LAPORAN PRAKTIKUM

ALGORITMA DAN PEMROGRAMAN 1

MODUL 16

“SKEMA PEMROSESAN SEKUENSIAL”

DISUSUN OLEH:

RAIHAN ADI ARBA

103112400071

S1 IF-12-01

DOSEN:

Yohani Setiya Rafika Nur, M. Kom.

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024/2025

DASAR TEORI

Pemrosesan sekuensial merupakan suatu metode pengolahan data yang dilakukan secara berurutan dan sistematis, di mana setiap elemen data diproses satu per satu dengan urutan yang telah ditentukan. Dalam konteks ini, setiap elemen memiliki suksesor atau penerus, sehingga membentuk suatu rangkaian yang disebut sebagai "deret" elemen. Elemen-elemen yang diproses dapat berupa tipe data dasar seperti integer, real, character, dan boolean, atau dapat juga berupa tipe data komposisi seperti Point.

Dalam implementasinya, deret elemen dapat bersumber dari berbagai media, antara lain pembacaan langsung dari perangkat input, nilai-nilai yang tersimpan dalam tabel atau matriks, data yang tersimpan dalam media penyimpanan sekunder (file), atau elemen-elemen dalam struktur data list. Untuk mengelola dan mengakses elemen-elemen tersebut, digunakan empat primitif utama: First_Elmt (penunjuk elemen pertama), Current_Elmt (elemen yang sedang diproses), Next_Elmt (elemen berikutnya), dan EOP (End of Process) sebagai penanda akhir proses

.Skema pemrosesan sekuensial ini dikembangkan untuk memberikan pola yang lebih komprehensif dibandingkan dengan instruksi pengulangan biasa. Hal ini karena skema ini tidak hanya berfokus pada mekanisme pengulangan, tetapi juga mencakup berbagai abstraksi yang diperlukan dalam pemrosesan sekuensial. Implementasi primitif-primitif tersebut diwujudkan dalam bentuk prosedur dan fungsi yang spesifik, di mana First_Elmt bertugas menginisiasi proses dengan menyediakan elemen pertama, Next_Elmt mengatur perpindahan ke elemen berikutnya, dan EOP berfungsi sebagai penanda akhir proses.

Dalam penerapannya, EOP dapat diimplementasikan dalam dua model: model dengan MARK dan model tanpa MARK. Pada model dengan MARK, digunakan elemen fiktif sebagai penanda akhir yang berbeda dari elemen-elemen yang diproses secara normal. Sementara itu, pada model tanpa MARK, elemen terakhir sendiri mengandung informasi yang menandakan bahwa elemen tersebut merupakan elemen terakhir dalam rangkaian. Model dengan MARK lebih umum digunakan dalam praktik pemrograman karena kemudahan implementasinya.

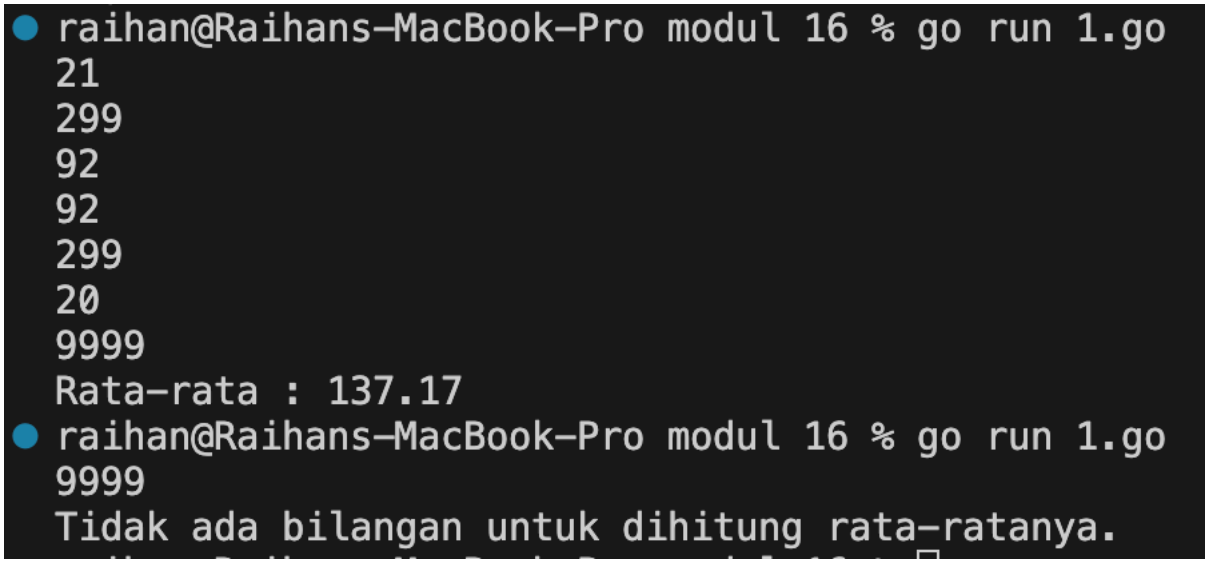
A. UNGUIDED

1. Latihan 1
   Diberikan sejumlah bilangan riil yang diakhiri dengan marker 9999, cari rerata dari bilangan-bilangan tersebut.

   Source Code:

```
 package main

import "fmt"

func main() {
	var number, sum float64
	var count int

	for {
		fmt.Scan(&number)

		if number == 9999 {
			break
		}

		sum += number
		count++
	}

	if count == 0 {
		fmt.Println("Tidak ada bilangan untuk dihitung rata-ratanya.")
	} else {
		average := sum / float64(count)
		fmt.Printf("Rata-rata : %.2f\n", average)
	}
}
```

Output :



Penjelasan Program:

Program ini dirancang untuk menghitung rata-rata dari serangkaian bilangan yang diinput oleh pengguna. Program dimulai dengan mendeklarasikan variabel number dan sum bertipe float64 untuk menyimpan input bilangan dan total penjumlahan, serta variabel count bertipe integer untuk menghitung jumlah bilangan yang diinput. Program menggunakan perulangan tanpa batas yang akan terus meminta input bilangan dari pengguna hingga ditemukan input 9999 sebagai tanda berhenti. Setiap kali bilangan diinput (selain 9999), program akan menambahkan bilangan tersebut ke dalam sum dan menambah count. Setelah perulangan selesai, program melakukan pengecekan: jika count bernilai 0 (tidak ada bilangan yang diinput), program akan menampilkan pesan "Tidak ada bilangan untuk dihitung rata-ratanya." Namun jika ada bilangan yang diinput, program akan menghitung rata-rata dengan membagi sum dengan count dan menampilkan hasilnya dengan format dua angka desimal menggunakan fmt.Printf("Rata-rata : %.2f\n", average).

2. Latihan 2

Diberikan string x dan n buah string. x adalah data pertama yang dibaca, n adalah data bilangan yang dibaca kedua, dan n data berikutnya adalah data string. Buat algoritma untuk menjawab pertanyaan berikut:

a. Apakah string x ada dalam kumpulan n data string tersebut?
b. Pada posisi ke berapa string x tersebut ditemukan?
c. Ada berapakah string x dalam kumpulan n data string tersebut?
d. Adakah sedikitnya dua string x dalam n data string tersebut?

Source Code :

```
package main

import "fmt"
```

```
func main() {
	var x, input string
	var n, count, firstPos int
	var found bool

	fmt.Print("Masukkan string yang dicari: ")
	fmt.Scan(&x)
	fmt.Print("Masukkan jumlah data: ")
	fmt.Scan(&n)

	count = 0
	firstPos = -1
	found = false

	for i := 1; i <= n; i++ {
		fmt.Printf("Masukkan string ke-%d: ", i)
		fmt.Scan(&input)

		if input == x {
			count++
			if !found {
				firstPos = i
				found = true
			}
		}
	}
	fmt.Println("\nHasil:")

	if found {
		fmt.Println("a. Ya, string ditemukan")
	} else {
		fmt.Println("a. Tidak, string tidak ditemukan")
	}

	if firstPos != -1 {
		fmt.Printf("b. Ditemukan pada posisi ke-%d\n", firstPos)
	} else {
		fmt.Println("b. String tidak ditemukan")
	}

	fmt.Printf("c. String muncul sebanyak %d kali\n", count)

	if count >= 2 {
		fmt.Println("d. Ya, terdapat minimal dua string yang sama")
	} else {
		fmt.Println("d. Tidak, tidak terdapat minimal dua string yang sama")
```

```
        }
}
```

Output :

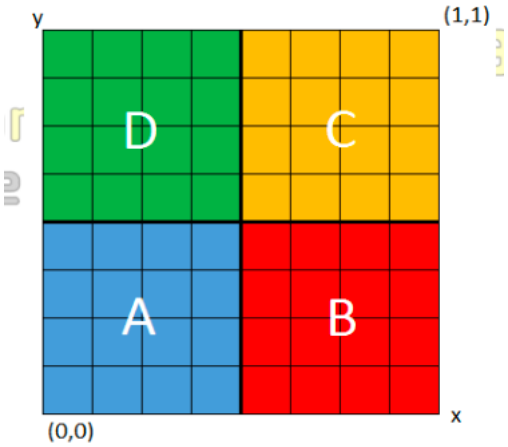
```
raihan@Raihans-MacBook-Pro modul 16 % go run 2.go
Masukkan string yang dicari: 30
Masukkan jumlah data: 10
Masukkan string ke-1: 3
Masukkan string ke-2: 5
Masukkan string ke-3: 2
Masukkan string ke-4: 1
Masukkan string ke-5: 9
Masukkan string ke-6: 30
Masukkan string ke-7: 20
Masukkan string ke-8: 10
Masukkan string ke-9: 29
Masukkan string ke-10: 29

Hasil:
a. Ya, string ditemukan
b. Ditemukan pada posisi ke-6
c. String muncul sebanyak 1 kali
d. Tidak, tidak terdapat minimal dua string yang sama
```

Pembahasan :

Program ini merupakan sistem pencarian string yang memiliki beberapa fitur analisis. Program dimulai dengan mendeklarasikan beberapa variabel: x untuk string yang dicari, input untuk string yang akan dimasukkan, n untuk jumlah data, count untuk menghitung kemunculan string, firstPos untuk posisi pertama string ditemukan, dan found sebagai penanda string ditemukan atau tidak. Program pertama meminta input string yang akan dicari dan jumlah data yang akan diproses. Selanjutnya, program menggunakan perulangan for untuk meminta input string sebanyak n kali. Setiap kali string diinput, program akan membandingkannya dengan string yang dicari (x). Jika string cocok, count akan bertambah, dan jika ini adalah penemuan pertama (found masih false), maka posisi tersebut akan disimpan di firstPos dan found diubah menjadi true. Setelah semua input selesai, program menampilkan empat hasil analisis: (a) apakah string ditemukan, (b) posisi pertama string ditemukan, (c) berapa kali string muncul, dan (d) apakah string muncul minimal dua kali. Output ditampilkan dengan format yang rapi dan informatif sesuai dengan hasil pencarian.

3. Latihan 3

Empat daerah A, B, C, dan D yang berdekatan ingin mengukur curah hujan. Keempat daerah tersebut digambarkan pada bidang berikut:



Misal curah hujan dihitung berdasarkan banyaknya tetesan air hujan. Setiap tetesan berukuran 0.0001 ml curah hujan. Tetesan air hujan turun secara acak dari titik (0,0) sampai (1,1). Jika diterima input yang menyatakan banyaknya tetesan air hujan.
Tentukan curah hujan untuk keempat daerah tersebut.
Buatlah program yang menerima input berupa banyaknya tetesan air hujan. Kemudian buat koordinat/titik (x, y) secara acak dengan menggunakan fungsi rand.Float64().
Hitung dan tampilkan banyaknya tetesan yang jatuh pada daerah A, B, C dan D. Konversikan satu tetesan berukuran 0.0001 milimeter.
Catatan: Lihat lampiran untuk informasi menggunakan paket math/rand untuk menggunakan rand.Float64() yang menghasilkan bilangan riil acak [0..1].
Berikut contoh masukan dan keluarannya:

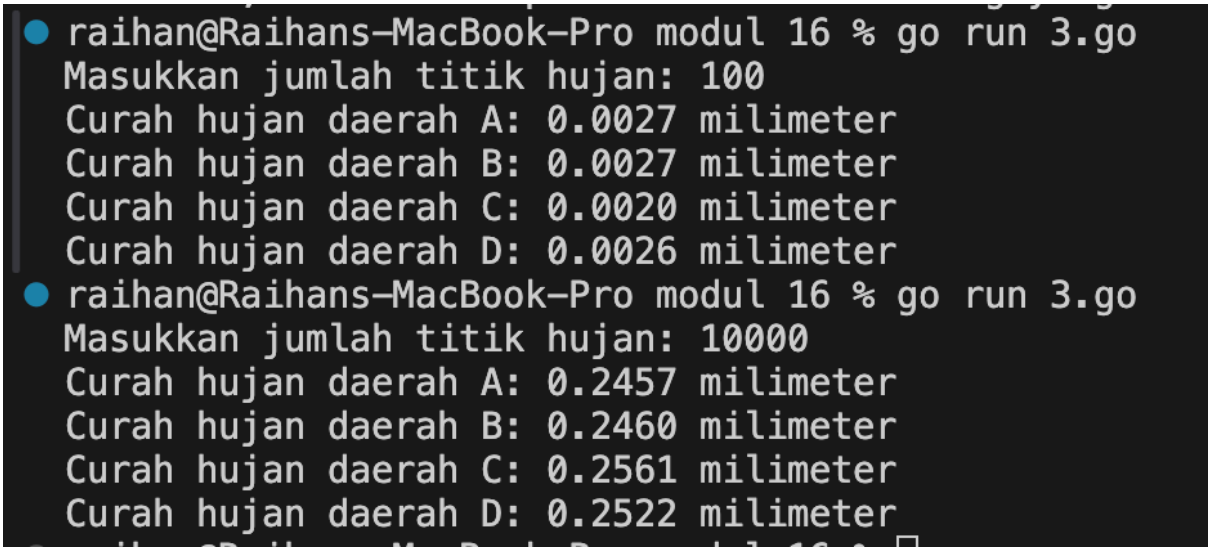
No	Masukan	Keluaran
1	10000000	Curah hujan daerah A: 250.0066 milimeter Curah hujan daerah B: 249.8981 milimeter Curah hujan daerah C: 249.9930 milimeter Curah hujan daerah D: 250.1023 milimeter

Source Code:

```
package main

import (
	"fmt"
	"math/rand"
)

func main() {
```

```
	var daerahA, daerahB, daerahC, daerahD int
	var totalTitik int

	fmt.Print("Masukkan jumlah titik hujan: ")
	fmt.Scan(&totalTitik)

	for i := 0; i < totalTitik; i++ {
		x := rand.Float64()
		y := rand.Float64()

		// Cek posisi titik
		if x < 0.5 && y < 0.5 {
			daerahA++
		} else if x >= 0.5 && y < 0.5 {
			daerahB++
		} else if x >= 0.5 && y >= 0.5 {
			daerahC++
		} else {
			daerahD++
		}
	}

	curahA := float64(daerahA) * 0.0001
	curahB := float64(daerahB) * 0.0001
	curahC := float64(daerahC) * 0.0001
	curahD := float64(daerahD) * 0.0001

	fmt.Printf("Curah hujan daerah A: %.4f milimeter\n", curahA)
	fmt.Printf("Curah hujan daerah B: %.4f milimeter\n", curahB)
	fmt.Printf("Curah hujan daerah C: %.4f milimeter\n", curahC)
	fmt.Printf("Curah hujan daerah D: %.4f milimeter\n", curahD)
}
```

Output:



Deskripsi Program:

Program ini mensimulasikan distribusi curah hujan di empat daerah berbeda menggunakan metode simulasi titik hujan acak. Program dimulai dengan mendeklarasikan variabel untuk menghitung jumlah titik di setiap daerah (daerahA, daerahB, daerahC, daerahD) dan totalTitik untuk menyimpan jumlah total titik hujan yang akan disimulasikan. Program meminta input jumlah titik hujan dari pengguna, kemudian menggunakan perulangan for untuk menghasilkan titik-titik hujan acak menggunakan rand.Float64() yang menghasilkan angka acak antara 0 dan 1 untuk koordinat x dan y. Area dibagi menjadi empat kuadran dengan titik tengah (0.5, 0.5): daerah A ($x < 0.5, y < 0.5$), daerah B ($x \geq 0.5, y < 0.5$), daerah C ($x \geq 0.5, y \geq 0.5$), dan daerah D ($x < 0.5, y \geq 0.5$). Setiap titik hujan akan dihitung masuk ke daerah mana berdasarkan koordinatnya. Setelah semua titik diproses, program menghitung curah hujan untuk setiap daerah dengan mengalikan jumlah titik dengan faktor 0.0001 untuk mengkonversi ke satuan milimeter. Akhirnya, program menampilkan hasil curah hujan untuk setiap daerah dengan format empat angka desimal.

DAFTAR PUSTAKA

Prayogo, N. A. (2021). ***Dasar Pemrograman Go. Ebook***