk mencari total berat ikan tiap wadah dan mencari rata-rata berat ikan tiap wadah. Masukkan nilai x dan y. x untuk jumlah ikan dan y untuk jumlah ikan per-wadah. Lalu masukkan berat tiap ikan yang berjumlah x. Membuat array bernama totalBeratTiapWadah untuk menyimpan nilai berat setiap wadah. Kemudian menjumlahkan berat ikan untuk tiap wadah dengan for loop, terdapat variabel iWadah untuk menentukan index wadah yang akan ditambahkan ikan yang diketahui nomor indexnya dengan membagi iterasi perulangan dengan y, kemudiann menambahkan berat ikan ke dalam index wadah. Untuk mencari rata-rata berat tiap wadah, total berat ikan akan dibagi oleh jumlah wadah.

3. Latihan no. 3

Source Code:

```
// KEISHIN NAUFA ALFARIDZHI
// 103112400061
package main
import "fmt"

type arrBalita [100]float64

func main() {
    var (
        n int
        arrBalita arrBalita
    )

    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(&arrBalita[i])
    }

    beratMin := arrBalita[0]
```

```
    beratMax := arrBalita[0]

    hitungMinMax(arrBalita, &beratMin, &beratMax)
    rataRata := rerata(arrBalita)

    fmt.Printf("Berat balita minimum: %.2f kg\n", beratMin)
    fmt.Printf("Berat balita maximum: %.2f kg\n", beratMax)
    fmt.Printf("Rerata berat balita: %.2f kg\n", rataRata)
}

func hitungMinMax(arrBerat arrBalita, bMin, bMax *float64) { // 103112400061
    for i := 0; i < len(arrBerat) && arrBerat[i] != 0; i++ {
        if arrBerat[i] != 0 {
            if arrBerat[i] < *bMin {
                *bMin = arrBerat[i]
            }
            if arrBerat[i] > *bMax {
                *bMax = arrBerat[i]
            }
        }
    }
}

func rerata(arrBerat arrBalita) float64 {
    var avg, totalBerat float64
    var n int
    for i := 0; arrBerat[i] != 0; i++ {
        n++
    }
    for _, berat := range arrBerat {
        totalBerat += berat
    }
    avg = totalBerat / float64(n)
    return avg
}
```

Output:

```
D:\Tugas\SEM2\Alpro\Praktikum>go run "d:\Tugas\SEM2\Alpro\Praktikum\modul7\103112400061_
Masukkan banyak data berat balita: 4
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 maximum: 9.90 kg
Rerata berat balita: 6.38 kg
```

Deskripsi Program:

Program untuk mencari min, max dan rata-rata dari berat balita menggunakan function dan prosedur. Mendeklarasikan tipe arrBalita yang merupakan array sebesar 100 dengan tipe data float64.

Prosedur hitungMinMax(), memiliki 3 parameter diantaranya arrBerat dengan tipe data arrBalita dan bMin, bMax yang merupakan pointer dengan tipe data float64. Mencari berat min dan max nya jika index i < min maka min = berat index i. Jika index i > max maka max = berat index i. Untuk menggunakan pointer, perlu menandai variabel yang dipanggil menggunakan '&' sebagai address pointer.

Function rerata(), memiliki 1 parameter arrBerat dengan tipe arrBalita. Mencari rata-rata dengan mencari data dengan berat tidak nol, kemudian total berat akan dibagi dengan total data yang tidak nol.

IV. KESIMPULAN

Pada praktikum ini telah dibahas perihal cara menentukan nilai ekstrim pada array. Mencari nilai ekstrim dari suatu array dapat berguna dalam mendapatkan statistik dari suatu kumpulan data. Misalnya seperti pada praktikum ini, mencari nilai minimum dan maksimum, mencari rerata, dan menghitung rerata dalam kondisi tertentu.

V. DAFTAR PUSTAKA

Noval Agung Prayogo. *Dasar Pemrograman Golang.* Diakses pada 01 Oktober 2024. https://dasarpemrogramangolang.novalagung.com

Annisa Nur Isnaeni. *Golang — Seleksi Kondisi.* Diakses pada 01 Oktober 2024.

https://medium.com/@annisaisna/golang-seleksi-kondisi-f988ead004b4

Parvez Alam, *Golang for loop example | Golang Loops Tutorial – Phpflow.com* https://medium.com/@parvez1487/golang-for-loop-example-golang-loops-tutorial-phpflow-com-f4b2b0e57944