

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 7
STRUCT & ARRAY**



**Universitas
Telkom**

Oleh:

Ervan Hapiz

2311102205

IF-11-02

**S1 TEKNIK INFORMATIKA
TELKOM UNIVERSITY PURWOKERTO**

2024

I. DASAR TEORI

7.1 Tipe Bentuk

Tipe bentuk memungkinkan pemrograman untuk mendefinisikan suatu tipe data baru pada suatu bahasa pemrograman. Tipe bentuk ini dapat dibedakan atas dua jenis, yaitu Alias dan Struct.

1) Alias (Type)

Bahasa pemrograman pada umumnya mengizinkan pemrograman untuk mengubah nama suatu tipe data dengan nama baru yang lebih ringkas dan familiar. Sebagai contoh "**integer**" dapat dirubah dengan nama alias "**bilangan**". Caranya dengan menggunakan kata kunci "**type**".

	Notasi Algoritma	Notasi dalam bahasa Go
1	kamus	package main
2	type <nama alias> <tipe data>	type <nama alias> <tipe data>
3		
4	algoritma	func main(){
5
6		}
7		

Sebagai contoh perhatikan program Go berikut beserta hasil eksekusinya!

```
1 package main
2 import "fmt"
3 type bilangan int
4 type pecahan float64
5 func main(){
6     var a,b bilangan
7     var hasil pecahan
8     a = 9
9     b = 5
10    hasil = pecahan(a) / pecahan(b)
11    fmt.Println(hasil)
12 }
```

```
E:\DEV\GO>go build Demo.go
E:\DEV\GO> Demo.exe
1.8
```

2) Struct atau Record

Stucture memungkinkan pemrograman untuk mengelompokkan beberapa data atau nilai yang memiliki relasi atau keterkaitan tertentu menjadi suatu kesatuan. Masing-masing nilai tersimpan dalam field dari stucture tersebut.

	Notasi Algoritma	Notasi dalam bahasa Go
1	kamus	package main
2	type <nama struct> <	type <nama struct> struct {
3	<field 1> <tiipe data>	<field 1> <tiipe data>
4	<field 2> <tiipe data>	<field 2> <tiipe data>
5	<field 3> <tiipe data>	<field 3> <tiipe data>
6	>	}
7		

Berbeda dengan bahasa pemrograman yang lain. Kesamaan tipe dari dua variabel berjenis stucture bukan karena namanya tetapi karena strukturnya. Dua variabel dengan nama-nama field dan tipe field yang sama (dan dalam urutan yang sama) dianggap mempunyai tipe yang sama. Tentunya akan lebih memudahkan jika stucture tersebut didefinisikan sebagai sebuah tipe baru, sehingga deklarasi stucture tidak perlu lagi seluruh field-nya ditulis ulang berkali-kali.

```

1 package main
2 import "fmt"
3 type waktu struct {
4     jam, menit, detik int
5 }
6
7 func main(){
8     var wParkir, wPulang, durasi waktu
9     var dParkir, dPulang, lParkir int
10    fmt.Scan(&wParkir.jam, &wParkir.menit, &wParkir.detik)
11    fmt.Scan(&wPulang.jam, &wPulang.menit, &wPulang.detik)
12    dParkir = wParkir.detik + wParkir.menit*60 + wParkir.jam*3600
13    dPulang = wPulang.detik + wPulang.menit*60 + wPulang.jam*3600
14    lParkir = dPulang - dParkir
15    durasi.jam = lParkir / 3600
16    durasi.menit = lParkir % 3600 / 60
17    durasi.detik = lParkir % 3600 % 60
18    fmt.Printf("Lama parkir: %d jam %d menit %d detik",
19        durasi.jam, durasi.menit, durasi.detik)
20 }

```

```

E:\DEV\GO>go build Demo.go
E:\DEV\GO> Demo.exe
7 30 0
10 45 15
Lama parkir: 3 jam 15 menit 15 detik

```

7.2 Array

Array mempunyai ukuran jumlah elemen) yang tetap (static) selama eksekusi program, sehingga jumlah elemen array menjadi bagian dari deklarasi variabel dengan tipe array.

	Notasi dalam bahasa Go
1	var (
2	// array arr mempunyai 73 elemen, masing-masing bertipe CircType2
3	arr [73]CircType
4	
5	// array buf dengan 5 elemen, dengan nilai awal 7, 3, 5, 2, dan 11.
6	buf = [5]byte{7, 3, 5, 2, 11}
7	
8	// mhs adalah array dengan 2000 elemen bertipe NewType
9	mhs [2000]NewType
10	
11	// rec adalah array dari array, yaitu matriks, atau array berdimensi-2
12	rec [20][40]float64
13)

Jumlah elemen array dapat diminta dengan fungsi **len** yang tersedia. Sebagai contoh **len(arr)** akan menghasilkan 73 untuk contoh di atas.

Indeks array dimulai dari **0**, sehingga indeks arr pada contoh adalah **0, 1.. len(arr)-1**

Contoh:

```
1 // Mengganti isi elemen ke-0 dengan nilai dari elemen ke-7
2 arr[0] = arr[7]
3
4 // Mengambil data field x dari elemen ke-i
5 currX = arr[i].center.x
6
7 // Mengambil elemen terakhir
8 n := len(arr)
9 buf := arr[n-1]
```

Slice (Array dinamik)

Array dalam Go juga dapat mempunyai ukuran yang dinamik. (Tidak digunakan di kelas Algoritma Pemrograman). Deklarasinya mirip dengan deklarasi array, tetapi jumlah elemennya dikosongkan.

```

1 // declaring chop as an empty slice of float64
2 var chop []float64
3
4 // declaring sl01 as a slice
5 var sl01 = []int{ 11, 2, 3, 5, 7, 13 }

```

Sebuah slice dapat diprealokasi menggunakan fungsi built-in **make**

```

1 // Prealokasi 10 elemen untuk sl02 dan sejumlah tempat tambahan
2 var sl02 []int = make([]int, 10, 20)
3
4 // Prealokasi 7 elemen untuk sl03 tanpa tempat tambahan
5 var sl03 []circType = make([]circType, 7)

```

Fungsi built-in `len` dapat digunakan untuk mengetahui ukuran slice. Fungsi lain, `cap`, dapat digunakan untuk mengetahui total tempat yang disediakan untuk slice tersebut.

```

1 // Cetak jumlah elemen dan tempat yang tersedia untuk sl02
2 fmt.Println( len(sl02), cap(sl02) )

```

Fungsi built-in `append` dapat digunakan untuk menambahkan elemen ke suatu slice, dan bila perlu memperbesar tempat untuk slice tersebut.

Sebuah slice baru juga dapat terbentuk dengan mengambil slice dari suatu array atau slice yang lain.

```

1 // Ambil 3 elemen pertama dari suatu slice atau array
2 sl04 = arr[:4]
3
4 // Ambil beberapa elemen terakhir, dimulai dari indeks 5
5 sl05 = sl01[5:]
6
7 // Salin semua dari slice/array aslinya
8 sl06 = sl05[:]
9
10 // Salin element dari indeks 3 sampai, tapi tidak termasuk, 5.
11 // Jadi dalam contoh hanya 2 elemen sl06[3] dan sl06[4] yang disalin
12 sl07 = sl06[3:5]

```

Map

Tipe array lain, sebuah array dinamik. Indeksnya (di sini disebut kunci) tidak harus berbentuk integer. Indeks dapat berasal dari tipe apa saja. Struktur ini disebut map.

```

1 // Deklarasi variabel dct sebagai map bilangan bulat dengan kunci string
2 var dct map[string]int
3
4 // Deklarasi map lain dct1 dari elemen string dengan kunci juga string
5 // Mempunyai nilai awal dct1["john"] = "hi", dct1["anne"] = "darling"
6 var dct1 = map[string]string{ "john":"hi", "anne":"darling" }
7
8 // Deklarasi dan prealokasi tempat untuk map dct2
9 var dct2 map[float64]int = make(map[float64]int, 10)
10
11 // Mengambil nilai yang tersimpan dengan kunci "john"
12 fmt.Println( dct1["john"] )
13
14 // Mengganti nilai yang tersimpan pada kunci "anne", dan
15 // Membuat entri baru dengan kunci "boy"
16 dct1["anne"] = "lovely"
17 dct1["boy"] = "runaround"

```

```

18
19 // Menghapus entri dengan kunci "john"
20 delete(dct1, "john")

```

II. GUIDED

1. Guided 1

Source Code

```
package main

import (
    "fmt"
    "sort"
)

type mahasiswa struct {
    nama        string
    matematika  int
    fisika      int
    kimia       int
    rata_rata   float64
}

func HitungRata(n *mahasiswa) {
    n.rata_rata =
float64(n.matematika+n.fisika+n.kimia) /
float64(3)
}

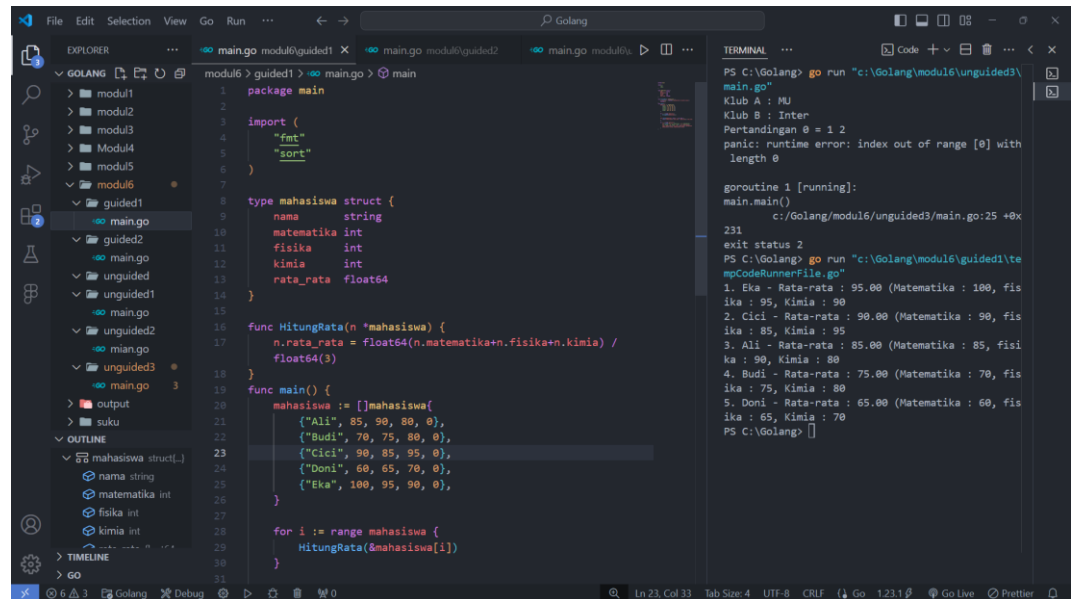
func main() {
    mahasiswa := []mahasiswa{
        {"Ali", 85, 90, 80, 0},
        {"Budi", 70, 75, 80, 0},
        {"Cici", 90, 85, 95, 0},
        {"Doni", 60, 65, 70, 0},
        {"Eka", 100, 95, 90, 0},
    }

    for i := range mahasiswa {
        HitungRata(&mahasiswa[i])
    }

    sort.Slice(mahasiswa, func(i, j int) bool {
        return mahasiswa[i].rata_rata >
mahasiswa[j].rata_rata
    })

    for i := range mahasiswa {
        fmt.Printf("%d. %s - Rata-rata : %.2f\n",
(Matematika : %d, fisika : %d, Kimia : %d\n",
i+1, mahasiswa[i].nama, mahasiswa[i].rata_rata,
mahasiswa[i].matematika, mahasiswa[i].fisika,
mahasiswa[i].kimia)
    }
}
```

Screenshot



Deskripsi

Program ini adalah program untuk mencari nilai rata rata. type mahasiswa struct {} Mendefinisikan tipe data mahasiswa dengan beberapa field (nama, matematika, fisika, kimia, dan rata_rata). Terdapat juga func HitungRata(n *mahasiswa) fungsi ini menerima pointer ke struct mahasiswa dan menghitung rata-rata nilai matematika, fisika, dan kimia, kemudian menyimpannya di Struck rata_rata. Dalam func main terdapat slice/ array mahasiswa yang berisi beberapa data mahasiswa. Dengan menggunakan perulangan for i := range mahasiswa untuk menghitung rata-rata nilai setiap mahasiswa dengan memanggil fungsi HitungRata. Kemudaian untuk mengurutkan slice mahasiswa berdasarkan nilai rata_rata dari yang tertinggi ke yang terendah menggunakan sort.Slice. loop for i := range mahasiswa untuk mencetak daftar mahasiswa beserta rata-rata nilai dan nilai mata pelajaran mereka.

2. Guided 2

Source Code

```
package main

import "fmt"

func main() {
    // Membuat map dengan NIM sebagai kunci dan
    Nama sebagai nilai
    mahasiswa := map[string]string{
        "20231001": "Andi",
        "20231002": "Budi",
        "20231003": "Cici",
    }
}
```



```

// Menambahkan data baru ke map
mahasiswa["20231004"] = "Dedi"

// Menampilkan seluruh isi map dalam format
kolom dan baris
fmt.Println("Daftar Mahasiswa:")
fmt.Println("NIM\t\tNama")
fmt.Println("-----")
for nim, nama := range mahasiswa {
    fmt.Printf("%s\t%s\n", nim, nama)
}

// Mengakses data berdasarkan NIM
nim := "20231002"
fmt.Println("\nNama Mahasiswa dengan NIM",
nim, "adalah", mahasiswa[nim])

// Menghapus data berdasarkan NIM
delete(mahasiswa, "20231003")

// Menampilkan isi map setelah data dihapus
dalam format kolom dan baris
fmt.Println("\nDaftar Mahasiswa setelah
dihapus:")
fmt.Println("NIM\t\tNama")
fmt.Println("-----")
for nim, nama := range mahasiswa {
    fmt.Printf("%s\t%s\n", nim, nama)
}
}

```

Screenshot

The screenshot shows a Go IDE with a file explorer on the left, a code editor in the center, and a terminal on the right. The code in the editor is a Go program that uses a map to store student data. The terminal output shows the program's execution, including the initial list of students, the name of a specific student, and the list after one student has been deleted.

```

PS C:\Golang> go run "c:\Golang\modul6\guided2\main.go"
Daftar Mahasiswa:
NIM      Nama
-----
20231001  Andi
20231002  Budi
20231003  Cici
20231004  Dedi

Nama Mahasiswa dengan NIM 20231002 adalah Budi

Daftar Mahasiswa setelah dihapus:
NIM      Nama
-----
20231004  Dedi
20231001  Andi
20231002  Budi
PS C:\Golang>

```

Deskripsi

Program ini adalah program untuk menambahkan dan menghapus data pada map. Terdapat deklarasi mahasiswa := map[string]string{ } membuat map dengan tipe kunci string (NIM) dan nilai string (Nama). Kemudian untuk menampilkan data mahasiswa tersebut menggunakan for nim, nama := range mahasiswa { } perulangan untuk mengiterasi setiap pasangan NIM-Nama dalam map dan mencetaknya dalam format kolom. Kemudian untuk mencari data mahasiswa berdasarkan NIM nim := "20231002": Mendefinisikan variabel nim dengan nilai "20231002". fmt.Println("\nNama Mahasiswa dengan NIM", nim, "adalah", mahasiswa[nim]) mencetak nama mahasiswa yang sesuai dengan NIM "20231002". Untuk menghapus data dalam map delete(mahasiswa, "20231003") menghapus pasangan NIM-Nama dengan kunci "20231003" dari map. Kemudian program akan menampilkan data yang dengan nama yang sudah dihapus

III. UNGUIDED

1. Unguided 1

Source Code

```
package main

import (
    "fmt"
    "math"
)

type Titik struct {
    x, y float64
}

type Lingkaran struct {
    a, b float64
    Jari float64
}

func hitungjarak(titik Titik, titik_2 Lingkaran)
float64 {
    hasil := math.Sqrt(math.Pow((titik.x-
titik_2.a), 2) + math.Pow(titik.y-titik_2.b, 2))
    return hasil
}

func posisi(titik Titik, lingkaran Lingkaran)
bool {
    hasil := hitungjarak(titik, lingkaran) <=
lingkaran.Jari
```

```

        return hasil
    }

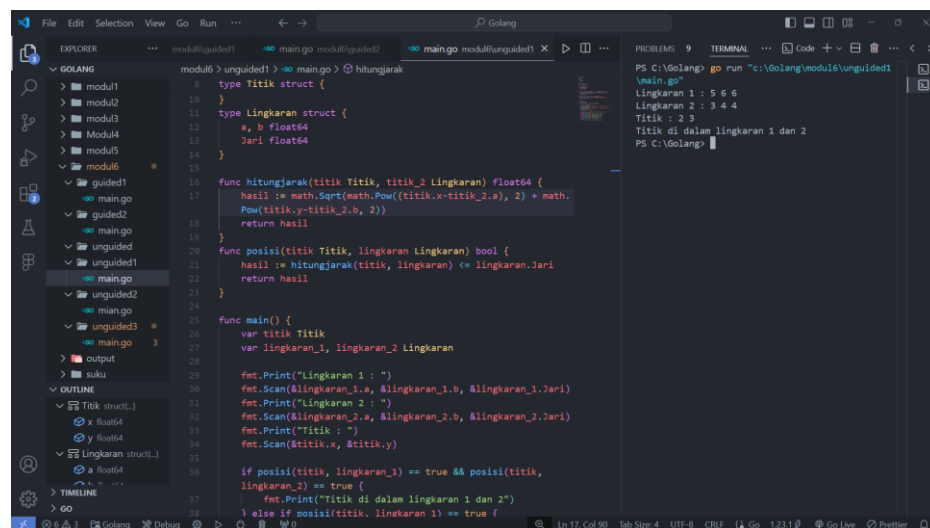
    func main() {
        var titik Titik
        var lingkaran_1, lingkaran_2 Lingkaran

        fmt.Print("Lingkaran 1 : ")
        fmt.Scan(&lingkaran_1.a, &lingkaran_1.b,
        &lingkaran_1.Jari)
        fmt.Print("Lingkaran 2 : ")
        fmt.Scan(&lingkaran_2.a, &lingkaran_2.b,
        &lingkaran_2.Jari)
        fmt.Print("Titik : ")
        fmt.Scan(&titik.x, &titik.y)

        if posisi(titik, lingkaran_1) == true &&
        posisi(titik, lingkaran_2) == true {
            fmt.Print("Titik di dalam lingkaran 1 dan
            2")
        } else if posisi(titik, lingkaran_1) == true
        {
            fmt.Print("Titik di dalam lingkaran 1")
        } else if posisi(titik, lingkaran_2) == true
        {
            fmt.Print("Titik di dalam lingkaran 2")
        } else {
            fmt.Print("Titik di luar lingkaran 1 dan
            2")
        }
    }
}

```

Screenshot



Deskripsi

Program ini untuk mencari letak titik di dalam atau luar lingkaran. Terdapat type Titik struct { x, y float64 } mendefinisikan tipe data Titik dengan dua field x dan y yang berisi koordinat titik dalam tipe float64. type Lingkaran struct { a, b float64; Jari float64 } mendefinisikan tipe data Lingkaran dengan tiga field a dan b untuk koordinat pusat lingkaran dan Jari untuk jari-jari lingkaran dalam tipe float64. Kemudian terdapat func hitungjarak(titik Titik, titik_2 Lingkaran) float64 fungsi ini menghitung jarak antara titik. Kemudian func posisi(titik Titik, lingkaran Lingkaran) bool fungsi ini memeriksa apakah jarak antara titik dan pusat lingkaran lebih kecil atau sama dengan Jari lingkaran. Jika iya, maka titik berada di dalam lingkaran, dan fungsi mengembalikan true. Dalam func main terdapat deklarasi variabel titik dari tipe Titik dan variabel lingkaran_1 dan lingkaran_2 dari tipe Lingkaran. Meminta pengguna untuk memasukkan data untuk lingkaran_1, lingkaran_2, dan titik menggunakan fmt.Scan. Menggunakan fungsi posisi untuk memeriksa apakah titik berada di dalam lingkaran_1 dan/atau lingkaran_2, kemudian mencetak hasilnya sesuai dengan kondisi tersebut.

2. Unguided 2

Source Code

```
package main

import (
    "fmt"
    "math"
)

type arrData [100]int

func InputData(Data *arrData, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Masukan Data ke-%v =", i+1)
        fmt.Scan(&Data[i])
    }
}

func Display(Data *arrData, n int) {
    fmt.Print("{")
    for i := 0; i < n; i++ {
        fmt.Printf("%v ", Data[i])
    }
    fmt.Println("}")
}
```

```

func Genap(Data *arrData, n int) {
    fmt.Println("data indeks genap : ")
    for i := 0; i < n; i++ {
        if i%2 == 0 {
            fmt.Printf("indek ke- %v = %v\n", i,
Data[i])
        }
    }
    fmt.Println()
}

func Ganjil(Data *arrData, n int) {
    fmt.Println("data Indeks ganjil: ")
    for i := 0; i < n; i++ {
        if i%2 != 0 {
            fmt.Printf("indek ke- %v = %v\n", i,
Data[i])
        }
    }
    fmt.Println()
}

func Rata(Data *arrData, n int) float64 {
    sum := 0
    for i := 0; i < n; i++ {
        sum += Data[i]
    }
    mean := float64(sum / n)
    return mean
}

func simpanganBaku(Data *arrData, n int) {
    mean := Rata(Data, n)
    var variansisum float64
    variansisum = 0.0
    for i := 0; i < n; i++ {
        variansisum += math.Pow((float64(Data[i])
- mean), 2)
    }
    var Simpang float64
    Simpang = math.Sqrt(variansisum / float64(n-
1))

    fmt.Printf("Simpangan Baku : %.2f\n",
Simpang)
}

func CariIndeks(Data *arrData, n int) {
    var x int
    fmt.Print("Masukan indeks yang dicari : ")
    fmt.Scan(&x)
    if x > n {
        fmt.Println("Indeks yang kamu cari tidak
ada")
    } else {
        fmt.Printf("Indeks %v dan kelipatannya
\n", x)
    }
}

```

```

        for i := 0; i < n; i++ {
            if i%x == 0 && i >= x {
                fmt.Printf("Indeks ke-%v = %v\n",
i, Data[i])
            }
        }
        fmt.Println()
    }

func Hapus(Data *arrData, n int) {
    var index int
    fmt.Print("Masukan index array yang ingin
dihapus : ")
    fmt.Scan(&index)
    if index >= 0 && index < n {
        Data := append(Data[:index],
Data[index+1:]...)
        n--
        fmt.Print("Array setelah dihapus : ")
        for i := 0; i < n; i++ {
            fmt.Printf("%v ", Data[i])
        }
        fmt.Println()
    } else {
        fmt.Println("index tidak valid")
    }
}

func frekuensi(Data *arrData, n int) {
    F := 0
    var index int
    fmt.Print("Masukan index array yang ingin
dihapus : ")
    fmt.Scan(&index)
    for i := 0; i < n; i++ {
        if Data[i] == index {
            F++
        }
    }
    fmt.Printf("Frekuensi dari %v adalah %v",
index, F)
}

func main() {
    var Data arrData
    var n int
    pil := 1

    fmt.Print("Masukan Banyak Data: ")
    fmt.Scan(&n)
    InputData(&Data, n)
    for pil != 0 {

        fmt.Println("1. Tampilkan")
        fmt.Println("2. Tampilkan indeks ganjil")
    }
}

```

```

        fmt.Println("3. Tampilkan indeks genap")
        fmt.Println("4. Tampilkan array dengan
indeks kelipatan x")
        fmt.Println("5. Hapus")
        fmt.Println("6. Rata Rata")
        fmt.Println("7. Simpangan Baku")
        fmt.Println("8. Frekuensi")
        fmt.Println("Masukan Pilihan menu :")
        fmt.Scan(&pil)

        switch pil {
        case 1:
            Display(&Data, n)
            fmt.Println()
            break
        case 2:
            Ganjil(&Data, n)
            fmt.Println()
            break
        case 3:
            Genap(&Data, n)
            fmt.Println()
            break
        case 4:
            CariIndeks(&Data, n)
            fmt.Println()
            break
        case 5:
            Hapus(&Data, n)
            fmt.Println()
            break
        case 6:
            fmt.Println(Rata(&Data, n))
            fmt.Println()

            break
        case 7:
            simpanganBaku(&Data, n)
            fmt.Println()

            break
        case 8:
            frekuensi(&Data, n)
            fmt.Println()
            break
        default:
            fmt.Println("tidak ada pilihan")
        }
    }
}

```

Screenshot

The screenshot shows a Go IDE with a file explorer on the left, a code editor in the center, and a terminal on the right. The code in the editor defines several functions: `InputData`, `Display`, `Genap`, `Ganjil`, `Rata`, `simpanganBaku`, and `CariIndeks`. The terminal shows the output of running the program, which prompts the user for the number of data elements (5) and then displays the data, indices, and various statistical calculations.

```
func InputData(Data *arrData, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Indek ke- %v = %v\n", i, Data[i])
    }
}

func Display(Data *arrData, n int) {
    for i := 0; i < n; i++ {
        fmt.Printf("Indek ke- %v = %v\n", i, Data[i])
    }
}

func Genap(Data *arrData, n int) {
    for i := 0; i < n; i++ {
        if i%2 == 0 {
            fmt.Printf("Indek ke- %v = %v\n", i, Data[i])
        }
    }
}

func Ganjil(Data *arrData, n int) {
    for i := 0; i < n; i++ {
        if i%2 != 0 {
            fmt.Printf("Indek ke- %v = %v\n", i, Data[i])
        }
    }
}

func Rata(Data *arrData, n int) float64 {
    sum := 0
    for i := 0; i < n; i++ {
        sum += Data[i]
    }
    mean := float64(sum / n)
    return mean
}

func simpanganBaku(Data *arrData, n int) {
    mean := Rata(Data, n)
    var variansum float64
    variansum = 0.0
    for i := 0; i < n; i++ {
        variansum += math.Pow(float64(Data[i]) - mean, 2)
    }
    var simpang float64
    simpang = math.Sqrt(variansum / float64(n-1))
}
```

Deskripsi

Program ini adalah program untuk menampilkan dan memanipulasi data array

- **Fungsi InputData:**func InputData(Data *arrData, n int) ini menerima pointer ke arrData dan integer n. Fungsi ini meminta pengguna untuk memasukkan nilai data sebanyak n elemen dan menyimpannya dalam Data.
- **Fungsi Display:**
func Display(Data *arrData, n int) ini menampilkan elemen-elemen dalam Data sebanyak n elemen dalam format { ... }.
- **Fungsi Genap:**
func Genap(Data *arrData, n int) ini menampilkan elemen-elemen dalam Data yang berada di indeks genap.
- **Fungsi Ganjil:**
func Ganjil(Data *arrData, n int) ini menampilkan elemen-elemen dalam Data yang berada di indeks ganjil.
- **Fungsi Rata:**
func Rata(Data *arrData, n int) float64 ini menghitung rata-rata elemen dalam Data sebanyak n elemen dan mengembalikan nilai rata-rata tersebut.
- **Fungsi simpanganBaku:**
func simpanganBaku(Data *arrData, n int) ini menghitung simpangan baku elemen dalam Data sebanyak n elemen dan menampilkannya.
- **Fungsi CariIndeks:**

func CariIndeks(Data *arrData, n int) ini meminta pengguna untuk memasukkan indeks yang ingin dicari, kemudian menampilkan elemen pada indeks tersebut dan kelipatannya.

- **Fungsi Hapus:**

func Hapus(Data *arrData, n int) ini meminta pengguna untuk memasukkan indeks yang ingin dihapus, kemudian menghapus elemen pada indeks tersebut dari Data.

- **Fungsi frekuensi:**

func frekuensi(Data *arrData, n int) fungsi ini meminta pengguna untuk memasukkan nilai yang ingin dihitung frekuensinya dalam Data, kemudian menghitung dan menampilkan frekuensi nilai tersebut.

- **Fungsi utama (main):**

func main() { ... }: Fungsi main adalah titik masuk dari aplikasi Go. Mendeklarasikan variabel Data dari tipe arrData dan n dari tipe int. Meminta pengguna untuk memasukkan jumlah data n dan nilai-nilai data tersebut. Menyediakan menu interaktif dengan berbagai pilihan untuk menampilkan data, menampilkan data dengan indeks ganjil/genap, menampilkan data dengan indeks kelipatan tertentu, menghapus data, menghitung rata-rata, menghitung simpangan baku, dan menghitung frekuensi nilai.

3. Unguided 3

Source Code

```
package main
import "fmt"

type nama struct {
    Nama string
    Skor int
}

func main() {
    var nama1, nama2 nama
    var pemenang []string
    var tanding int = 1

    fmt.Print("Masukkan nama nama pertama: ")
    fmt.Scanln(&nama1.Nama)
    fmt.Print("Masukkan nama nama kedua: ")
    fmt.Scanln(&nama2.Nama)
```

```

for {
    fmt.Printf("pertandingan %v: ",tanding)
    fmt.Scanln(&nama1.Skor,&nama2.Skor)
    tanding++

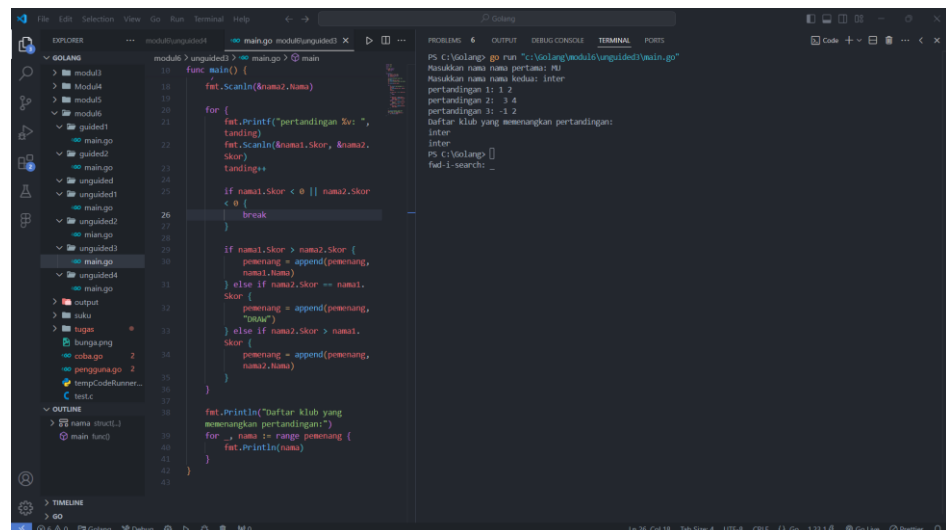
    if nama1.Skor < 0 || nama2.Skor < 0 {
        break
    }

    if nama1.Skor > nama2.Skor {
        pemenang = append(pemenang, nama1.Nama)
    } else if nama2.Skor == nama1.Skor {
        pemenang = append(pemenang, "DRAW")
    } else if nama2.Skor > nama1.Skor {
        pemenang = append(pemenang, nama2.Nama)
    }
}

fmt.Println("Daftar klub yang memenangkan
pertandingan:")
for _, nama := range pemenang {
    fmt.Println(nama)
}
}

```

Screenshot



Deskripsi

Untuk menentukan pemenang setiap pertandingan dengan menyimpan di dalam array

4. Unguided 4 Source Code

```

package main

import (
    "fmt"
)

const NMAX int = 127

type tabel [NMAX]rune

func isiArray(t *tabel, n *int) {
    var char rune
    *n = 0
    fmt.Print("Masukkan karakter (gunakan (.)
untuk berhenti): ")

    for *n < NMAX {
        fmt.Scanf("%c", &char)
        if char == '.' {
            break
        }

        if char != '\n' {
            t[*n] = char
            *n++
        }
    }
}

func balikanArray(t tabel, n int) tabel {
    var hasil tabel
    for i := 0; i < n; i++ {
        hasil[i] = t[n-1-i]
    }
    return hasil
}

func palindrom(t tabel, n int) bool {
    for i := 0; i < n/2; i++ {
        if t[i] != t[n-1-i] {
            return false
        }
    }
    return true
}

func cetakArray(t tabel, n int) {
    var balik tabel = balikanArray(t, n)
    isPalindrome := palindrom(t, n)

    fmt.Print("Teks\t\t:")
    for i := 0; i < n; i++ {
        fmt.Printf(" %c", t[i])
    }
    fmt.Println()
}

```

```

        fmt.Print("Reverse teks\t:")
        for i := 0; i < n; i++ {
            fmt.Printf(" %c", balik[i])
        }
        fmt.Println()

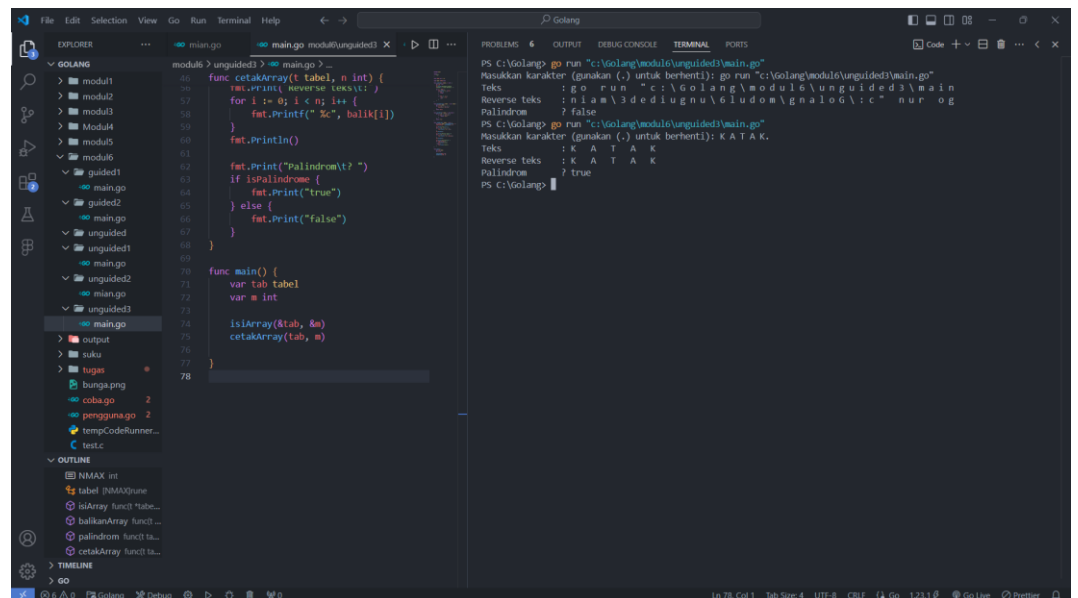
        fmt.Print("Palindrom\t? ")
        if isPalindrome {
            fmt.Print("true")
        } else {
            fmt.Print("false")
        }
    }

    func main() {
        var tab tabel
        var m int

        isiArray(&tab, &m)
        cetakArray(tab, m)
    }

```

Screenshot



Deskripsi

Program ini adalah implementasi sederhana yang bertujuan untuk memeriksa apakah sebuah teks merupakan palindrom, yaitu teks yang terbaca sama dari depan maupun dari belakang. Program dimulai dengan mendefinisikan konstanta NMAX sebesar 127 dan tipe array tabel yang

berisi karakter (rune). Fungsi `isiArray` bertugas untuk mengisi array tabel dengan karakter yang dimasukkan oleh pengguna hingga karakter titik (.) dimasukkan atau hingga array mencapai batas maksimal. Setelah itu, fungsi `balikanArray` akan membuat array baru yang merupakan kebalikan dari array asli. Kemudian, fungsi `palindrom` digunakan untuk memeriksa apakah array tabel adalah palindrom dengan membandingkan elemen pertama dengan elemen terakhir, elemen kedua dengan elemen kedua dari akhir, dan seterusnya. Fungsi `cetakArray` akan mencetak array asli, array yang telah dibalik, dan hasil pengecekan apakah array tersebut adalah palindrom. Akhirnya, fungsi `main` menginisialisasi array tabel, memanggil `isiArray` untuk mengisi array, dan memanggil `cetakArray` untuk menampilkan hasilnya.

5. Unguided 2

Source Code

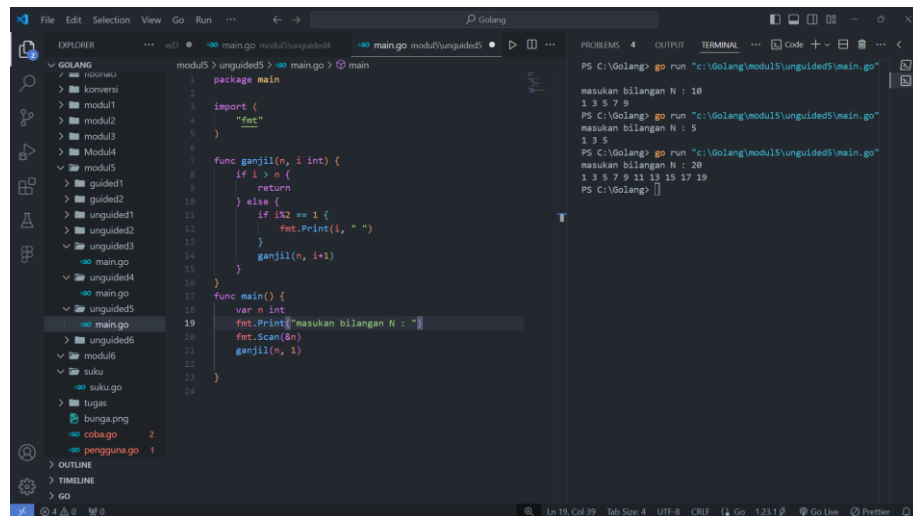
```
package main

import (
    "fmt"
)

func ganjil(n, i int) {
    if i > n {
        return
    } else {
        if i%2 == 1 {
            fmt.Print(i, " ")
        }
        ganjil(n, i+1)
    }
}

func main() {
    var n int
    fmt.Print("masukan bilangan N : ")
    fmt.Scan(&n)
    ganjil(n, 1)
}
```

Screenshot



Deskripsi

Program ini adalah program untuk menampilkan bilangan ganjil dari 1 sampai n. terdapat dunc ganjil () sebagai fungsi rekursif dengan parameter n (sebagai batas angka) dan i (sebagai angka yang akan dicek). If i > n sebagai batas atau base case dari rekursif. Pada else terdapat if i % 2 yang akan mengecek I apakah bilangan ganjil atau tidak. Jika memenuhi maka akan dicetak i. dan terjadi pemanggilan fungsi rekursif ganjil(n, i + 1). Pada func main() terdapat deklarasi variable n , program akan menerima input untuk nilai n. kemudian pemanggilan func ganjil (n, 1) i=1 untuk nilai awal.

6. Unguided 6

Source Code

```

package main

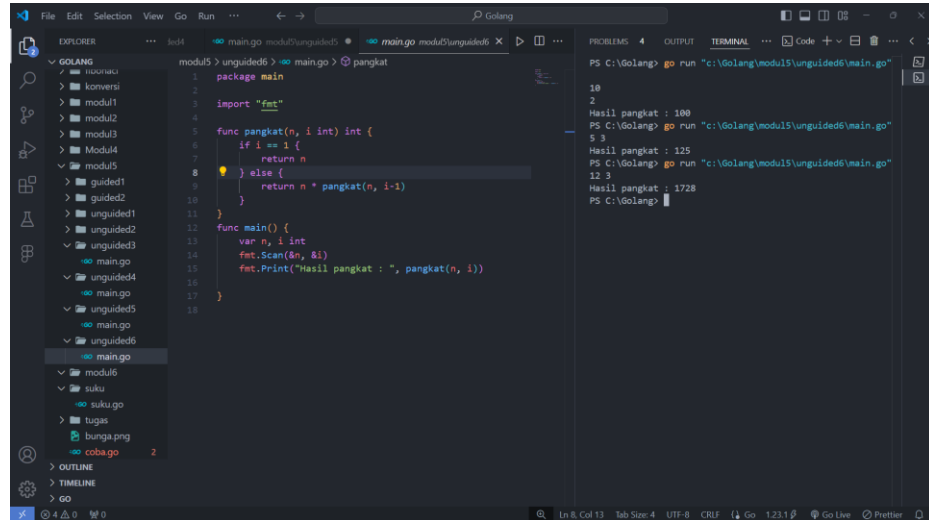
import "fmt"

func pangkat(n, i int) int {
    if i == 1 {
        return n
    } else {
        return n * pangkat(n, i-1)
    }
}

func main() {
    var n, i int
    fmt.Scan(&n, &i)
    fmt.Print("Hasil pangkat : ", pangkat(n, i))
}

```

Screenshot



The screenshot shows a Go IDE with a project structure on the left, a code editor in the center, and a terminal on the right. The code editor displays a Go program with a recursive function `pangkat` and a `main` function. The terminal shows the output of running the program for different input values.

```
package main

import "fmt"

func pangkat(n, i int) int {
    if i == 1 {
        return n
    } else {
        return n * pangkat(n, i-1)
    }
}

func main() {
    var n, i int
    fmt.Scan(&n, &i)
    fmt.Print("Hasil pangkat : ", pangkat(n, i))
}
```

Terminal Output:

```
PS C:\Golang> go run "c:\Golang\modul5\unguided6\main.go"
10
2
Hasil pangkat : 100
PS C:\Golang> go run "c:\Golang\modul5\unguided6\main.go"
5 3
Hasil pangkat : 125
PS C:\Golang> go run "c:\Golang\modul5\unguided6\main.go"
12 3
Hasil pangkat : 1728
PS C:\Golang>
```

Deskripsi

Program ini adalah program untuk menampilkan hasil pangkat dari n pangkat i . pada func `pangkat()` sebagai fungsi rekursif dengan parameter n sebagai bilangan yang akan dipangkatkan dan i sebagai bilangan pangkat. If $i == 1$ maka fungsi rekursif akan berhenti. Pada else terdapat `return n * pangkat(n, i-1)` ini akan mengalikan n dengan fungsi rekursif yang dipanggil dengan i decrement. Pada func `main()` terdapat deklarasi variable n , program akan menerima input untuk nilai n . kemudian pemanggilan func ganjil $(n, 1) i=1$ untuk nilai awal.