**LAPORAN PRAKTIKUM  
ALGORITMA PEMROGRAMAN 2**

**MODUL II**

**REVIEW STRUKTUR KONTROL**



**Disusun Oleh :**

Rizkulloh Alpriyansah

2311102142

IF-11-08

**Dosen Pengampu :**

Arif Amrulloh, S.Kom., M.Kom

**PROGRAM STUDI S1 INFORMATIKA**

**FAKULTAS INFORMATIKA**

**UNIVERSITAS TELKOM PURWOKERTO**

**2024**

1. **DASAR TEORI**
2. Setiap file program dalam Golang harus didefinisikan dalam sebuah package. Dalam setiap proyek Go, harus ada setidaknya satu file yang menggunakan package bernama main. File dengan package main ini adalah file yang akan dieksekusi pertama kali saat program dijalankan. Keyword import digunakan untuk menambahkan package lain ke dalam program, sehingga kita dapat memanfaatkan fungsionalitas yang disediakan oleh package tersebut. Salah satu package bawaan Go adalah fmt, yang menawarkan berbagai fungsi untuk kebutuhan input-output (I/O) terkait teks.  
   Untuk menampilkan teks ke layar (seperti di terminal atau CMD), kita dapat menggunakan fungsi fmt.Println(). Mengingat bahwa fungsi fmt.Println() merupakan bagian dari package fmt, package tersebut harus diimpor terlebih dahulu agar bisa digunakan. Fungsi fmt.Println() dapat menerima sejumlah parameter yang tidak terbatas, dan setiap parameter yang diterima akan dipisahkan dengan spasi saat ditampilkan di layar.
3. Struktur kontrol adalah bagian dari bahasa pemrograman yang digunakan untuk menentukan alur eksekusi program. Dalam Go, struktur kontrol dapat dibagi menjadi beberapa kategori utama:

1.Percabangan (Branching):

Percabangan digunakan untuk mengambil keputusan dalam program berdasarkan kondisi tertentu. Go memiliki beberapa bentuk percabangan, termasuk:

if` statement:

Digunakan untuk mengeksekusi blok kode jika kondisi yang diberikan bernilai true. Contoh:

|  |
| --- |
| if x > 0 { |
| fmt.Println("x adalah positif") |
| } else { |
| fmt.Println("x adalah negatif atau nol") |
| } |

- switch statement:

Menyediakan cara yang lebih bersih untuk memilih di antara beberapa kemungkinan berdasarkan nilai tertentu. Contoh:

|  |
| --- |
| switch day { |
| case "Senin": |
| fmt.Println("Hari kerja") |
| case "Sabtu", "Minggu": |
| fmt.Println("Akhir pekan") |
| default: |
| fmt.Println("Hari biasa") |
| } |

2. Perulangan (Looping):

Perulangan digunakan untuk mengeksekusi blok kode berulang kali selama kondisi tertentu terpenuhi. Dalam Go, terdapat dua jenis perulangan:

- for loop:

Merupakan satu-satunya pernyataan perulangan dalam Go. `for` dapat digunakan dalam beberapa cara:

- Dengan inisialisasi, kondisi, dan pernyataan akhir.

- Hanya dengan kondisi.

- Sebagai perulangan tak terbatas.

Contoh:

|  |
| --- |
| for i := 0; i < 5; i++ { |
| fmt.Println(i) |
| } |

- range loop:

Digunakan untuk mengiterasi elemen dari array, slice, string, map, atau channel.

Contoh:

|  |
| --- |
| for index, value := range slice { |
| fmt.Println(index, value) |
| } |

3. Pengendalian Alur (Control Flow):

Struktur kontrol ini digunakan untuk mengubah alur eksekusi program secara langsung. Terdapat beberapa cara untuk mengendalikan alur dalam Go:

- break statement:

Menghentikan perulangan atau switch case dan melanjutkan eksekusi setelah blok tersebut.

|  |
| --- |
| for i := 0; i < 10; i++ { |
| if i == 5 { |
| break |
| } |
| fmt.Println(i) |
| } |
|  |

- continue statement:

Melewatkan iterasi saat ini dan melanjutkan ke iterasi berikutnya dalam perulangan.

|  |
| --- |
| for i := 0; i < 10; i++ { |
| if i%2 == 0 { |
| continue |
| } |
| fmt.Println(i) |
| } |
|  |

- goto statement:

Memindahkan alur eksekusi program ke label yang ditentukan. Namun, penggunaan `goto` tidak disarankan karena dapat membuat kode sulit dibaca.

|  |
| --- |
| goto label |
| label: |
| fmt.Println("Ini adalah label") |

Kesimpulan

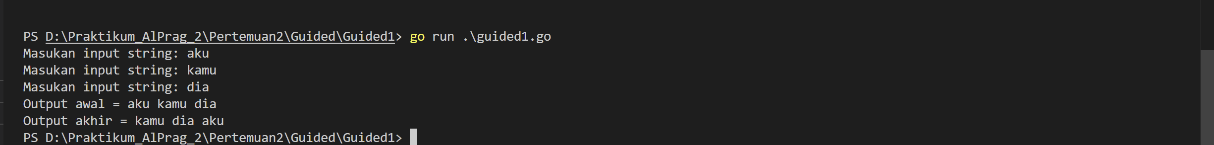
Struktur kontrol dalam Go memudahkan pengembang untuk mengatur alur eksekusi program. Dengan memahami dan menggunakan berbagai bentuk percabangan dan perulangan, programmer dapat membuat program yang lebih dinamis dan responsif terhadap kondisi dan input yang berbeda. Penggunaan struktur kontrol yang tepat juga dapat meningkatkan keterbacaan dan maintainability dari kode yang ditulis.

1. **GUIDED**

**Guided 1**

|  |
| --- |
| 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) |
| } |

**Screenshots Output**



**Deskripsi:**

Program di atas meminta tiga input string dari pengguna, lalu mencetak output awal berupa ketiga string tersebut dalam urutan yang diinputkan. Setelah itu, program menukar urutannya dengan cara memindahkan nilai dari string pertama ke string kedua, string kedua ke string ketiga, dan string ketiga ke string pertama. Akhirnya, program mencetak output akhir dengan urutan string yang sudah diubah.

**Guided 2**

|  |
| --- |
| package main |
|  |
| import "fmt" |
|  |
| func main() { |
| var tahun int |
| fmt.Print("Tahun: ") |
| fmt.Scanln(&tahun) |
|  |
| kabisat := cekKabisat(tahun) |
| fmt.Println("Kabisat: ", kabisat) |
| } |
|  |
| func cekKabisat(tahun int) bool { |
| if tahun%400 == 0 { |
| return true |
| } else if tahun%4 == 0 { |
| return true |
| } else if tahun%100 == 0 { |
| return false |
| } |
| return false |
| } |

**Screenshots Output**

A black screen with white text

Description automatically generated

**Deskripsi:**

Program di atas digunakan untuk mengecek apakah suatu tahun adalah tahun kabisat atau bukan. Alurnya:

1. Pengguna diminta untuk memasukkan nilai “tahun”.

2. Program memanggil fungsi “cekKabisat(tahun)” yang akan mengevaluasi apakah tahun tersebut memenuhi syarat sebagai tahun kabisat.

3. Di dalam fungsi “cekKabisat”, tahun kabisat didefinisikan dengan kondisi:

- Tahun habis dibagi 400 (`tahun%400 == 0`), maka tahun tersebut kabisat.

- Jika tidak, tetapi habis dibagi 4 (`tahun%4 == 0`), maka juga kabisat.

- Namun, jika habis dibagi 100 (`tahun%100 == 0`), maka bukan tahun kabisat.

- Selain kondisi di atas, tahun dianggap bukan tahun kabisat.

4. Hasil dari fungsi “cekKabisat” ditampilkan di konsol, menunjukkan apakah `tahun` tersebut kabisat (`true`) atau tidak (`false`).

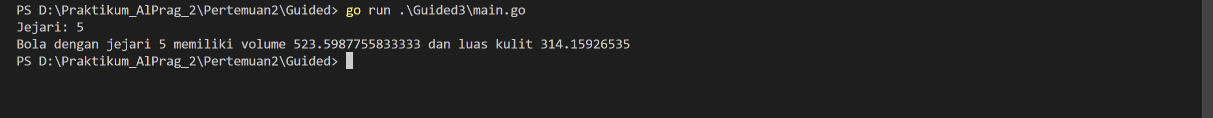
Kondisi yang harus diperbaiki: Urutan pengecekan `tahun%100 == 0` harus didahulukan dari `tahun%4 == 0`.

Jika tidak, tahun yang habis dibagi 100 dan 4 akan salah terdeteksi sebagai kabisat.

**Guided 3**

|  |
| --- |
| package main |
|  |
| import ( |
| "fmt" |
| "math" |
| ) |
|  |
| func main() { |
| var jejari int |
| fmt.Print("Jejari: ") |
| fmt.Scanf("%d", &jejari) |
|  |
| volume := hitungVolumeBola(float64(jejari)) |
| luas := hitungLuasKulitBola(float64(jejari)) |
|  |
| fmt.Println("Bola dengan jejari", jejari, "memiliki volume", volume, "dan luas kulit", luas) |
| } |
|  |
| func hitungVolumeBola(jejari float64) float64 { |
| const pi = 3.1415926535 |
| return (4.0 / 3.0) \* pi \* math.Pow(jejari, 3) |
| } |
|  |
| func hitungLuasKulitBola(jejari float64) float64 { |
| const pi = 3.1415926535 |
| return 4 \* pi \* math.Pow(jejari, 2) |
| } |

**Screenshots Output**



**Deskripsi:**

Program ini meminta pengguna memasukkan jejari bola, lalu menghitung volume dan luas permukaan bola. Volume dihitung dengan rumus \( \frac{4}{3} \times \pi \times jejari^3 \), dan luas permukaan dengan rumus \( 4 \times \pi \times jejari^2 \), menggunakan konstanta `pi` dan fungsi `math.Pow` untuk operasi pangkat. Hasilnya berupa volume dan luas kulit bola yang ditampilkan berdasarkan jejari yang diinputkan.

**Guided 4**

|  |
| --- |
| package main |
|  |
| import ( |
| "fmt" |
| ) |
|  |
| func main() { |
| // Input suhu dalam Celsius |
| var celsius float64 |
| fmt.Print("Masukkan suhu dalam derajat Celsius: ") |
| fmt.Scan(&celsius) |
|  |
| // Konversi ke Fahrenheit |
| fahrenheit := (celsius \* 9 / 5) + 32 |
|  |
| // Konversi ke Reamur |
| reamur := celsius \* 4 / 5 |
|  |
| // Konversi ke Kelvin |
| kelvin := celsius + 273.15 |
|  |
| // Output hasil konversi |
| fmt.Printf("Temperatur Celsius: %.2f\n", celsius) |
| fmt.Printf("Derajat Reamur: %.2f\n", reamur) |
| fmt.Printf("Derajat Fahrenheit: %.2f\n", fahrenheit) |
| fmt.Printf("Derajat Kelvin: %.2f\n", kelvin) |
| } |

**Screenshots Output**

A close-up of a flag

Description automatically generated

**Deskripsi:**

Program ini mengubah suhu dari Celsius ke Fahrenheit, Reamur, dan Kelvin. Setelah pengguna memasukkan suhu dalam Celsius, program menghitung Fahrenheit dengan rumus \( F = \frac{9}{5} \times C + 32 \), Reamur dengan \( R = \frac{4}{5} \times C \), dan Kelvin dengan \( K = C + 273.15 \), lalu menampilkan hasil konversi dalam format desimal dua angka.

**Guided 5**

|  |
| --- |
| //By Rizkulloh NIM 2311102142 |
| package main |
|  |
| import ( |
| "fmt" |
| ) |
|  |
| func main() { |
| // Inisialisasi array untuk menampung 5 integer dan 3 karakter |
| var nums [5]int |
| var chars [3]rune |
|  |
| // Meminta input untuk 5 data integer |
| fmt.Println("Masukkan 5 angka integer (32-127):") |
| for i := 0; i < 5; i++ { |
| fmt.Scanf("%d", &nums[i]) |
| } |
|  |
| // Meminta input untuk 3 karakter |
| fmt.Println("Masukkan 3 karakter:") |
| for i := 0; i < 3; i++ { |
| fmt.Scanf("%c", &chars[i]) |
| // Mengabaikan newline dari input sebelumnya |
| if chars[i] == '\n' { |
| i-- |
| } |
| } |
|  |
| // Output pertama: 5 integer yang dikonversi menjadi karakter ASCII |
| fmt.Println("Keluaran:") |
| for i := 0; i < 5; i++ { |
| fmt.Printf("%c", nums[i]) |
| } |
| fmt.Println() |
|  |
| // Output kedua: 3 karakter yang diinputkan |
| for i := 0; i < 3; i++ { |
| fmt.Printf("%c", chars[i]) |
| } |
| fmt.Println() |
| } |

**Screenshots Output**

**A black background with white dots

Description automatically generated**

**Deskripsi:**

Program ini meminta pengguna menginput 5 angka integer (32-127) yang kemudian dikonversi menjadi karakter ASCII, serta 3 karakter tambahan. Setelah semua input diterima, program menampilkan dua output: angka-angka yang diinput ditampilkan sebagai karakter ASCII, dan karakter-karakter yang diinput ditampilkan secara langsung. Program juga mengabaikan newline agar tidak mempengaruhi input karakter.

1. **UNGUIDED**

**Unguided 1**

|  |
| --- |
| //By Rizkulloh NIM 2311102142  package main  import (      "bufio"      "fmt"      "os"      "strings"  )  func main() {      // Urutan warna yang benar      correctOrder\_142 := []string{"merah", "kuning", "hijau", "ungu"}      // Membaca input untuk 5 percobaan      reader\_142 := bufio.NewReader(os.Stdin)      success\_142 := true      for i\_142 := 1; i\_142 <= 5; i\_142++ {          fmt.Printf("Percobaan %d: ", i\_142)          // Membaca input dari pengguna          input\_142, \_ := reader\_142.ReadString('\n')          input\_142 = strings.TrimSpace(input\_142)          // Memisahkan input berdasarkan spasi          colors\_142 := strings.Split(input\_142, " ")          // Mengecek apakah urutan warna sesuai          for j\_142 := 0; j\_142 < 4; j\_142++ {              if colors\_142[j\_142] != correctOrder\_142[j\_142] {                  success\_142 = false                  break              }          }          // Jika ada percobaan yang tidak sesuai, keluar dari loop          if !success\_142 {              break          }      }      // Menampilkan hasil      if success\_142 {          fmt.Println("BERHASIL : true")      } else {          fmt.Println("BERHASIL : false")      }  } |

**Screenshots Output**

**A black screen with white text

Description automatically generated**

**Deskripsi:**

Program ini mengecek apakah urutan warna yang dimasukkan pengguna sesuai dengan urutan yang benar, yaitu "merah", "kuning", "hijau", dan "ungu". Pengguna diminta untuk memasukkan urutan warna sebanyak 5 kali, yang kemudian dibaca dan dipecah menjadi array `colors\_142`. Program membandingkan setiap warna yang dimasukkan dengan urutan yang benar dan mengubah variabel `success\_142` menjadi `false` jika ada warna yang tidak sesuai, sehingga proses percobaan berhenti. Pada akhir, jika semua percobaan berhasil, program menampilkan "BERHASIL : true"; jika ada kesalahan, ditampilkan "BERHASIL : false". Semua variabel dalam program diberi akhiran `\_142` sesuai permintaan.

**Unguided 2**

|  |
| --- |
| //By Rizkulloh NIM 2311102142  package main  import (      "bufio"      "fmt"      "os"      "strconv"      "strings"  )  func main() {      // Membuat reader untuk membaca input dari pengguna      reader\_142 := bufio.NewReader(os.Stdin)      // Meminta input jumlah bunga yang akan dimasukkan (bilangan bulat positif N)      fmt.Print("N: ")      var N\_142 int      for {          // Baca input dari pengguna          input\_142, err := reader\_142.ReadString('\n')          if err != nil {              fmt.Printf("Error membaca input: %v\n", err)              return          }          // Konversi input ke integer          N\_142, err = strconv.Atoi(strings.TrimSpace(input\_142))          if err != nil || N\_142 <= 0 {              fmt.Println("Harap masukkan bilangan bulat positif.")          } else {              break          }      }      // Inisialisasi variabel pita (string) untuk menyimpan nama bunga      var pita\_142 string      var count\_142 int // Menyimpan jumlah bunga yang dimasukkan      // Loop untuk menerima input nama bunga sebanyak N kali      for i\_142 := 1; i\_142 <= N\_142; i\_142++ {          fmt.Printf("Bunga %d: ", i\_142) // Menambahkan instruksi          // Membaca input dari pengguna          input\_142, err := reader\_142.ReadString('\n')          if err != nil {              fmt.Printf("Error: %v\n", err)              return // Keluar dari program jika ada kesalahan          }          // Menghapus spasi dan karakter newline dari input          input\_142 = strings.TrimSpace(input\_142)          // Cek jika pengguna mengetik "SELESAI"          if strings.ToUpper(input\_142) == "SELESAI" {              break // Menghentikan input jika "SELESAI" dimasukkan          }          // Menggabungkan nama bunga dengan pita menggunakan " – " sebagai pemisah          if pita\_142 == "" {              pita\_142 = input\_142 // Jika pita masih kosong, langsung masukkan nama bunga          } else {              pita\_142 = pita\_142 + " – " + input\_142 // Jika sudah ada isinya, tambahkan dengan pemisah " – "          }          count\_142++ // Menambah jumlah bunga yang dimasukkan      }      // Menampilkan isi pita dan jumlah bunga setelah semua input dimasukkan      fmt.Println("Pita:", pita\_142)      fmt.Printf("Bunga: %d\n", count\_142)  } |

**Screenshots Output**

**A black screen with white text

Description automatically generated**

**Deskripsi:**

Program ini meminta pengguna untuk memasukkan jumlah bunga yang ingin diinput (bilangan bulat positif) dan kemudian mengumpulkan nama-nama bunga sebanyak jumlah yang ditentukan. Setelah pengguna memasukkan jumlah bunga, program membaca nama bunga satu per satu, dan jika pengguna mengetik "SELESAI", input akan dihentikan. Nama bunga yang dimasukkan akan digabungkan dalam satu string bernama `pita\_142`, dengan pemisah " – " antara setiap nama bunga. Program juga menghitung jumlah bunga yang dimasukkan dengan variabel `count\_142`. Akhirnya, program menampilkan isi dari pita bunga dan jumlah total bunga yang dimasukkan.

**Unguided 3**

|  |
| --- |
| //By Rizkulloh NIM 2311102142  package main  import (      "fmt"  )  func main() {      var beratKantongKiri\_142, beratKantongKanan\_142 float64      for {          // Menerima input berat kedua kantong dalam satu baris          fmt.Print("Masukan berat belanjaan di kedua kantong (kg): ")          fmt.Scan(&beratKantongKiri\_142, &beratKantongKanan\_142)          // Cek jika salah satu kantong mencapai atau lebih dari 9 kg          if beratKantongKiri\_142 >= 9 || beratKantongKanan\_142 >= 9 {              fmt.Println("Proses selesai.")              break          }          // Hitung selisih berat antara kantong kiri dan kanan          selisihBerat\_142 := beratKantongKiri\_142 - beratKantongKanan\_142          if selisihBerat\_142 < 0 {              selisihBerat\_142 = -selisihBerat\_142          }          // Cek apakah selisih berat melebihi 9 kg          if selisihBerat\_142 > 9 {              fmt.Println("Selisih berat antara kantong kiri dan kanan melebihi 9 kg.")          }      }  } |

**Screenshots Output**

A black background with white text

Description automatically generated

**Deskripsi:**

Program ini meminta pengguna untuk memasukkan berat dari dua kantong belanja. Variabel beratKantongKiri\_142 dan beratKantongKanan\_142 digunakan untuk menyimpan berat masing-masing kantong. Program akan memeriksa apakah salah satu dari kantong tersebut memiliki berat 9 kg atau lebih. Jika iya, proses akan dihentikan dan program mencetak "Proses selesai." Jika tidak, program akan menghitung selisih berat antara kedua kantong dan menyimpannya di variabel selisihBerat\_142. Jika selisih berat ini lebih dari 9 kg, program akan menampilkan pesan yang menyatakan bahwa selisih berat antara kantong kiri dan kanan melebihi 9 kg.

**Setelah di modif**

|  |
| --- |
| package main |
|  |
| import ( |
| "fmt" |
| ) |
|  |
| func main() { |
| var beratKantongKiri\_142, beratKantongKanan\_142 float64 |
| for { |
| // Menerima input berat kedua kantong dalam satu baris |
| fmt.Print("Masukan berat belanjaan di kedua kantong (kg): ") |
| fmt.Scan(&beratKantongKiri\_142, &beratKantongKanan\_142) |
|  |
| // Cek jika salah satu kantong memiliki berat negatif |
| if beratKantongKiri\_142 < 0 || beratKantongKanan\_142 < 0 { |
| fmt.Println("Proses selesai.") |
| break |
| } |
|  |
| // Cek jika total berat kedua kantong melebihi 150 kg |
| totalBerat\_142 := beratKantongKiri\_142 + beratKantongKanan\_142 |
| if totalBerat\_142 > 150 { |
| fmt.Println("Proses selesai.") |
| break |
| } |
|  |
| // Hitung selisih berat antara kantong kiri dan kanan |
| selisihBerat\_142 := beratKantongKiri\_142 - beratKantongKanan\_142 |
| if selisihBerat\_142 < 0 { |
| selisihBerat\_142 = -selisihBerat\_142 |
| } |
|  |
| // Menampilkan hasil apakah sepeda motor akan oleng atau tidak |
| if selisihBerat\_142 >= 9 { |
| fmt.Println("Sepeda motor Pak Andi akan oleng: true") |
| } else { |
| fmt.Println("Sepeda motor Pak Andi akan oleng: false") |
| } |
| } |
| } |

**Output**

**A computer screen shot of a black screen

Description automatically generated**

**Deskripsi**

Program ini berfungsi untuk memantau berat dua kantong belanjaan. Pengguna diminta untuk memasukkan berat dari kantong kiri (beratKantongKiri\_142) dan kanan (beratKantongKanan\_142). Program akan terus meminta input selama kedua berat tidak negatif dan total berat dari kedua kantong tidak melebihi 150 kg. Jika salah satu kantong memiliki berat negatif atau totalnya lebih dari 150 kg, program akan mencetak "Proses selesai." dan keluar dari loop. Selanjutnya, program menghitung selisih berat antara kantong kiri dan kanan, disimpan dalam variabel selisihBerat\_142. Jika selisih berat lebih dari atau sama dengan 9 kg, program akan menampilkan bahwa sepeda motor Pak Andi akan oleng; jika tidak, program akan menampilkan bahwa sepeda motor tidak akan oleng.

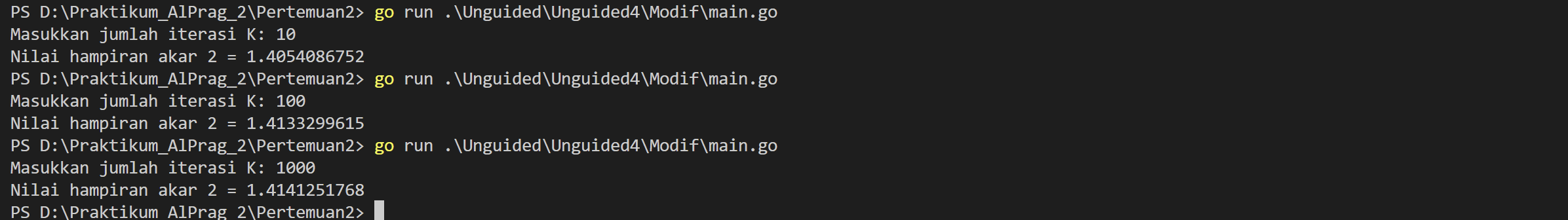
**Unguided 4**

**Setelah DiModif**

**Source code**

|  |
| --- |
| package main |
|  |
| import ( |
| "fmt" |
| ) |
|  |
| // Fungsi untuk menghitung nilai hampiran akar 2 berdasarkan iterasi K |
| func hitungAkar2(iterasi\_142 int) float64 { |
| hasilPerkalian\_142 := 1.0 // Mulai dengan nilai produk 1 |
|  |
| // Iterasi dari 0 hingga iterasi yang dimasukkan |
| for indeks\_142 := 0; indeks\_142 < iterasi\_142; indeks\_142++ { // Menggunakan < bukan <= |
| pembilang\_142 := (4\*float64(indeks\_142) + 2) \* (4\*float64(indeks\_142) + 2) // (4k + 2)^2 |
| penyebut\_142 := (4\*float64(indeks\_142) + 1) \* (4\*float64(indeks\_142) + 3) // (4k + 1)(4k + 3) |
| hasilPerkalian\_142 \*= pembilang\_142 / penyebut\_142 |
| } |
| return hasilPerkalian\_142 // Mengembalikan nilai hasil perkalian |
| } |
|  |
| func main() { |
| var jumlahIterasi\_142 int |
|  |
| // Membaca input nilai jumlah iterasi |
| fmt.Print("Masukkan jumlah iterasi K: ") |
| fmt.Scan(&jumlahIterasi\_142) |
|  |
| // Memastikan jumlah iterasi tidak negatif |
| if jumlahIterasi\_142 < 0 { |
| fmt.Println("Jumlah iterasi K tidak boleh negatif.") |
| return |
| } |
|  |
| // Menghitung nilai hampiran sqrt(2) |
| hasilAkar2\_142 := hitungAkar2(jumlahIterasi\_142) |
|  |
| // Menampilkan hasil dengan 10 angka di belakang koma |
| fmt.Printf("Nilai hampiran akar 2 = %.10f\n", hasilAkar2\_142) |
| } |

**Output**

****

**Deskripsi**

Program ini menghitung nilai hampiran akar dua (√2) menggunakan metode iterasi berdasarkan rumus tertentu. Fungsi hitungAkar2 menerima parameter iterasi\_142, yang menentukan berapa kali perhitungan dilakukan. Di dalam fungsi ini, hasilPerkalian\_142 diinisialisasi dengan nilai 1.0, dan perulangan dilakukan dari 0 hingga jumlah iterasi yang ditentukan. Dalam setiap iterasi, pembilang dihitung dengan rumus (4k+2)2(4k + 2)^2(4k+2)2 dan penyebut dengan rumus (4k+1)(4k+3)(4k + 1)(4k + 3)(4k+1)(4k+3), lalu hasil pembagian pembilang dengan penyebut dikalikan dengan hasilPerkalian\_142. Setelah semua iterasi selesai, fungsi mengembalikan nilai akhir dari hasilPerkalian\_142. Di dalam fungsi main, program meminta pengguna untuk memasukkan jumlah iterasi, memeriksa apakah nilai tersebut negatif, dan jika valid, akan memanggil fungsi untuk menghitung dan menampilkan **hasil hampiran akar dua hingga 10 angka di belakang koma.**

**Unguided 5**

|  |
| --- |
| //By Rikulloh NIM 2311102142  package main  import (      "fmt"  )  func hitungBiayaKirim(berat\_142 int) int {      // Menghitung berat dalam kilogram dan gram      kg\_142 := berat\_142 / 1000      gram\_142 := berat\_142 % 1000      // Biaya pengiriman per kilogram      biayaPerKg\_142 := 10000      biayaTotal\_142 := kg\_142 \* biayaPerKg\_142      // Biaya tambahan untuk sisa gram      biayaTambahan\_142 := 0      if kg\_142 >= 10 {          biayaTambahan\_142 = 0 // Gratis biaya tambahan jika berat lebih dari 10 kg      } else {          if gram\_142 >= 500 {              biayaTambahan\_142 = gram\_142 \* 5 // Rp. 5 per gram jika sisa >= 500 gram          } else {              biayaTambahan\_142 = gram\_142 \* 15 // Rp. 15 per gram jika sisa < 500 gram          }      }      // Total biaya      return biayaTotal\_142 + biayaTambahan\_142  }  func main() {      var berat\_142 int      // Meminta input berat dari pengguna      fmt.Print("Berat parsel (gram): ")      fmt.Scan(&berat\_142)      // Menghitung berat dalam kg dan gram      kg\_142 := berat\_142 / 1000      gram\_142 := berat\_142 % 1000      // Menghitung total biaya pengiriman      biayaPerKg\_142 := 10000 \* kg\_142      biayaTambahan\_142 := 0      // Kondisi untuk biaya tambahan berdasarkan sisa berat gram      if kg\_142 >= 10 {          biayaTambahan\_142 = 0 // Gratis biaya tambahan jika lebih dari 10 kg      } else {          if gram\_142 >= 500 {              biayaTambahan\_142 = gram\_142 \* 5 // Rp. 5 per gram untuk sisa >= 500 gram          } else {              biayaTambahan\_142 = gram\_142 \* 15 // Rp. 15 per gram untuk sisa < 500 gram          }      }      // Menghitung total biaya      totalBiaya\_142 := biayaPerKg\_142 + biayaTambahan\_142      // Menampilkan hasil      fmt.Printf("Detail berat: %d kg + %d gr\n", kg\_142, gram\_142)      fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", biayaPerKg\_142, biayaTambahan\_142)      fmt.Printf("Total biaya: Rp. %d\n", totalBiaya\_142)  } |

**Screenshots Output**

**A black screen with white text

Description automatically generated**

**Deskripsi:**

Program ini menghitung biaya pengiriman parsel berdasarkan beratnya yang diinput dalam gram. Fungsi hitungBiayaKirim menerima parameter berat\_142 untuk menghitung berat dalam kilogram (kg\_142) dan gram (gram\_142). Biaya pengiriman per kilogram ditetapkan Rp. 10.000, dan biaya tambahan dihitung berdasarkan sisa berat gram: jika berat mencapai 10 kg, biaya tambahan gratis; jika sisa lebih dari atau sama dengan 500 gram, biaya tambahan Rp. 5 per gram; jika kurang dari 500 gram, Rp. 15 per gram. Di dalam fungsi main, pengguna diminta untuk memasukkan berat parsel, dan total biaya ditampilkan bersama dengan rincian berat dan biaya.

**Unguided 6**

|  |
| --- |
| //By Rizkulloh NIM 2311102142  package main  import "fmt"  func main() {      var nam float64      var nmk string      // Meminta input nilai      fmt.Print("Nilai akhir mata kuliah: ")      fmt.Scan(&nam)      // Logika penentuan nilai huruf berdasarkan nilai numerik      if nam > 80 {          nmk = "A"      } else if nam > 72.5 {          nmk = "AB"      } else if nam > 65 {          nmk = "B"      } else if nam > 57.5 {          nmk = "BC"      } else if nam > 50 {          nmk = "C"      } else if nam > 40 {          nmk = "D"      } else {          nmk = "E"      }      // Menampilkan hasil      fmt.Printf("Nilai mata kuliah: %s\n", nmk)  } |

**Screenshots Output**

* 1. ****
  2. **Apa saja kesalahan dari program tersebut? Mengapa demikian? Jelaskan alur program seharusnya!**

Kesalahan dalam Program: Variabel nam = "B" seharusnya ditulis sebagai nmk = "B"

Dalam kode ini, semua kondisi menggunakan if tanpa else if untuk rentang nilai yang saling tumpang tindih. Dengan cara ini, jika nilai nam lebih dari 80, maka nilai nmk akan ditetapkan ke "A", tetapi program akan tetap melanjutkan pemeriksaan kondisi lain. Hal ini tidak efisien dan dapat menghasilkan output yang tidak diinginkan.

Pada pernyataan fmt.Printf, format string tidak digunakan. Seharusnya ada format placeholder untuk variabel nmk.

Dalam kondisi penentuan nilai E, menggunakan else if setelah kondisi if biasa. Sebaiknya gunakan else tanpa syarat sebelumnya.

Alur Program seharusnya:

Meminta input

Menentukan nilai huruf:

Jika nam lebih dari 80, maka nmk menjadi "A".

Jika nam lebih dari 72.5 dan kurang dari atau sama dengan 80, maka nmk menjadi "AB".

Jika nam lebih dari 65 dan kurang dari atau sama dengan 72.5, maka nmk menjadi "B".

Jika nam lebih dari 57.5 dan kurang dari atau sama dengan 65, maka nmk menjadi "BC".

Jika nam lebih dari 50 dan kurang dari atau sama dengan 57.5, maka nmk menjadi "C".

Jika nam lebih dari 40 dan kurang dari atau sama dengan 50, maka nmk menjadi "D".

Jika nam kurang dari atau sama dengan 40, maka nmk menjadi "E".

1. package main
2. import "fmt"
3. func main() {
4. var nam\_142 float64
5. var nmk\_142 string
6. // Meminta input nilai
7. fmt.Print("Nilai akhir mata kuliah: ")
8. fmt.Scan(&nam\_142)
9. // Logika penentuan nilai huruf berdasarkan nilai numerik
10. if nam\_142 > 80 {
11. nmk\_142 = "A"
12. } else if nam\_142 > 72.5 {
13. nmk\_142 = "AB"
14. } else if nam\_142 > 65 {
15. nmk\_142 = "B"
16. } else if nam\_142 > 57.5 {
17. nmk\_142 = "BC"
18. } else if nam\_142 > 50 {
19. nmk\_142 = "C"
20. } else if nam\_142 > 40 {
21. nmk\_142 = "D"
22. } else {
23. nmk\_142 = "E"
24. }
25. // Menampilkan hasil
26. fmt.Printf("Nilai Indeks untuk nilai akhir mata kuliah: %s\n", nmk\_142)
27. }

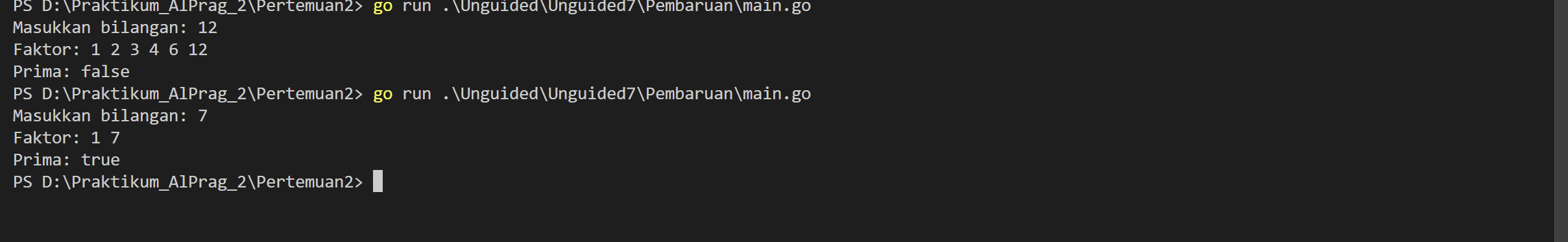
**Unguided 7**

**Setelah Pembaruan**

**Source Code**

|  |
| --- |
| package main |
|  |
| import ( |
| "fmt" |
| ) |
|  |
| func main() { |
| var number\_142 int |
|  |
| // Input bilangan bulat |
| fmt.Print("Masukkan bilangan: ") |
| fmt.Scan(&number\_142) |
|  |
| // Mencari dan menampilkan semua faktor dari bilangan |
| fmt.Printf("Faktor: ") |
| factors\_142 := findFactors(number\_142) |
|  |
| // Menampilkan faktor-faktor |
| for \_, factor\_142 := range factors\_142 { |
| fmt.Printf("%d ", factor\_142) |
| } |
| fmt.Println() |
|  |
| // Mengecek apakah bilangan tersebut prima |
| if len(factors\_142) == 2 { |
| fmt.Println("Prima: true") |
| } else { |
| fmt.Println("Prima: false") |
| } |
| } |
|  |
| // Fungsi untuk mencari faktor dari bilangan |
| func findFactors(n\_142 int) []int { |
| var factors\_142 []int |
| for i\_142 := 1; i\_142 <= n\_142; i\_142++ { |
| if n\_142%i\_142 == 0 { |
| factors\_142 = append(factors\_142, i\_142) |
| } |
| } |
| return factors\_142 |
| } |

**Output**



**Deskripsi**

Program ini meminta pengguna untuk memasukkan sebuah bilangan bulat dan kemudian mencari serta menampilkan semua faktor dari bilangan tersebut. Setelah menerima input, program menggunakan fungsi findFactors untuk mendapatkan faktor-faktor dari bilangan yang dimasukkan dan menyimpannya dalam slice factors\_142. Setelah itu, program mencetak semua faktor yang ditemukan. Selanjutnya, program memeriksa apakah jumlah faktor tersebut sama dengan 2, yang menunjukkan bahwa bilangan tersebut adalah bilangan prima (hanya memiliki faktor 1 dan dirinya sendiri), dan mencetak hasilnya.