

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL 2
REVIEW STRUKTUR KONTROL**



Oleh:

ADINDA OLIVIA

2311102245

IF-11-07

**S1 TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI TELKOM PURWOKERTO**

2024

I. DASAR TEORI

Struktur kontrol adalah elemen fundamental dalam pemrograman yang memungkinkan pengembang mengatur alur eksekusi kode berdasarkan kondisi tertentu. Dengan menggunakan struktur kontrol, programmer dapat membuat keputusan, melakukan perulangan, dan mengendalikan aliran program sesuai dengan logika yang diinginkan. Pemahaman yang baik tentang struktur kontrol sangat penting untuk membangun program yang efisien dan responsif. Dalam berbagai bahasa pemrograman, termasuk Go, Java, dan Python, terdapat beberapa jenis struktur kontrol yang umum digunakan, yang dapat dibedakan menjadi struktur percabangan dan struktur perulangan.

Struktur kontrol percabangan memungkinkan program untuk mengambil keputusan berdasarkan kondisi tertentu. If statement adalah salah satu bentuk paling dasar dari percabangan, di mana sebuah blok kode dieksekusi jika kondisi yang diberikan benar. Misalnya, jika nilai suatu variabel lebih besar dari 10, maka program dapat menjalankan serangkaian instruksi tertentu. Untuk memberikan alternatif, digunakan if-else statement, yang memungkinkan program untuk menjalankan blok kode yang berbeda jika kondisi tidak terpenuhi. Selain itu, switch statement juga sering digunakan untuk memeriksa nilai dari suatu variabel dan menentukan blok kode mana yang harus dijalankan, menjadikannya lebih terstruktur dan mudah dibaca dibandingkan dengan penggunaan beberapa if-else yang bersarang.

Di sisi lain, struktur kontrol perulangan memungkinkan kita untuk menjalankan blok kode berulang kali berdasarkan kondisi yang ditentukan. For loop adalah jenis perulangan yang paling umum, digunakan untuk mengulangi blok kode dengan jumlah iterasi yang diketahui. Contohnya, jika kita ingin mencetak angka dari 1 hingga 10, kita dapat menggunakan for loop dengan batasan iterasi. While loop juga digunakan untuk perulangan, tetapi dengan cara yang sedikit berbeda: blok kode akan terus dijalankan selama kondisi tertentu terpenuhi. Misalnya, jika kita ingin membaca input dari pengguna hingga mereka memasukkan angka negatif, kita bisa menggunakan while loop. Do-while loop berfungsi mirip dengan while loop, tetapi memastikan bahwa blok kode dijalankan setidaknya sekali, karena kondisi diperiksa setelah eksekusi blok.

Struktur kontrol juga memungkinkan penggunaan perulangan bersarang, di mana kita dapat menempatkan satu loop di dalam loop lainnya. Ini sangat berguna dalam situasi di mana kita ingin melakukan operasi yang melibatkan beberapa dimensi, seperti pengolahan data dalam bentuk matriks atau array dua dimensi. Misalnya, ketika kita ingin mencetak tabel, kita dapat menggunakan loop luar untuk iterasi baris dan loop dalam untuk iterasi kolom, sehingga menghasilkan output yang terstruktur dengan baik. Dengan memahami dan menerapkan perulangan bersarang, programmer dapat mengelola data yang lebih kompleks dengan lebih efektif.

Secara keseluruhan, pemahaman yang mendalam tentang struktur kontrol sangat penting bagi setiap programmer. Dengan menggunakan struktur kontrol yang tepat, pengembang dapat membangun logika program yang efisien, memudahkan pengambilan keputusan, dan memastikan bahwa program berfungsi sesuai dengan harapan. Dengan menguasai berbagai jenis struktur kontrol, mulai dari percabangan hingga perulangan, programmer dapat menciptakan aplikasi yang lebih responsif dan mudah dipelihara. Struktur kontrol tidak hanya membantu dalam menyusun logika program, tetapi juga berkontribusi pada peningkatan performa dan keterbacaan kode, yang sangat penting dalam pengembangan perangkat lunak.

II. GUIDED

2a. 1.

```
package main





import "fmt"

func main() {
    var satu, dua, tiga string
    var 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)

    // Menggabungkan string
    temp = satu + " " + dua + " " + dua + " " + tiga
    tiga = temp + dua + " " + tiga

    fmt.Println("Output akhir:", satu, dua, tiga)
}
```

main.go	  Share  Run  Clear
<pre>1 package main 2 3 import "fmt" 4 5 func main() { 6 var satu, dua, tiga string 7 var temp string 8 9 fmt.Print("Masukan input string: ") 10 fmt.Scanln(&satu) 11 fmt.Print("Masukan input string: ") 12 fmt.Scanln(&dua) 13 fmt.Print("Masukan input string: ") 14 fmt.Scanln(&tiga) 15 16 // Menggabungkan string 17 temp = satu + " " + dua + " " + dua + " " + tiga 18 tiga = temp + dua + " " + tiga 19 20 fmt.Println("Output akhir:", satu, dua, tiga) 21 } 22</pre>	<pre>go run /tmp/3T1Jy0DCJr.go Masukan input string: v Masukan input string: i Masukan input string: a Output akhir: v i v i i a i a</pre>

Penjelasan: Program ini meminta pengguna untuk memasukkan tiga string secara berurutan dan kemudian menggabungkan string tersebut dengan format tertentu. Setelah pengguna memasukkan input, program menggabungkan string pertama, kedua, dan kedua lagi, diikuti oleh string ketiga untuk menghasilkan nilai sementara yang disimpan di variabel temp. Selanjutnya, variabel tiga diperbarui dengan hasil gabungan dari temp, dua, dan tiga. Terakhir, program mencetak output akhir yang menunjukkan ketiga string yang telah dimodifikasi.

2.

```
package main

import "fmt"

func main() {

    var tahun int

    fmt.Print("Tahun: ")

    fmt.Scanln(&tahun)

    kabisat := false

    // Memeriksa tahun kabisat
```

```

        if (tahun%400 == 0) || (tahun%4 == 0 && tahun%100 !=
0) {

            kabisat = true

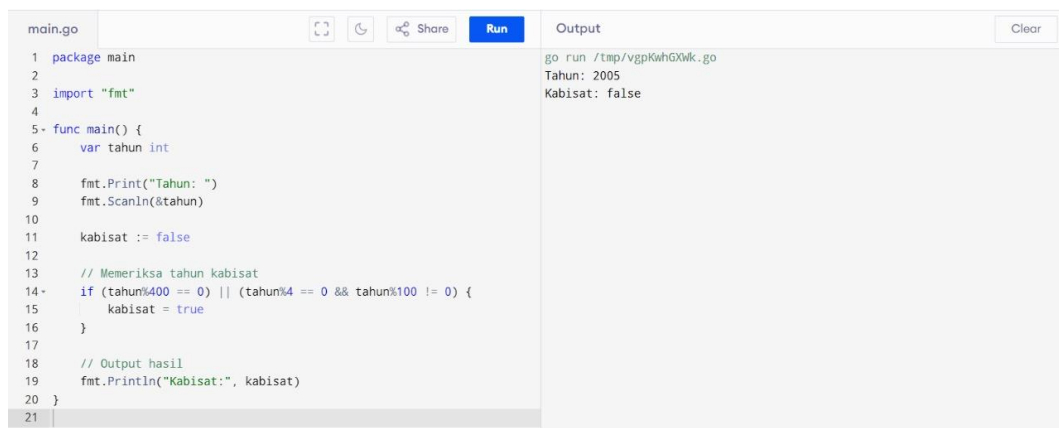
        }

        // Output hasil

        fmt.Println("Kabisat:", kabisat)

    }

```



The screenshot shows a Go Playground interface. On the left, the editor contains a Go program named 'main.go'. The code defines a 'main' package, imports 'fmt', and defines a 'main' function. Inside 'main', it declares a variable 'tahun' of type 'int', prompts the user for input, scans the input, and then checks if the year is a leap year using the condition: `if (tahun%400 == 0) || (tahun%4 == 0 && tahun%100 != 0) { kabisat = true }`. It then prints the result. On the right, the 'Output' pane shows the command `go run /tmp/vgpKwhGXwk.go` and the program's output: `Tahun: 2005` followed by `Kabisat: false`.

Penjelasan: Program ini meminta pengguna untuk memasukkan sebuah tahun dan kemudian memeriksa apakah tahun tersebut adalah tahun kabisat. Setelah input diterima, program akan menentukan apakah tahun kabisat berdasarkan dua kondisi: tahun yang habis dibagi 400 adalah kabisat, dan tahun yang habis dibagi 4 tetapi tidak habis dibagi 100 juga dianggap kabisat. Hasil pemeriksaan akan ditampilkan sebagai true jika tahun tersebut kabisat, dan false jika tidak.

3.

```

package main

import "fmt"

func main() {

    // Minta input temperatur dalam Celsius

```

```

var celsius float64

fmt.Print("Temperatur Celsius: ")

fmt.Scanln(&celsius)

// Konversi ke Fahrenheit
fahrenheit := (celsius * 9 / 5) + 32

// Konversi ke Reamur
reamur := celsius * 4 / 5

// Konversi ke Kelvin
kelvin := celsius + 273.15

// Tampilkan hasil konversi
fmt.Printf("Derajat Fahrenheit: %.2f\n", fahrenheit)
fmt.Printf("Derajat Reamur: %.2f\n", reamur)
fmt.Printf("Derajat Kelvin: %.2f\n", kelvin)
}

```

The screenshot shows a Go Playground interface. On the left, the code is displayed with line numbers 1 through 23. The code is a Go program that prompts the user for a temperature in Celsius, converts it to Fahrenheit, Reamur, and Kelvin, and then prints the results. On the right, the 'Output' tab is selected, showing the execution results for the command 'go run /tmp/hlqu6Uo8ft.go'. The output shows the user input '50' and the calculated values: Fahrenheit 122.00, Reamur 40.00, and Kelvin 323.15.

main.go	Output
<pre> 1 package main 2 3 import "fmt" 4 5 func main() { 6 // Minta input temperatur dalam Celsius 7 var celsius float64 8 fmt.Print("Temperatur Celsius: ") 9 fmt.Scanln(&celsius) 10 11 // Konversi ke Fahrenheit 12 fahrenheit := (celsius * 9 / 5) + 32 13 // Konversi ke Reamur 14 reamur := celsius * 4 / 5 15 // Konversi ke Kelvin 16 kelvin := celsius + 273.15 17 18 // Tampilkan hasil konversi 19 fmt.Printf("Derajat Fahrenheit: %.2f\n", fahrenheit) 20 fmt.Printf("Derajat Reamur: %.2f\n", reamur) 21 fmt.Printf("Derajat Kelvin: %.2f\n", kelvin) 22 } 23 </pre>	<pre> go run /tmp/hlqu6Uo8ft.go Temperatur Celsius: 50 Derajat Fahrenheit: 122.00 Derajat Reamur: 40.00 Derajat Kelvin: 323.15 </pre>

Penjelasan: Program ini meminta pengguna untuk memasukkan temperatur dalam derajat Celsius dan kemudian mengonversinya ke derajat Fahrenheit, Reamur, dan Kelvin. Setelah pengguna memasukkan nilai Celsius, program melakukan perhitungan untuk setiap satuan temperatur: Fahrenheit dihitung dengan rumus \((

(Celsius $\times \frac{9}{5}$) + 32 \), Reamur dihitung dengan $(\text{Celsius} \times \frac{4}{5})$ \), dan Kelvin dihitung dengan $(\text{Celsius} + 273.15)$ \). Hasil konversi ditampilkan dengan format dua angka desimal untuk setiap satuan temperatur.

III. UNGUIDED

2B. 1.

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    // Tentukan urutan warna yang benar
    warnaDiharapkan := []string{"merah", "kuning",
    "hijau", "ungu"}

    hasilPercobaan := make([]bool, 5) // Slice untuk
    menyimpan hasil 5 percobaan

    // Lakukan sebanyak 5 kali percobaan
    for i := 0; i < 5; i++ {
        fmt.Printf("Percobaan %d:\n", i+1)

        // Ambil input warna untuk 4 gelas
        var gelas1, gelas2, gelas3, gelas4 string
        fmt.Print("Gelas 1: ")
        fmt.Scan(&gelas1)
        fmt.Print("Gelas 2: ")
        fmt.Scan(&gelas2)
        fmt.Print("Gelas 3: ")
        fmt.Scan(&gelas3)
```

```

        fmt.Print("Gelas 4: ")

        fmt.Scan(&gelas4)

        // Simpan warna-warna yang diinput ke dalam
slice
        warnaInput := []string{gelas1, gelas2,
gelas3, gelas4}

        // Bandingkan warna yang diinput dengan
warna yang diharapkan

        if strings.Join(warnaInput, " ") ==
strings.Join(warnaDiharapkan, " ") {

            hasilPercobaan[i] = true // Urutan
warna sesuai

        } else {

            hasilPercobaan[i] = false // Urutan
warna tidak sesuai

        }

        // Tampilkan input warna yang dimasukkan

        fmt.Printf("%s %s %s %s\n", gelas1, gelas2,
gelas3, gelas4)

    }

    // Cek hasil akhir

    berhasil := true

    for _, hasil := range hasilPercobaan {

        if !hasil {

            berhasil = false

```

```

        break
    }

}

// Tampilkan hasil akhir

if berhasil {

    fmt.Println("BERHASIL: true")

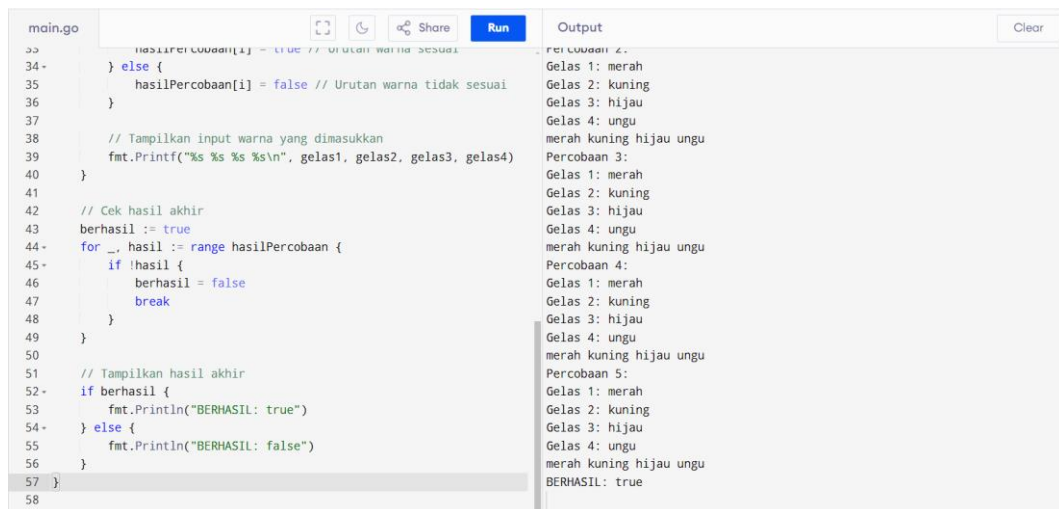
} else {

    fmt.Println("BERHASIL: false")

}

}

```



```

main.go
33
34- } else {
35     hasilPercobaan[i] = false // Urutan warna tidak sesuai
36 }
37
38 // Tampilkan input warna yang dimasukkan
39 fmt.Printf("%s %s %s %s\n", gelas1, gelas2, gelas3, gelas4)
40 }
41
42 // Cek hasil akhir
43 berhasil := true
44- for _, hasil := range hasilPercobaan {
45     if !hasil {
46         berhasil = false
47         break
48     }
49 }
50
51 // Tampilkan hasil akhir
52- if berhasil {
53     fmt.Println("BERHASIL: true")
54- } else {
55     fmt.Println("BERHASIL: false")
56 }
57 }
58
Output
Clear
Percobaan 1:
Gelas 1: merah
Gelas 2: kuning
Gelas 3: hijau
Gelas 4: ungu
merah kuning hijau ungu
Percobaan 2:
Gelas 1: merah
Gelas 2: kuning
Gelas 3: hijau
Gelas 4: ungu
merah kuning hijau ungu
Percobaan 3:
Gelas 1: merah
Gelas 2: kuning
Gelas 3: hijau
Gelas 4: ungu
merah kuning hijau ungu
Percobaan 4:
Gelas 1: merah
Gelas 2: kuning
Gelas 3: hijau
Gelas 4: ungu
merah kuning hijau ungu
Percobaan 5:
Gelas 1: merah
Gelas 2: kuning
Gelas 3: hijau
Gelas 4: ungu
merah kuning hijau ungu
BERHASIL: true

```

Penjelasan: Program ini bertujuan untuk meminta pengguna memasukkan warna dari 4 gelas selama 5 percobaan dan mengecek apakah urutan warna yang dimasukkan sesuai dengan merah, kuning, hijau, dan ungu. Setelah 5 percobaan, program akan menampilkan BERHASIL: true jika semua percobaan sesuai dan BERHASIL: false jika ada yang tidak sesuai.

2.

```
package main
```

```
import (  
    "fmt"  
)  
  
func main() {  
    var pita string  
    var bunga string  
    var count int  
  
    fmt.Println("Masukkan nama bunga (ketik 'SELESAI' untuk  
berhenti):")  
  
    for {  
        fmt.Printf("Bunga %d: ", count+1)  
        fmt.Scanln(&bunga)  
  
        if bunga == "SELESAI" {  
            break // Hentikan input jika pengguna mengetik  
'SELESAI'  
        }  
  
        if count == 0 {  
            pita = bunga // Jika ini adalah bunga pertama,  
langsung masukkan  
        } else {  
            pita += " - " + bunga // Jika sudah ada,  
tambahkan dengan pemisah ' - '  
        }  
    }  
}
```

```

        count++ // Increment jumlah bunga
    }

    // Tampilkan isi pita dan jumlah bunga
    fmt.Println("Pita:", pita)

    fmt.Printf("Jumlah bunga: %d\n", count)
}

```

The screenshot shows a Go IDE with a file named `main.go`. The code is a Go program that prompts the user to enter flower names until they type "SELESAI". It then prints the concatenated list of names and the total count. The output window shows the execution results: the user entered "edelweis", "lily", "mawar", and "SELESAI", resulting in the output "Pita: edelweis - lily - mawar" and "Jumlah bunga: 3".

```

main.go
12  fmt.Println("Masukkan nama bunga (ketik 'SELESAI' untuk berhenti):")
13  for {
14      fmt.Printf("Bunga %d: ", count+1)
15      fmt.Scanln(&bunga)
16
17      if bunga == "SELESAI" {
18          break // Hentikan input jika pengguna mengetik 'SELESAI'
19      }
20
21      if count == 0 {
22          pita = bunga // Jika ini adalah bunga pertama, langsung
                masukkan
23      } else {
24          pita += " - " + bunga // Jika sudah ada, tambahkan
                dengan pemisah ' - '
25      }
26
27      count++ // Increment jumlah bunga
28  }
29
30  // Tampilkan isi pita dan jumlah bunga
31  fmt.Println("Pita:", pita)
32  fmt.Printf("Jumlah bunga: %d\n", count)
33
Output
go run /tmp/321DkcQH2c.go
Masukkan nama bunga (ketik 'SELESAI' untuk berhenti):
Bunga 1: edelweis
Bunga 2: lily
Bunga 3: mawar
Bunga 4: SELESAI
Pita: edelweis - lily - mawar
Jumlah bunga: 3

```

Penjelasan: Program ini meminta pengguna untuk memasukkan nama-nama bunga satu per satu hingga pengguna mengetik "SELESAI". Setiap nama bunga yang dimasukkan akan disimpan dalam sebuah string bernama `pita`, dengan nama-nama bunga dipisahkan oleh tanda " — ". Program juga menghitung jumlah bunga yang dimasukkan. Setelah pengguna selesai memasukkan nama bunga, program menampilkan isi `pita` yang berisi semua nama bunga dan jumlah total bunga yang telah dimasukkan.

3.

```

package main

import (
    "fmt"
    "math"
)

```

```
func main() {  
    var beratKantong1, beratKantong2, totalBerat float64  
  
    for {  
        fmt.Print("Masukan berat belanja di kedua  
kantong: ")  
        _, err := fmt.Scanf("%f %f", &beratKantong1,  
&beratKantong2)  
        if err != nil {  
            fmt.Println("Input tidak valid. Silakan  
coba lagi.")  
            continue  
        }  
        if beratKantong1 < 0 || beratKantong2 < 0 {  
            fmt.Println("Proses selesai.")  
            break  
        }  
        totalBerat = beratKantong1 + beratKantong2  
  
        if totalBerat > 150 {  
            fmt.Println("Proses selesai.")  
            break  
        }  
  
        selisih := math.Abs(beratKantong1 -  
beratKantong2)
```

```

        if selisih >= 9 {

            fmt.Println("Sepeda motor pak Andi akan
oleng: true")

        } else {

            fmt.Println("Sepeda motor pak Andi akan
oleng: false")

        }

    }

}

```

The screenshot shows a Go IDE with a file named `main.go`. The code implements a program that repeatedly asks for two weights, checks for negative values or a total weight exceeding 150, and then checks if the absolute difference between the two weights is greater than or equal to 9. The output window shows the following results:

```

go run /tmp/EHYJhcOKD2.go
Masukan berat belanja di kedua kantong: 5 10
Sepeda motor pak Andi akan oleng: false
Masukan berat belanja di kedua kantong: 55.6 70.2
Sepeda motor pak Andi akan oleng: true
Masukan berat belanja di kedua kantong: 72.3 66.9
Sepeda motor pak Andi akan oleng: false
Masukan berat belanja di kedua kantong: 59.5 98.7
Proses selesai.

```

Penjelasan: Program ini meminta pengguna untuk memasukkan berat belanjaan di dua kantong secara berulang. Setelah setiap input, program memeriksa apakah berat salah satu kantong negatif atau jika total berat kedua kantong melebihi 150 kg, yang akan menghentikan proses. Jika tidak, program menghitung selisih berat antara kedua kantong dan menampilkan apakah sepeda motor Pak Andi akan oleng berdasarkan selisih tersebut: jika selisihnya 9 kg atau lebih, program akan mencetak true, sebaliknya, akan mencetak false.

4.

```

package main

import (

    "fmt"

```

```

    "math"
)

func f(k int) float64 {
    numerator := math.Pow(float64(4*k+2), 2)
    denominator := float64((4*k + 1) * (4*k + 3))
    return numerator / denominator
}

func sqrt2Approximation(K int) float64 {
    product := 1.0
    for k := 0; k <= K; k++ {
        product *= f(k)
    }
    return product
}

func main() {
    var K int

    // Input dan hitung untuk K pertama
    fmt.Print("Masukkan nilai K: ")
    fmt.Scan(&K)
    fmt.Printf("Nilai K = %d\n", K)
    fmt.Printf("Nilai akar 2 = %.10f\n",
sqrt2Approximation(K))

```



```

// Input dan hitung untuk K kedua

fmt.Print("Masukkan nilai K: ")

fmt.Scan(&K)

fmt.Printf("Nilai K = %d\n", K)

fmt.Printf("Nilai akar 2 = %.10f\n",
sqrt2Approximation(K))


// Input dan hitung untuk K ketiga

fmt.Print("Masukkan nilai K: ")

fmt.Scan(&K)

fmt.Printf("Nilai K = %d\n", K)

fmt.Printf("Nilai akar 2 = %.10f\n",
sqrt2Approximation(K))

}

```

The screenshot shows a Go code editor with a file named 'main.go'. The code is a Go program that prompts the user to input a value 'K' and then calculates the square root of 'K' using a function called 'sqrt2Approximation'. The code is structured with comments and function calls. The output window on the right shows the results of running the program for three different input values: 10, 100, and 1000. The output for each input shows the value of 'K' and the calculated square root with 10 decimal places of precision.

```

main.go
18 // Input dan hitung untuk K pertama
19 return product
20 }
21
22 func main() {
23     var K int
24
25     // Input dan hitung untuk K pertama
26     fmt.Print("Masukkan nilai K: ")
27     fmt.Scan(&K)
28     fmt.Printf("Nilai K = %d\n", K)
29     fmt.Printf("Nilai akar 2 = %.10f\n", sqrt2Approximation(K))
30
31     // Input dan hitung untuk K kedua
32     fmt.Print("Masukkan nilai K: ")
33     fmt.Scan(&K)
34     fmt.Printf("Nilai K = %d\n", K)
35     fmt.Printf("Nilai akar 2 = %.10f\n", sqrt2Approximation(K))
36
37     // Input dan hitung untuk K ketiga
38     fmt.Print("Masukkan nilai K: ")
39     fmt.Scan(&K)
40     fmt.Printf("Nilai K = %d\n", K)
41     fmt.Printf("Nilai akar 2 = %.10f\n", sqrt2Approximation(K))
42 }

```

Output

```

go run /tmp/mEPJxgWhJN.go
Masukkan nilai K: 10
Nilai K = 10
Nilai akar 2 = 1.4062058441
Masukkan nilai K: 100
Nilai K = 100
Nilai akar 2 = 1.4133387072
Masukkan nilai K: 1000
Nilai K = 1000
Nilai akar 2 = 1.4141252651

```

Penjelasan: Program ini menghitung pendekatan nilai akar dua berdasarkan input dari pengguna. Fungsi yang digunakan menghitung nilai berdasarkan rumus tertentu, di mana hasilnya didapatkan dari pembagian dua ekspresi yang terdiri dari operasi aritmetika. Fungsi lain mengalikan nilai-nilai yang dihasilkan dari fungsi pertama untuk semua iterasi yang diminta, sehingga menghasilkan pendekatan untuk akar dua. Program meminta pengguna untuk memasukkan nilai beberapa kali, dan setelah setiap input, ia menampilkan nilai yang dimasukkan serta hasil pendekatan akar dua dengan presisi yang tinggi.

2C.1.

```
package main

import "fmt"

func main() {
    var berat, kg, gr, biayaKirim, tambahanBiaya,
    totalBiaya int

    fmt.Print("Berat parsel (gram): ")
    fmt.Scan(&berat)

    kg = berat / 1000
    gr = berat % 1000

    if kg > 10 {
        tambahanBiaya = 0
    } else if gr >= 500 {
        tambahanBiaya = gr * 5
    } else {
        tambahanBiaya = gr * 15
    }

    biayaKirim = kg * 10000
    totalBiaya = biayaKirim + tambahanBiaya

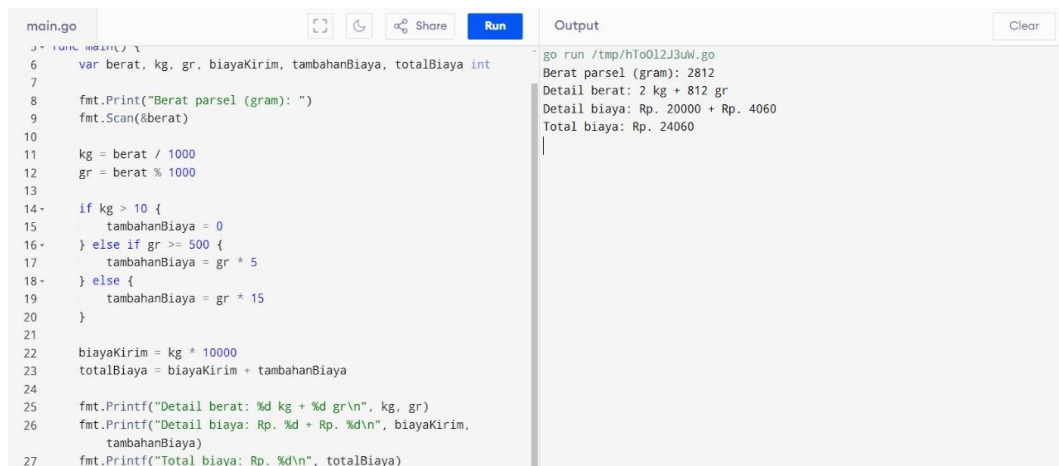
    fmt.Printf("Detail berat: %d kg + %d gr\n", kg, gr)
```

```

        fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n",
biayaKirim, tambahanBiaya)

        fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)
    }

```



```

main.go
6  var berat, kg, gr, biayaKirim, tambahanBiaya, totalBiaya int
7
8  fmt.Print("Berat parcel (gram): ")
9  fmt.Scan(&berat)
10
11  kg = berat / 1000
12  gr = berat % 1000
13
14- if kg > 10 {
15     tambahanBiaya = 0
16- } else if gr >= 500 {
17     tambahanBiaya = gr * 5
18- } else {
19     tambahanBiaya = gr * 15
20 }
21
22 biayaKirim = kg * 10000
23 totalBiaya = biayaKirim + tambahanBiaya
24
25 fmt.Printf("Detail berat: %d kg + %d gr\n", kg, gr)
26 fmt.Printf("Detail biaya: Rp. %d + Rp. %d\n", biayaKirim,
    tambahanBiaya)
27 fmt.Printf("Total biaya: Rp. %d\n", totalBiaya)

```

```

Output
go run /tmp/hTo012J3uW.go
Berat parcel (gram): 2812
Detail berat: 2 kg + 812 gr
Detail biaya: Rp. 20000 + Rp. 4060
Total biaya: Rp. 24060

```

Penjelasan: Program ini menghitung biaya pengiriman parcel berdasarkan berat yang dimasukkan dalam gram. Pengguna diminta untuk memasukkan berat parcel, kemudian program menghitung berat dalam kilogram dan gram. Jika berat melebihi sepuluh kilogram, tidak ada tambahan biaya; jika gram lebih dari atau sama dengan 500, tambahan biaya dihitung berdasarkan tarif tertentu; jika kurang dari 500, tarif tambahan yang berbeda diterapkan. Biaya pengiriman dasar dihitung dari berat dalam kilogram, dan total biaya adalah jumlah dari biaya pengiriman dan tambahan biaya. Program akhirnya menampilkan detail berat dan biaya yang dihitung.

2.

```

package main

import "fmt"

func main() {

    var nilaiAkhir float64

    var grade string

    fmt.Print("Nilai akhir mata kuliah: ")

```

```

    fmt.Scanln(&nilaiAkhir)

    if nilaiAkhir > 80 {
        grade = "A"
    } else if nilaiAkhir > 72.5 {
        grade = "AB"
    } else if nilaiAkhir > 65 {
        grade = "B"
    } else if nilaiAkhir > 57.5 {
        grade = "BC"
    } else if nilaiAkhir > 50 {
        grade = "C"
    } else if nilaiAkhir > 40 {
        grade = "D"
    } else {
        grade = "E"
    }

    fmt.Println("Nilai mata kuliah: ", grade)
}

```

main.go	Output
<pre> 1 import "fmt" 2 3 func main() { 4 var nilaiAkhir float64 5 var grade string 6 fmt.Print("Nilai akhir mata kuliah: ") 7 fmt.Scanln(&nilaiAkhir) 8 9 if nilaiAkhir > 80 { 10 grade = "A" 11 } else if nilaiAkhir > 72.5 { 12 grade = "AB" 13 } else if nilaiAkhir > 65 { 14 grade = "B" 15 } else if nilaiAkhir > 57.5 { 16 grade = "BC" 17 } else if nilaiAkhir > 50 { 18 grade = "C" 19 } else if nilaiAkhir > 40 { 20 grade = "D" 21 } else { 22 grade = "E" 23 } 24 25 fmt.Println("Nilai mata kuliah: ", grade) 26 } </pre>	<pre> go run /tmp/rAulMx6why.go Nilai akhir mata kuliah: 90 Nilai mata kuliah: A </pre>

Penjelasan: Program ini meminta pengguna untuk memasukkan nilai akhir mata kuliah dan kemudian menentukan grade atau nilai huruf berdasarkan rentang nilai yang telah ditentukan. Jika nilai akhir lebih dari 80, grade akan menjadi A, dan seterusnya hingga nilai 40 yang akan mendapatkan grade D. Jika nilai akhir kurang dari atau sama dengan 40, grade yang diberikan adalah E. Setelah menentukan grade, program menampilkan nilai huruf yang sesuai dengan nilai akhir yang dimasukkan.

3.

```
package main

import "fmt"

func cariFaktor(b int) []int {
    var faktor []int
    for i := 1; i <= b; i++ {
        if b%i == 0 {
            faktor = append(faktor, i)
        }
    }
    return faktor
}

func adalahPrima(b int) bool {
    faktor := cariFaktor(b)
    return len(faktor) == 2
}

func main() {
```

```

    var b int

    fmt.Print("Bilangan: ")

    fmt.Scan(&b)

    faktor := cariFaktor(b)

    fmt.Print("Faktor: ")

    for _, f := range faktor {
        fmt.Printf("%d ", f)
    }

    fmt.Println()

    statusPrima := adalahPrima(b)

    fmt.Printf("Prima: %t\n", statusPrima)
}

```

The screenshot shows a Go IDE with a file named `main.go`. The code is as follows:

```

10- }
11- }
12- return faktor
13- }
14-
15- func adalahPrima(b int) bool {
16-     faktor := cariFaktor(b)
17-     return len(faktor) == 2
18- }
19-
20- func main() {
21-     var b int
22-     fmt.Print("Bilangan: ")
23-     fmt.Scan(&b)
24-
25-     faktor := cariFaktor(b)
26-     fmt.Print("Faktor: ")
27-     for _, f := range faktor {
28-         fmt.Printf("%d ", f)
29-     }
30-     fmt.Println()
31-
32-     statusPrima := adalahPrima(b)
33-     fmt.Printf("Prima: %t\n", statusPrima)

```

The `Output` pane on the right shows the result of running the program:

```

go run /tmp/R5ey57lij5.go
Bilangan: 28
Faktor: 1 2 4 7 14 28
Prima: false

```

Penjelasan: Program ini menerima input sebuah bilangan dan menghitung faktor-faktornya, yaitu angka-angka yang dapat membagi bilangan tersebut tanpa sisa. Fungsi `cariFaktor` digunakan untuk menemukan semua faktor dari bilangan yang dimasukkan, sementara fungsi `adalahPrima` menentukan apakah bilangan tersebut adalah bilangan prima, yaitu bilangan yang hanya memiliki dua faktor, yaitu 1 dan dirinya sendiri. Setelah menghitung faktor-faktor, program mencetak daftar faktor dan menunjukkan apakah bilangan tersebut adalah bilangan prima atau tidak.