LAPORAN PRAKTIKUM

Algoritma Pemrograman

MODUL 2

I/O, TIPE DATA & VARIABEL

Disusun oleh:

FAREL JULIYANDRA RESTU HERMAWAN

109082530038

S1IF-13-04

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

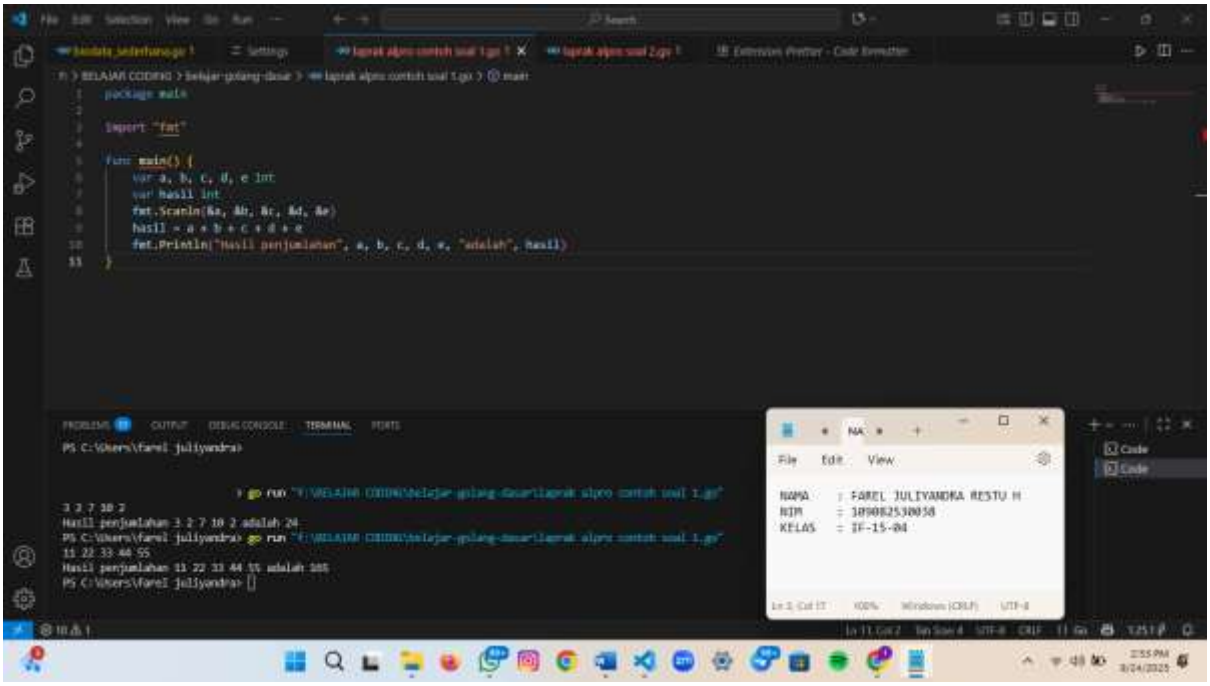
LATIHAN KELAS – GUIDED

1. **Guided 1**

Source Code

```
package main


import "fmt"


func main() {
	var a, b, c, d, e int
	var hasil int
	fmt.Scanln(&a, &b, &c, &d, &e)
	hasil = a + b + c + d + e
	fmt.Println("Hasil penjumlahan", a, b, c, d, e,
"adalah", hasil)
}
```

Screenshoot program



Deskripsi program

1. Struktur Program

- package main: Menandakan berkas ini merupakan program mandiri yang memiliki fungsi main sebagai titik awal eksekusi.
- import "fmt": Mengimpor paket fmt untuk kebutuhan input/output seperti Scanln dan Println.
- func main(): Fungsi utama tempat semua logika program dijalankan.

2. Deklarasi Variabel

- var a, b, c, d, e int: Mendeklarasikan lima variabel bertipe int untuk menampung lima bilangan bulat dari pengguna; zero value awalnya 0.
- var hasil int: Variabel penampung jumlah dari kelima bilangan yang dibaca; juga bertipe int dan berawal 0

3. Pembacaan Input
   fmt.Scanln(&a, &b, &c, &d, &e):
   - Scanln membaca input dari stdin hingga mencapai newline dan mem-parsing token yang dipisah spasi ke alamat variabel yang diberikan.
   - Ampersand (&) artinya memberikan alamat memori variabel (pointer), karena Scanln perlu menuliskan nilai hasil parsing ke variabel tersebut.
   - Urutan token input harus cocok dengan urutan variabel: token pertama ke a, kedua ke b, dan seterusnya; jika jumlah token kurang, Scanln akan mengembalikan error dan sebagian variabel mungkin tetap pada nilai default.
4. Operasi Aritmetika
   hasil = a + b + c + d + e:
   - Menjumlahkan kelima bilangan; karena semuanya int, operasi dilakukan dengan aritmetika bilangan bulat.
   - Tidak ada penanganan overflow; pada arsitektur 64-bit praktis aman untuk angka wajar, namun tetap perlu diwaspadai jika angka sangat besar

5. Keluran
   fmt.Println("Hasil penjumlahan", a, b, c, d, e, "adalah", hasil):
   - Println akan menyisipkan spasi antar argumen secara otomatis dan menambahkan newline di akhir, sehingga format rapi tanpa perlu “\n” manual.
   - Menampilkan ulang angka yang diinput agar jelas konteks penjumlahan yang dihitung
6. Terminal

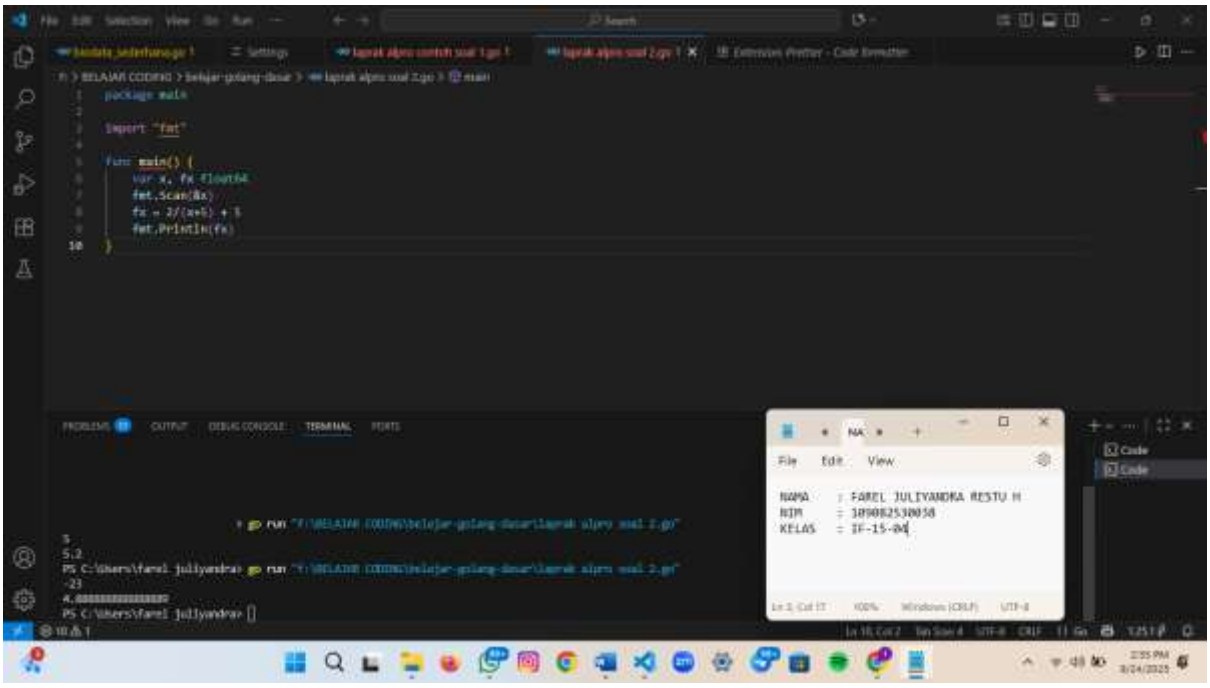
- Baris input “3 2 7 10 2” menghasilkan output “Hasil penjumlahan 3 2 7 10 2 adalah 24”, sesuai perhitungan 3+2+7+10+2.
- Baris input “11 22 33 44 55” menghasilkan “Hasil penjumlahan 11 22 33 44 55 adalah 165”, sesuai 11+22+33+44+55.
- Eksekusi menggunakan go run dari PowerShell di VS Code Terminal, terlihat dua kali pemanggilan dengan input berbeda dan hasil sesuai

2. Guided 2

Source Code

```
package main


import "fmt"


func main() {
	var x, fx float64
	fmt.Scan(&x)
	fx = 2/(x+5) + 5
	fmt.Println(fx)
}
```

Screenshoot program



Deskripsi program

- package main: Menandakan berkas adalah program eksekusi mandiri.
- import "fmt": Memakai paket fmt untuk input dengan Scan dan output dengan Println.
- var x, fx float64: Mendefinisikan dua variabel bertipe float64 agar perhitungan desimal akurat.
- fmt.Scan(&x): Membaca satu token dari stdin dan menyimpannya ke x. Tanda & mengoper alamat variabel karena fungsi akan menulis nilai ke sana.

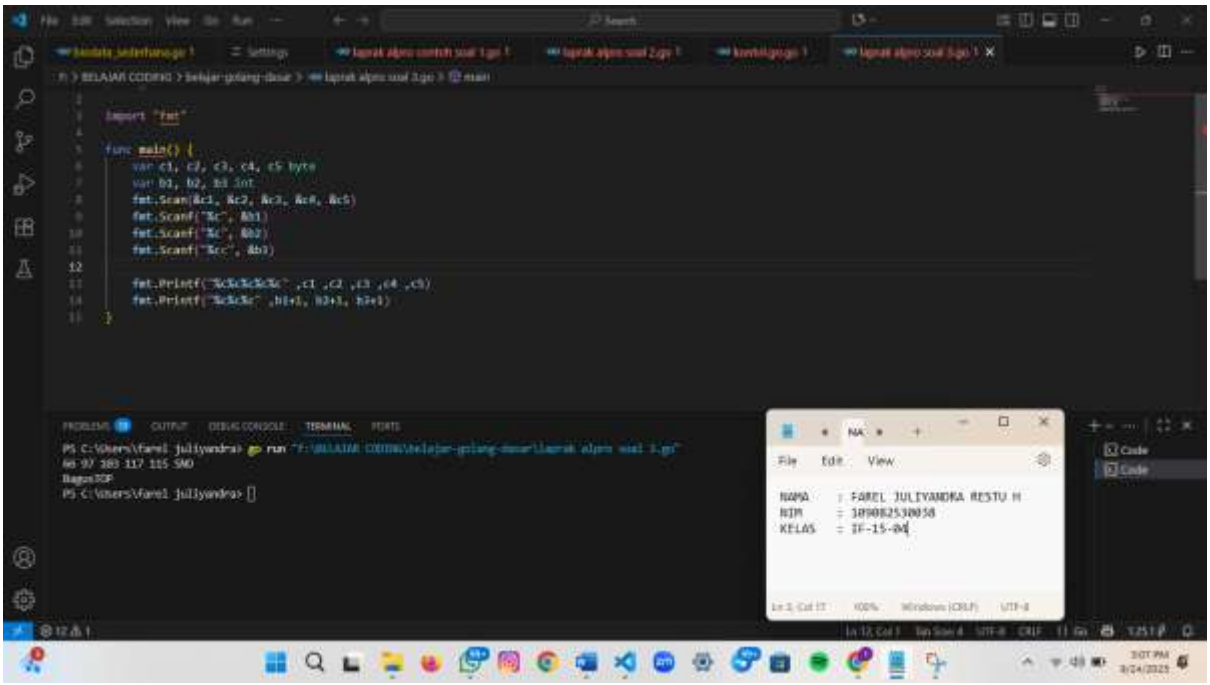
- fx = 2/(x+5) + 5: Menghitung f(x). Karena x bertipe float64, literal 2 dan 5 akan dipromosikan ke float64 pada operasi aritmetika.
- fmt.Println(fx): Mencetak hasil dalam format default, diakhiri newline

3. Guided 3

Source Code

```
package main


import "fmt"


func main() {
	var c1, c2, c3, c4, c5 byte
	var b1, b2, b3 int
	fmt.Scan(&c1, &c2, &c3, &c4, &c5)
	fmt.Scanf("%c", &b1)
	fmt.Scanf("%c", &b2)
	fmt.Scanf("%cc", &b3)


	fmt.Printf("%c%c%c%c%c" ,c1 ,c2 ,c3 ,c4 ,c5)
	fmt.Printf("%c%c%c" ,b1+1, b2+1, b3+1)
}
```

Screenshoot program



Deskripsi program

1. Deklarasi tipe
   - c1..c5 bertipe byte (alias uint8). Cocok untuk menyimpan satu karakter ASCII/byte.
   - b1..b3 bertipe int, tetapi akan diperlakukan sebagai karakter saat dicetak dengan %c. Secara teknis boleh, namun tidak konsisten; lebih rapi jika juga byte atau rune.
2. Membaca 5 karakter pertama : fmt.Scan(&c1,&c2,&c3,&c4,&c5)
   Scan membaca “token” yang dipisahkan whitespace (spasi, tab, newline). Untuk karakter tunggal, ini berarti:
   - Jika inputnya “A B C D E”, Scan akan mengisi 'A','B','C','D','E' dengan baik.
   - Jika input “ABCDE” tanpa spasi, Scan akan berusaha membaca satu token “ABCDE” ke c1 lalu gagal ke variabel selanjutnya, karena %v default untuk byte mengharapkan angka atau 1 token yang kemudian dikonversi. Intinya, Scan kurang cocok untuk “tepat 1 karakter per variabel” bila input tidak dipisah spasi.
   - Scan melewati whitespace awal, sehingga karakter pertama tidak akan menjadi spasi kecuali spasi itu dimasukkan sebagai token tersendiri (yang biasanya tidak).

   Implikasi: Penggunaan Scan di sini asumtif—butuh input dipisah whitespace agar stabil.
3. Membaca 3 karakter berikutnya dengan Scanf dan format %c
   - fmt.Scanf("%c", &b1) membaca tepat 1 byte, termasuk whitespace. Berbeda dengan Scan yang melewati whitespace, %c akan menangkap newline atau spasi sisa dari input sebelumnya.
   - Karena setelah fmt.Scan biasanya tersisa newline di buffer, panggilan pertama Scanf("%c",&b1) kemungkinan besar membaca '\n' alih-alih karakter yang diinginkan.
   - Solusi umum: beri spasi di depan verb " %c" agar Scanf melewati whitespace. Format " %c" artinya abaikan whitespace sebelum mengambil satu karakter.

4. Bug pada format "%cc"
   fmt.Scanf("%cc", &b3) berarti:
   - Baca tepat 1 karakter ke b3.

- Lalu cocokkan literal huruf 'c' pada input (bukan variabel), persis ‘c’. Jika input tidak mengandung 'c' segera setelah karakter yang dibaca, operasi gagal dan b3 mungkin tidak terisi sesuai harapan.

Jika tujuan hanya membaca 1 karakter, harusnya "%c" (atau " %c" untuk melewati whitespace).

5. Pencetakan
    - fmt.Printf("%c%c%c%c%c", c1, c2, c3, c4, c5): mencetak lima byte sebagai karakter berturut-turut.
    - fmt.Printf("%c%c%c", b1+1, b2+1, b3+1): menambah 1 pada nilai integer, lalu memetakannya sebagai karakter. Misal 'a'(97) menjadi 'b'(98). Jika b1..b3 bukan karakter ASCII (atau berada di batas 255 untuk byte), hasilnya bisa tidak terduga.

TUGAS

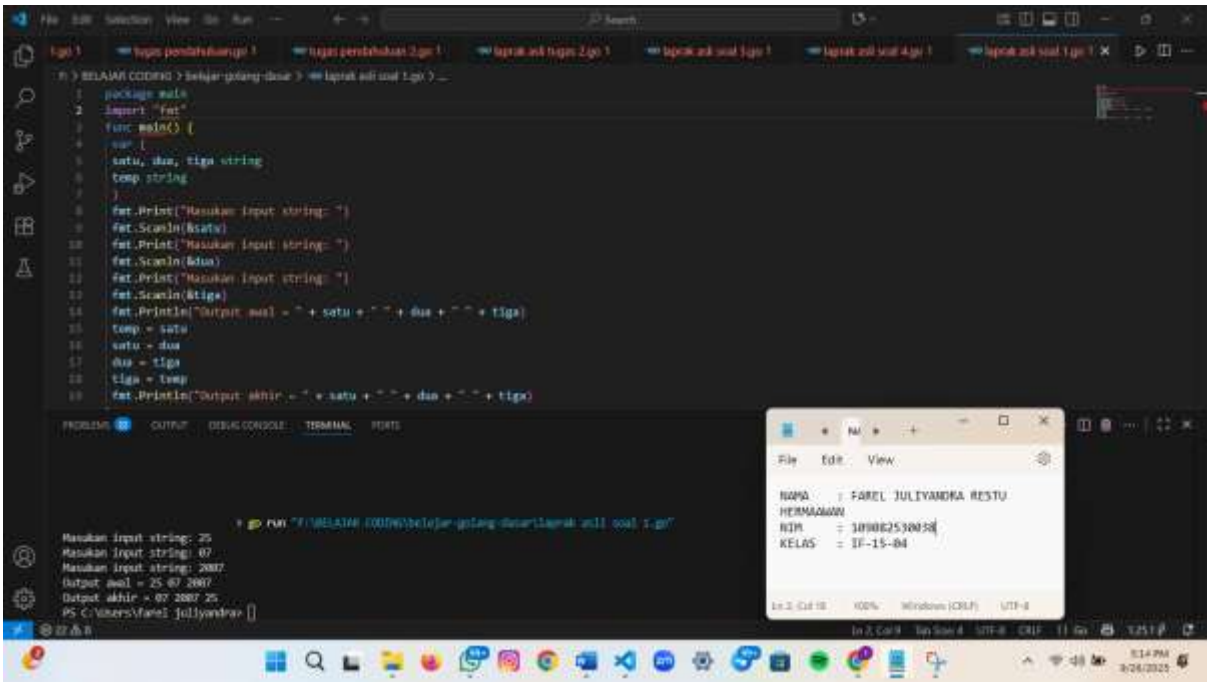
1. **Tugas 1**

Source code

```
package main

import "fmt"

func main() {

 var (

 satu, dua, tiga string

 temp string

 )

 fmt.Print("Masukan input string: ")

 fmt.Scanln(&satu)

 fmt.Print("Masukan input string: ")

 fmt.Scanln(&dua)

 fmt.Print("Masukan input string: ")

 fmt.Scanln(&tiga)

 fmt.Println("Output awal = " + satu + " " + dua + " " +
tiga)

 temp = satu
```

```
 satu = dua

 dua = tiga

 tiga = temp

 fmt.Println("Output akhir = " + satu + " " + dua + " " +
tiga)

}
```

Screenshoot program



Deskripsi program

Langkah-langkah yang Dilakukan Program :

1. Mendeklarasikan variabel:
   - Tiga variabel string: satu, dua, tiga untuk menampung input.
   - Satu variabel bantu: temp untuk proses penukaran.
2. Menerima input dari pengguna sebanyak tiga kali, lalu tiap input disimpan ke variabel berturut-turut (satu, dua, tiga).
3. Menampilkan nilai awal dari ketiga variabel.
4. Menukar posisi variabel:
   - Nilai awal satu disimpan sementara di temp.
   - Nilai dua dipindahkan ke satu.
   - Nilai tiga dipindahkan ke dua.
   - Nilai awal satu (yang sudah ada di temp) dipindahkan ke tiga.
5. Menampilkan hasil akhirnya: sekarang urutan variabelnya sudah berubah.
   - Contoh Run

   Jika pengguna memasukkan:
   - 25 (satu)
   - 07 (dua)

- 2007 (tiga)

Urutan awal:

- satu = 25
- dua = 07
- tiga = 2007

Setelah penukaran:

- satu = 07 (awalnya dari dua)
- dua = 2007 (awalnya dari tiga)
- tiga = 25 (awalnya dari satu)

Fungsi di Setiap Baris

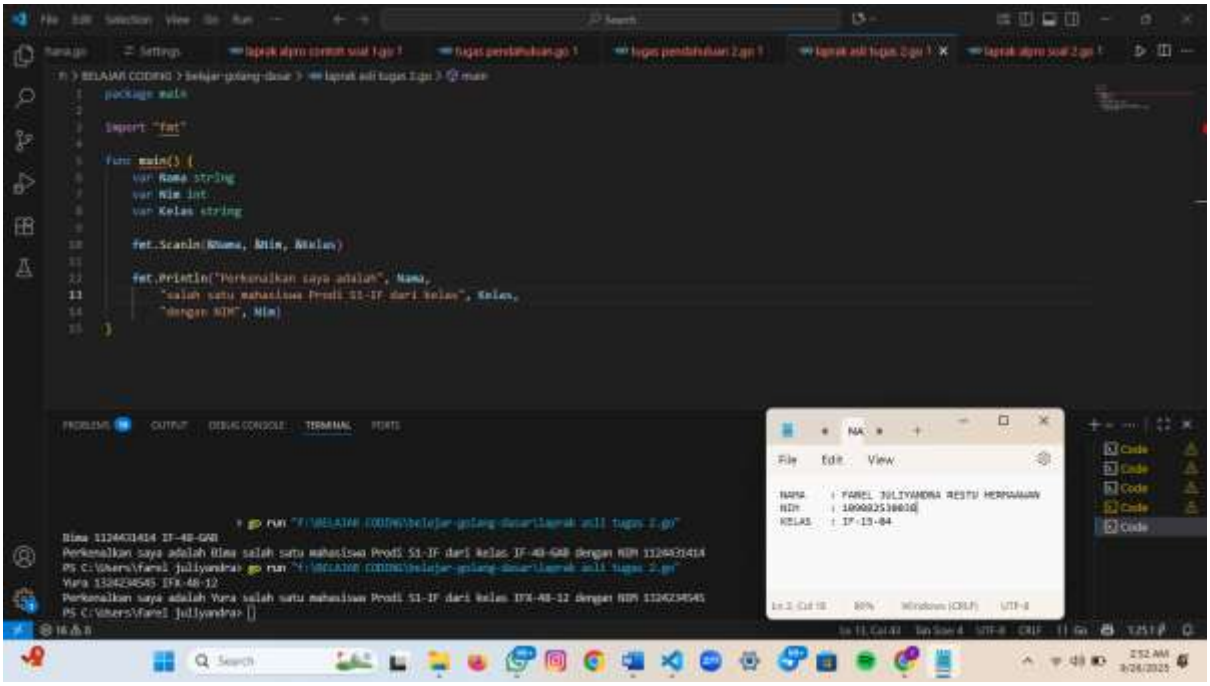
- fmt.Scanln(&satu) dst: membaca input user ke variabel keduanya.
- fmt.Println(...): menampilkan variabel ke layar.
- Blok penukaran: memastikan urutan nilai dari ketiga variabel berubah secara melingkar.

2. Tugas 2

Source code

```
package main


import "fmt"


func main() {
	var Nama string
	var Nim int
	var Kelas string
	fmt.Scanln(&Nama, &Nim, &Kelas)
	fmt.Println("Perkenalkan saya adalah", Nama,
		"salah satu mahasiswa Prodi dari kelas", Kelas,
		"dengan NIM", Nim)
	}
```

Screenshoot program



Deskripsi program

- Baris package main menandakan program executable dengan titik masuk fungsi main, bukan library; Go akan mengeksekusi fungsi main sebagai entry point.
- Import "fmt" memuat paket fmt yang menyediakan fungsi I/O terformat seperti Scanln, Println, dan Printf untuk membaca/menulis dari stdin/stdout
- var Nama string dan var Kelas string menyiapkan penampung teks, sedangkan var Nim int menyiapkan penampung bilangan bulat; tipe ini menentukan cara parsing input oleh Scanln.
- Dalam konteks Scanln, argumen yang diteruskan harus berupa pointer ke variabel menggunakan operator & agar fungsi dapat mengisi nilai variabel dari input standar
- fmt.Scanln(&Nama, &Nim, &Kelas) memindai teks dari stdin, memisah berdasarkan spasi, dan berhenti ketika menemukan newline; nilai disimpan berturut-turut ke variabel tujuan sesuai urutan argumen.
- Setelah item terakhir, harus ada newline atau EOF; fungsi mengembalikan n (jumlah item yang berhasil dipindai) dan err untuk deteksi masalah seperti kekurangan token atau kegagalan konversi tipe
- fmt.Scanln(&Nama, &Nim, &Kelas) memindai teks dari stdin, memisah berdasarkan spasi, dan berhenti ketika menemukan newline; nilai disimpan berturut-turut ke variabel tujuan sesuai urutan argumen.
- Setelah item terakhir, harus ada newline atau EOF; fungsi mengembalikan n (jumlah item yang berhasil dipindai) dan err untuk deteksi masalah seperti kekurangan token atau kegagalan konversi tipe
- fmt.Println("Perkenalkan saya adalah", Nama, "salah satu mahasiswa Prodi S1-IF dari kelas", Kelas, "dengan NIM", Nim) mencetak semua argumen dengan spasi otomatis di antaranya lalu menambahkan newline di akhir baris, tanpa perlu konversi manual tipe ke string.

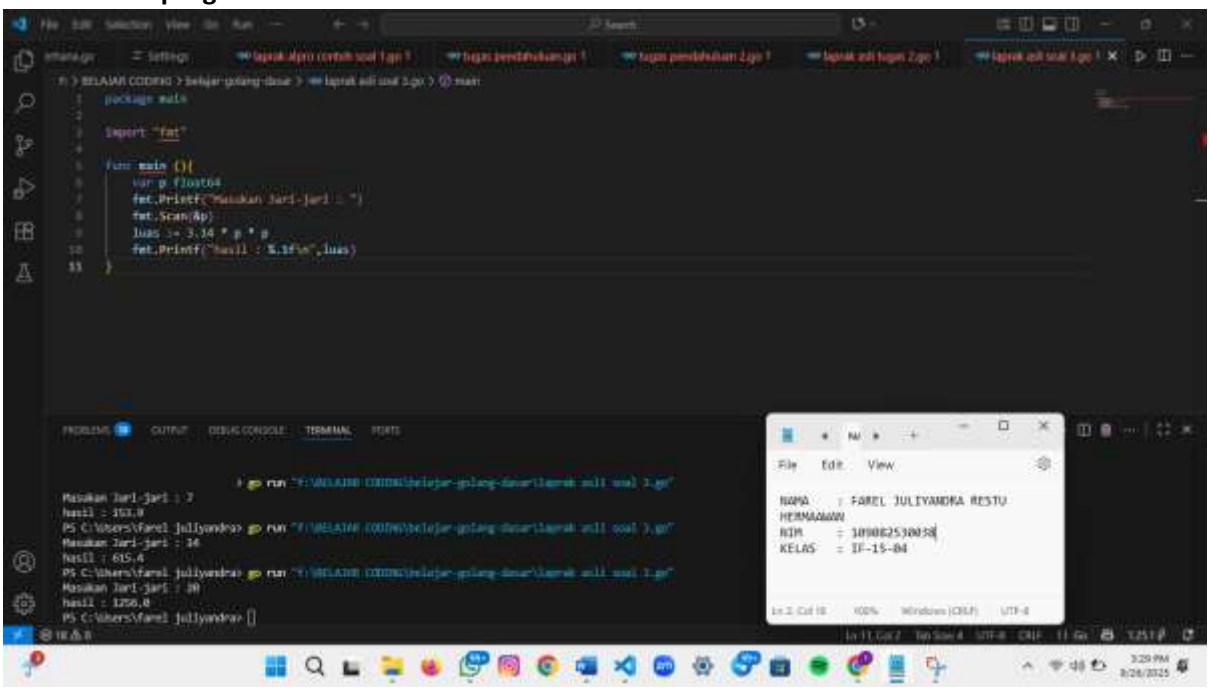
3. Tugas 3

Source code

```
package main
```

```
import "fmt"


func main (){
        var p float64
        fmt.Printf("Masukan Jari-jari : ")
        fmt.Scan(&p)
        luas := 3.14 * p * p
        fmt.Printf("hasil : %.1f\n",luas)
}
```

Screenshoot program



Deskripsi program

1. Package Main
   - Baris ini menandakan bahwa file Go tersebut adalah bagian dari package main.
   - package main adalah paket khusus di Go yang digunakan untuk membuat program yang bisa dieksekusi (bukan library).
   - Hanya package main yang boleh memiliki fungsi mai sebagnai titik awal eksekusi program.

2. import “fmt”
   - Baris ini mengimpor paket fmt dari pustaka standar Go.
   - Paket fmt menyediakan fungsi untuk input/output terformat, seperti Println, Printf, dan Scan.
3. func main()

- Fungsi ini adalah entry point atau titik awal eksekusi program Go.
- Fungsi main harus berada di package main, tidak menerima argumen, dan tidak mengembalikan nilai.
- Ketika program dijalankan, eksekusi dimulai dari fungsi ini.

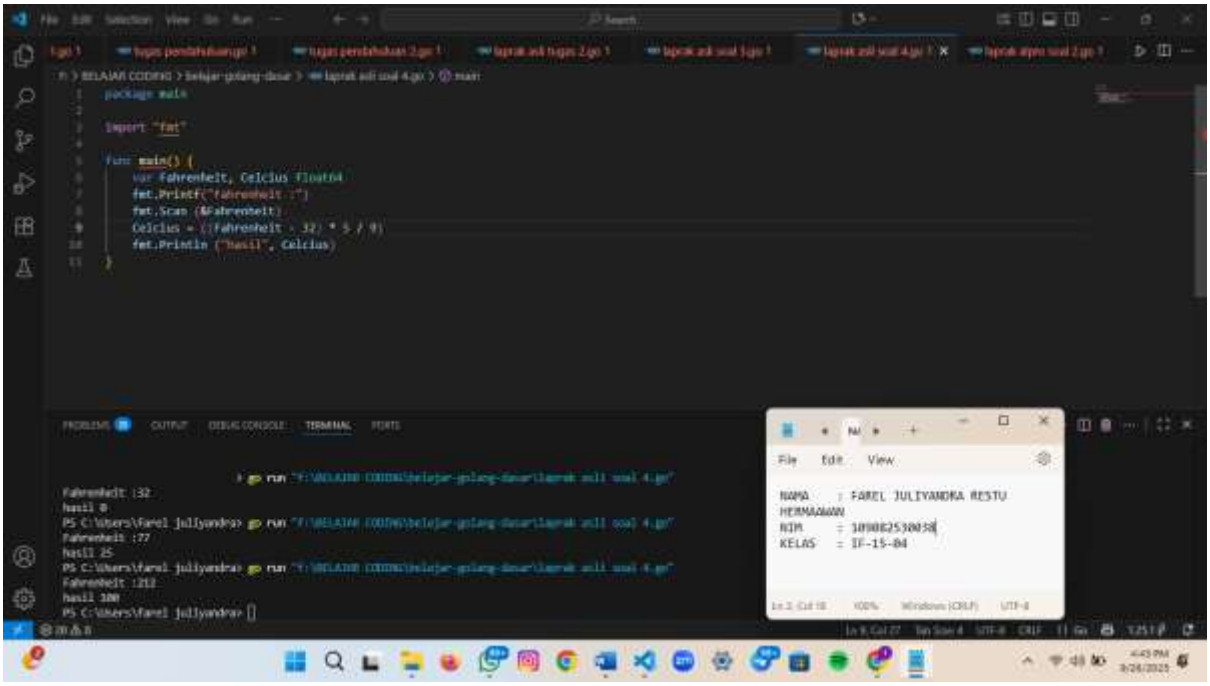
4. Deklarasi Variabel
   - Biasanya, variabel dideklarasikan di dalam fungsi main, misalnya:
     var nama string
     var nim int
     var kelas string
   - Variabel ini digunakan untuk menyimpan data yang akan diinput oleh pengguna.
5. Membaca Input
   - Untuk membaca input dari pengguna, digunakan fungsi dari paket fmt, misalnya:
     fmt.Scanln(&nama, &nim, &kelas)
   - Fungsi ini membaca input dari terminal dan menyimpannya ke variabel yang sudah dideklarasikan.
6. Menampilkan Output
   Untuk menampilkan hasil ke layar, digunakan fungsi seperti:
   fmt.Println("Perkenalkan saya", nama, "dengan NIM", nim, "dari kelas", kelas)
   Fungsi Println akan mencetak argumen ke layar dengan spasi otomatis di antara argumen dan baris baru di akhir.

4. Tugas 4

Source code

```
package main

import "fmt"

func main() {
	var Fahrenheit, Celcius float64
	fmt.Printf("Fahrenheit :")
	fmt.Scan (&Fahrenheit)
	Celcius = ((Fahrenheit - 32) * 5 / 9)
	fmt.Println ("hasil", Celcius)
}
```

Screenshoot program



Deskripsi program

1. Struktur dasar
   - package main mendeklarasikan berkas ini sebagai paket eksekusi; hanya paket bernama main yang bisa dikompilasi menjadi program yang bisa dijalankan.
   - import "fmt" memasukkan paket standar fmt yang menyediakan fungsi input/output seperti Print, Println, Printf, Scan, Scanf.
   - Fungsi utama
     Func main() { ... } adalah titik awal eksekusi program; saat program dijalankan, kode di dalam main akan diproses dari atas ke bawah.
2. Deklarasi variabel
   - var Fahrenheit, Celcius float64 membuat dua variabel bertipe float64 untuk menyimpan angka pecahan: Fahrenheit sebagai input, Celcius sebagai hasil konversi.
   - Penamaan idiomatis di Go biasanya huruf kecil untuk variabel lokal, misalnya fahrenheit, celsius, agar konsisten dengan konvensi gaya Go.
   - Output prompt ke pengguna
     fmt.Printf("Fahrenheit :") mencetak teks tanpa newline sebagai prompt, sehingga kursor tetap di baris yang sama menunggu input.
3. Membaca input
   - fmt.Scan(&Fahrenheit) membaca angka dari stdin hingga whitespace dan menaruhnya ke variabel Fahrenheit; tanda & artinya memberikan alamat variabel (pointer) supaya fungsi dapat mengisi nilainya.
   - Alternatifnya, fmt.Scanf("%f", &Fahrenheit) bisa dipakai untuk memastikan format angka pecahan sesuai spesifier %f, serta memudahkan validasi error dengan nilai balik dan error.
   - Rumus konversi
     Celcius = ((Fahrenheit - 32) * 5 / 9) menghitung suhu dalam Celcius menggunakan formula fisika: C = (F − 32) × 5/9.
   - Supaya lebih jelas bahwa pembagian dilakukan sebagai bilangan pecahan, sering ditulis (5.0/9.0) atau dikalikan konstanta 0.555555555555...; karena variabel bertipe float64, ekspresi ini sudah dievaluasi dalam domain float, tetapi menuliskan literal desimal meningkatkan keterbacaan.

- Urutan operasi: kurung dievaluasi dulu Fahrenheit - 32, kemudian dikali 5, lalu dibagi 9.

4. Menampilkan hasil
   - fmt.Println("hasil", Celcius) mencetak kata “hasil” diikuti nilai variabel Celcius, lalu menambahkan newline di akhir baris.
   - Jika ingin membatasi jumlah digit desimal, gunakan fmt.Printf("hasil: %.2f\n", Celcius) sehingga hanya dua angka di belakang koma yang ditampilkan.

TUGAS PENDAHULUAN

1. Jelaskan perbedaan fmt.Print(), fmt.Println(), dan fmt.Printf() di Go!
2. Jelaskan penggunaan tipe data int, float64, bool, dan string di Go!
3. Bagaimana Cara mendeklarasikan variabel dengan kata kunci var dan dengan acara singkat? Berikan contoh! Hint “:=”
4. Apa perbedaan antara operator == dan = dalam bahasa Go?
5. Buatlah program Go sederhana untuk meminta input nama kalian, lalu menampilkan nama.

JAWABAN

1. **fmt.Print**
   - Mencetak argumen dengan format default, tanpa newline otomatis. Spasi hanya muncul otomatis di antara argumen non-string; jika keduanya string, disatukan tanpa spasi.
   - Contoh: fmt.Print("A", 1) → A1; fmt.Print("A", "B") → AB.

   fmt.Println
   - Selalu menyisipkan spasi antar argumen dan menambah newline di akhir, sehingga cocok untuk “satu baris per cetak”.
   - Contoh: fmt.Println("A", 1) → "A 1\n"

   fmt.Printf
   - Menggunakan format specifier untuk kontrol presisi dan tata letak; tidak menambah newline otomatis.
   - Specifier umum: %s (string), %d (int), %f (float), %.2f (dua desimal), %t (bool), %T (tipe), %v/%#v (representasi nilai).
   - Contoh: fmt.Printf("Nama: %s, Umur: %d\n", "Rani", 20)

     Kapan memakai:
     Println untuk debug/print cepat per baris.
     Print untuk sambungan tanpa newline.
     Printf saat butuh format terkontrol (kolom, presisi, label).

2. **int**
   - Bilangan bulat bertanda; lebar mengikuti arsitektur (32/64 bit). Zero value 0. Cocok untuk perulangan, penghitung, indeks.
   - Contoh: var n int = 42.

   float64
   - Bilangan pecahan presisi ganda; default pilihan untuk perhitungan desimal; zero value 0. Gunakan Printf untuk mengontrol presisi tampilan.

- Contoh: var pi float64 = 3.14159; fmt.Printf("%.3f\n", pi) → 3.142

bool

- Nilai logika true/false; zero value false. Dipakai di if, for, operator &&, ||, !.
- Contoh: var ok bool = n > 0

string

- Rangkaian byte UTF-8 yang immutable; len(s) mengembalikan jumlah byte, bukan jumlah karakter rune.
- Contoh: var s string = "Halo"; s = s + " Go"

3. var: Dapat dipakai di tingkat paket atau di dalam fungsi; tipe bisa eksplisit atau diinfer dari initializer; boleh tanpa inisialisasi sehingga mendapat zero value.
   := (short declaration): Hanya di dalam fungsi, wajib diinisialisasi, dan minimal satu identifier baru di kiri; tipe diinfer dari ekspresi kanan.

   Contoh :

```
package main

import "fmt"

func main() {
   var nama string = "Farel Juliyandra Restu Hermawan"
   var umur int
   umur = 18

   alamat := "Jakarta"
   tinggi := 185.5

   fmt.Println("Nama:", nama)
   fmt.Println("Umur:", umur)
   fmt.Println("Alamat:", alamat)
   fmt.Println("Tinggi:", tinggi)
}
```

4. **Perbedaan == dan =**
   - = adalah assignment untuk memberi/mengubah nilai variabel.
   - == adalah comparison untuk membandingkan dua nilai dan menghasilkan bool.

5. **Program minta Nama, lalu tampilkan**

```
package main

import "fmt"

func main() {
   var name string
   fmt.Print("Masukkan nama: ")
   fmt.Scan(&name)
   fmt.Println("Halo,", name)
}
```