

**LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN 2**

MODUL II

REVIEW STRUKTUR KONTROL



Disusun Oleh :

ARIHQ RADHITYA PRADANA

2311102260

IF 11 06

Dosen Pengampu :

Abednego Dwi Septiadi

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO 2024

I. DASAR TEORI

Struktur kontrol dalam bahasa pemrograman Go (Golang) adalah elemen fundamental yang mengatur alur eksekusi program. Terdapat beberapa jenis struktur kontrol yang umum digunakan, yaitu percabangan dan perulangan.

Percabangan memungkinkan programmer untuk membuat keputusan berdasarkan kondisi tertentu. Dalam Golang, pernyataan `if` digunakan untuk mengevaluasi suatu kondisi dan menjalankan blok kode yang sesuai jika kondisi tersebut terpenuhi. Jika kondisi tidak terpenuhi, programmer dapat menggunakan pernyataan `else` untuk menentukan tindakan alternatif. Selain itu, pernyataan `switch` memberikan cara yang lebih terstruktur untuk memilih di antara beberapa kemungkinan berdasarkan nilai ekspresi, sehingga membuat kode lebih mudah dibaca dan dikelola.

Sementara itu, perulangan dalam Golang hanya menggunakan pernyataan `for`, yang merupakan satu-satunya cara untuk melakukan iterasi. Struktur ini dapat digunakan dalam berbagai cara, seperti iterasi dengan batasan tertentu atau sebagai perulangan tak terbatas. Keunikan dari pernyataan `for` di Golang adalah kemampuannya untuk menggantikan fungsi-fungsi lain seperti `while` dan `do while`, sehingga menyederhanakan sintaks.

Selain itu, Golang juga menyediakan mekanisme kontrol alur dalam perulangan melalui kata kunci seperti `break`, `continue`, dan `goto`. Kata kunci `break` digunakan untuk menghentikan eksekusi perulangan secara langsung, sedangkan `continue` memungkinkan program untuk melanjutkan ke iterasi berikutnya tanpa mengeksekusi sisa blok kode dalam iterasi saat ini. Kata kunci `goto`, meskipun kurang umum digunakan karena dapat membuat kode sulit dibaca, memberikan kemampuan untuk melompat ke label tertentu dalam kode.

Secara keseluruhan, pemahaman yang baik tentang struktur kontrol dalam Golang sangat penting bagi pengembang untuk menulis kode yang efisien dan terstruktur dengan baik. Dengan memanfaatkan percabangan dan perulangan secara efektif, programmer dapat menciptakan aplikasi yang responsif dan mudah dipelihara.

II. GUIDED

Guided 1

```
package main

import "fmt"

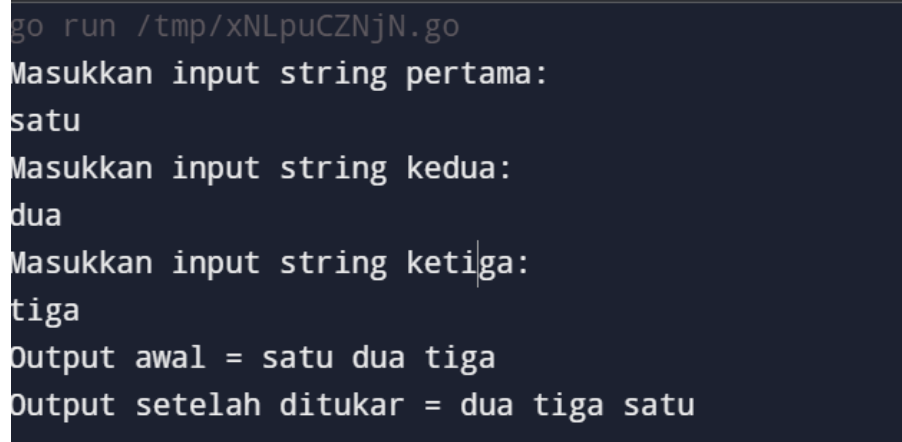
func main() {
    var (
        satu, dua, tiga string
        temp                string
    )
    fmt.Println("Masukkan input string pertama: ")
    fmt.Scanln(&satu)
    fmt.Println("Masukkan input string kedua: ")
    fmt.Scanln(&dua)
    fmt.Println("Masukkan input string ketiga: ")
    fmt.Scanln(&tiga)

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

    temp = satu
    satu = dua
    dua = tiga
    tiga = temp

    fmt.Println("Output setelah ditukar = " + satu + " " + dua + " " + tiga)
}
```

Screenshoot Output



```
go run /tmp/xNLpuCZNjN.go
Masukkan input string pertama:
satu
Masukkan input string kedua:
dua
Masukkan input string ketiga:
tiga
Output awal = satu dua tiga
Output setelah ditukar = dua tiga satu
```

Deskripsi Program

Kode di atas adalah program sederhana yang ditulis dalam bahasa pemrograman Go (Golang) untuk mengelola dan menukar tiga input string dari pengguna. Program ini dimulai dengan mendeklarasikan tiga variabel string (`satu`, `dua`, dan `tiga`) serta satu variabel tambahan (`temp`) yang digunakan untuk menyimpan nilai sementara saat melakukan penukaran. Pengguna diminta untuk memasukkan tiga string yang masing-masing disimpan dalam variabel yang sesuai. Setelah menerima input, program menampilkan output awal yang mencantumkan ketiga string tersebut. Proses penukaran dilakukan dengan menyimpan nilai dari `satu` ke dalam `temp`, kemudian mengubah `satu` menjadi nilai `dua`, `dua` menjadi nilai `tiga`, dan akhirnya `tiga` menjadi nilai yang tersimpan di `temp`. Dengan cara ini, hasil akhir dari penukaran adalah nilai `dua`, `tiga`, dan `satu`, yang kemudian ditampilkan sebagai output setelah ditukar. Program ini memberikan ilustrasi yang jelas tentang cara kerja variabel dan penukaran nilai dalam pemrograman.

Guided 2

```
package main

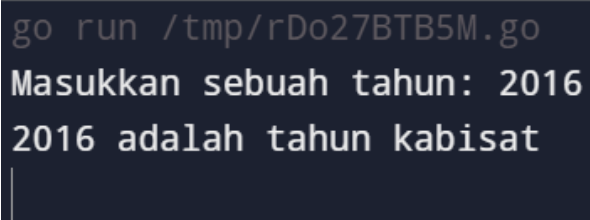
import (
    "fmt"
)

func cekTahunKabisat() {
    var tahun int
    fmt.Print("Masukkan sebuah tahun: ")
    fmt.Scanln(&tahun)

    if (tahun%400 == 0) || (tahun%4 == 0 && tahun%100 !=
0) {
        fmt.Println(tahun, "adalah tahun kabisat")
    } else {
        fmt.Println(tahun, "bukan tahun kabisat")
    }
}

func main() {
    cekTahunKabisat()
}
```

Screenshoot Output



```
go run /tmp/rDo27BTB5M.go
Masukkan sebuah tahun: 2016
2016 adalah tahun kabisat
|
```

Deskripsi Program

Kode program di atas ditulis dalam bahasa Go dan berfungsi untuk memeriksa apakah suatu tahun adalah tahun kabisat atau bukan. Program ini memiliki fungsi utama ``main()`` yang memanggil fungsi lain, yaitu ``cekTahunKabisat()``.

Dalam fungsi ``cekTahunKabisat()``, variabel ``tahun`` dideklarasikan untuk menampung input dari pengguna. Program meminta pengguna memasukkan sebuah angka yang mewakili tahun menggunakan perintah ``fmt.Scanln()``. Setelah menerima input, kondisi untuk menentukan apakah tahun tersebut merupakan tahun kabisat diperiksa menggunakan pernyataan ``if-else``.

Kondisi dalam pernyataan ``if`` memeriksa dua hal: apakah tahun tersebut habis dibagi 400, atau apakah tahun tersebut habis dibagi 4 tetapi tidak habis dibagi 100. Jika salah satu kondisi terpenuhi, maka tahun tersebut dianggap tahun kabisat dan program akan mencetak "<tahun> adalah tahun kabisat". Jika tidak, maka program akan mencetak "<tahun> bukan tahun kabisat".

Secara keseluruhan, program ini memanfaatkan konsep aritmatika sederhana untuk menentukan tahun kabisat berdasarkan aturan umum kalender Gregorian.

Guided 3

```
package main

import (
    "fmt"
    "math"
)

func main() {
    var radius int

    // Input jari-jari
    fmt.Print("Masukkan jari-jari bola: ")
    fmt.Scan(&radius)

    // Menghitung volume bola
    volume := (4.0 / 3.0) * math.Pi *
    math.Pow(float64(radius), 3)

    // Menghitung luas permukaan bola
    luas := 4 * math.Pi * math.Pow(float64(radius), 2)

    // Tampilkan hasil
    fmt.Printf("Volume bola = %.2f\n", volume)
    fmt.Printf("Luas permukaan bola = %.2f\n", luas)
}
```

Screenshoot Output

```
go run /tmp/LkJAnYe6d0.go
Masukkan jari-jari bola: 5
Volume bola = 523.60
Luas permukaan bola = 314.16
```

Deskripsi Program

Kode program di atas ditulis dalam bahasa Go dan digunakan untuk menghitung volume serta luas permukaan bola berdasarkan jari-jari yang diinput oleh pengguna.

Pertama, variabel `radius` dideklarasikan sebagai tipe integer, yang akan menyimpan nilai jari-jari bola yang dimasukkan oleh pengguna. Program kemudian menampilkan pesan kepada pengguna untuk memasukkan jari-jari menggunakan perintah `fmt.Scan()`, yang membaca input dari pengguna dan menyimpannya dalam variabel `radius`.

Setelah mendapatkan nilai jari-jari, program menghitung volume bola menggunakan rumus volume bola, Di mana `math.Pi` merepresentasikan nilai π (Pi) dan `math.Pow` digunakan untuk menghitung pangkat tiga dari jari-jari (`r^3`). Nilai jari-jari dikonversi ke tipe `float64` untuk memastikan operasi matematika dilakukan dengan benar.

Selanjutnya, luas permukaan bola. Setelah menghitung kedua nilai, program mencetak hasil perhitungan volume dan luas permukaan bola dengan format dua angka desimal menggunakan `fmt.Printf()`.

Secara keseluruhan, program ini menghitung dan menampilkan volume serta luas permukaan bola berdasarkan input jari-jari yang diberikan oleh pengguna.

III. UNGUIDED

Unguided 1

```
package main

import "fmt"

func main() {
    var celsius float64

    // Input nilai temperatur dalam derajat Celsius
    fmt.Print("Masukkan temperatur dalam derajat Celsius: ")
    fmt.Scan(&celsius)

    // Menghitung suhu dalam Fahrenheit
    fahrenheit := (celsius * 9 / 5) + 32

    // Menghitung suhu dalam Reamur
    reamur := celsius * 4 / 5

    // Menghitung suhu dalam Kelvin
    kelvin := celsius + 273.15

    // Menampilkan hasil konversi
    fmt.Printf("Temperatur dalam Fahrenheit: %.2f °F\n", fahrenheit)
    fmt.Printf("Temperatur dalam Reamur: %.2f °R\n", reamur)
    fmt.Printf("Temperatur dalam Kelvin: %.2f K\n", kelvin)
}
```

Screenshoot Output

```
go run /tmp/dX3WmduLPU.go
Masukkan temperatur dalam derajat Celsius: 50
Temperatur dalam Fahrenheit: 122.00 °F
Temperatur dalam Reamur: 40.00 °R
Temperatur dalam Kelvin: 323.15 K
```

Deskripsi Program

Program di atas ditulis dalam bahasa Go dan digunakan untuk mengonversi suhu dari Celsius ke Fahrenheit, Reamur, dan Kelvin. Pertama, program mendeklarasikan variabel ``celsius`` dengan tipe ``float64`` untuk menyimpan nilai suhu yang akan diinput oleh pengguna.

Setelah itu, program meminta pengguna untuk memasukkan suhu dalam derajat Celsius dengan menggunakan perintah ``fmt.Scan()``. Berdasarkan nilai yang diinput, program melakukan konversi suhu ke tiga satuan yang berbeda. Setelah konversi dilakukan, program akan menampilkan hasilnya dalam bentuk suhu Fahrenheit, Reamur, dan Kelvin dengan dua angka desimal untuk setiap hasil. Ini memungkinkan pengguna untuk melihat suhu dalam berbagai skala yang berbeda sesuai dengan input awal dalam derajat Celsius.

Unguided 2

```
package main

import "fmt"

func main() {
    var intVals [5]int
    var charVals [3]rune

    // Input baris pertama: 5 buah integer
    fmt.Println("Masukkan 5 buah data integer (nilai antara 32 s.d. 127):")
    for i := 0; i < 5; i++ {
        fmt.Scan(&intVals[i])
    }

    // Input baris kedua: 3 buah karakter
    fmt.Println("Masukkan 3 buah karakter tanpa spasi:")
    for i := 0; i < 3; i++ {
        fmt.Scanf("%c", &charVals[i])
    }

    // Cetak representasi karakter dari integer yang diberikan
    fmt.Println("Representasi karakter dari integer:")
    for i := 0; i < 5; i++ {
        fmt.Printf("%c", intVals[i])
    }
    fmt.Println()

    // Cetak karakter setelah karakter input (berdasarkan tabel ASCII)
    fmt.Println("3 karakter setelah inputan:")
    for i := 0; i < 3; i++ {
        fmt.Printf("%c", charVals[i]+3)
    }
    fmt.Println()
}
```

Screenshoot Output

```
go run /tmp/FVsySPK9wk.go
Masukkan 5 buah data integer (nilai antara 32 s.d. 127):
66 97 103 117 115
Masukkan 3 buah karakter tanpa spasi:
SNO
Representasi karakter dari integer:
Bagus
3 karakter setelah inputan:
VQR
```

Deskripsi Program

Program di atas ditulis dalam bahasa Go dan berfungsi untuk membaca input berupa lima buah bilangan bulat (integer) dan tiga buah karakter, kemudian menampilkan hasil yang terkait dengan representasi karakter dari bilangan-bilangan tersebut serta karakter ASCII yang berada tiga posisi setelah karakter input.

Pertama, program mendeklarasikan dua array: `intVals` dengan panjang 5 untuk menyimpan integer, dan `charVals` dengan panjang 3 untuk menyimpan karakter bertipe `rune`. Pengguna diminta memasukkan lima buah integer dengan nilai antara 32 hingga 127, yang merupakan rentang nilai ASCII yang mewakili karakter-karakter yang dapat dicetak. Setiap nilai integer dimasukkan ke dalam array `intVals` menggunakan perulangan `for` dan fungsi `fmt.Scan()`.

Selanjutnya, pengguna diminta memasukkan tiga karakter tanpa spasi. Karakter-karakter ini disimpan dalam array `charVals` melalui perulangan `for` dan fungsi `fmt.Scanf("%c", ...)`, yang memungkinkan input berupa karakter tunggal.

Setelah data input dikumpulkan, program mencetak representasi karakter yang sesuai dengan lima nilai integer yang diberikan oleh pengguna. Hal ini dilakukan dengan menggunakan format `"%c"` dalam fungsi `fmt.Printf()`, yang akan mengonversi integer ke karakter ASCII yang sesuai. Selain itu, program juga menampilkan tiga karakter baru yang merupakan hasil dari menambahkan tiga posisi ASCII pada masing-masing karakter yang dimasukkan pengguna. Ini dilakukan dengan menambahkan nilai 3 pada setiap elemen `charVals` sebelum dicetak.

Unguided 3

```
package main

import "fmt"

func main() {
    const percobaan = 5
    expectedColors := [4]string{"merah", "kuning",
    "hijau", "ungu"}
    var hasilPercobaan bool = true

    // Iterasi untuk 5 kali percobaan
    for i := 0; i < percobaan; i++ {
        var warna [4]string

        fmt.Printf("Masukkan warna untuk percobaan ke-%d
(pisahkan dengan spasi):\n", i+1)
        for j := 0; j < 4; j++ {
            fmt.Scan(&warna[j])
        }

        // Cek apakah urutan warna sesuai dengan yang
        diharapkan
        for j := 0; j < 4; j++ {
            if warna[j] != expectedColors[j] {
                hasilPercobaan = false
            }
        }
    }

    // Tampilkan hasil akhir: true jika semua percobaan
    berhasil, false jika ada yang gagal
    fmt.Println(hasilPercobaan)
}
```

Screenshoot Output

```
go run /tmp/H1KvLzJFZK.go
Masukkan warna untuk percobaan ke-1 (pisahkan dengan spasi):
merah kuning hijau ungu
Masukkan warna untuk percobaan ke-2 (pisahkan dengan spasi):
merah kuning hijau ungu
Masukkan warna untuk percobaan ke-3 (pisahkan dengan spasi):
merah kuning hijau ungu
Masukkan warna untuk percobaan ke-4 (pisahkan dengan spasi):
merah kuning hijau ungu
Masukkan warna untuk percobaan ke-5 (pisahkan dengan spasi):
merh kuning hijau ungu
false
|
```

Deskripsi Program

Program di atas ditulis dalam bahasa Go dan bertujuan untuk melakukan serangkaian percobaan dalam menginput urutan warna yang diharapkan, serta mengevaluasi apakah semua input sesuai dengan urutan yang ditentukan.

Program dimulai dengan mendeklarasikan sebuah konstanta `percobaan` yang bernilai 5, yang menunjukkan jumlah percobaan yang akan dilakukan. Kemudian, sebuah array `expectedColors` berisi empat warna, yaitu "merah", "kuning", "hijau", dan "ungu", yang menjadi urutan yang diharapkan. Variabel `hasilPercobaan` diinisialisasi dengan nilai `true`, yang akan digunakan untuk menunjukkan apakah semua percobaan berhasil.

Selanjutnya, program menjalankan sebuah loop yang berlangsung sebanyak lima kali, masing-masing untuk percobaan yang berbeda. Di dalam loop ini, program meminta pengguna untuk memasukkan empat warna pada setiap percobaan, yang disimpan dalam array `warna`. Input ini diharapkan dipisahkan dengan spasi.

Setelah mendapatkan input, program memeriksa apakah urutan warna yang dimasukkan sesuai dengan yang ada di `expectedColors`. Loop kedua membandingkan setiap warna dari array `warna` dengan elemen yang sesuai dari array `expectedColors`. Jika terdapat ketidaksesuaian, maka `hasilPercobaan` diubah menjadi `false`, menandakan bahwa setidaknya satu percobaan tidak berhasil.

Di akhir program, hasil akhir dari semua percobaan ditampilkan. Jika semua urutan warna sesuai dengan yang diharapkan, maka program akan mencetak `true`; jika ada satu atau lebih yang tidak sesuai, maka akan mencetak `false`. Program ini dapat digunakan untuk menguji kesesuaian input pengguna dengan urutan yang telah ditentukan.

Unguided 4

```
package main

import (
    "fmt"
    "strings"
)

func main() {
    var pita string
    var count int

    // Meminta pengguna untuk memasukkan batas jumlah
    bunga
    var limit int
    fmt.Print("Masukkan jumlah bunga yang ingin
    ditambahkan: ")
    fmt.Scan(&limit)

    // Proses input bunga hingga mencapai batas jumlah
    bunga
    for count < limit {
        var bunga string
        fmt.Printf("Bunga %d: ", count+1)
        fmt.Scan(&bunga)

        // Cek apakah input adalah 'SELESAI'
        if strings.ToUpper(bunga) == "SELESAI" {
            break
        }

        // Penggabungan string dengan operator +
        if count == 0 {
            pita += bunga
        } else {
            pita += " - " + bunga
        }

        count++
    }

    // Tampilkan isi pita dan banyaknya bunga
    fmt.Println("Pita:", pita)
    fmt.Println("Bunga:", count)
}
```


Screenshoot Output

```
go run /tmp/WxvLLZTWUq.go
Masukkan jumlah bunga yang ingin ditambahkan: 3
Bunga 1: kertas
Bunga 2: mawar
Bunga 3: tulip
Pita: kertas - mawar - tulip
Bunga: 3
|
```

Deskripsi Program

Kode di atas adalah program sederhana yang ditulis dalam bahasa pemrograman Go (Golang). Program ini bertujuan untuk mengumpulkan nama bunga dari pengguna dan menggabungkannya menjadi satu string yang disebut "pita".

Pertama, program meminta pengguna untuk memasukkan jumlah maksimum bunga yang ingin mereka tambahkan, yang disimpan dalam variabel `limit`. Selanjutnya, program menggunakan loop `for` untuk meminta pengguna memasukkan nama bunga satu per satu hingga jumlah bunga yang diinput mencapai batas yang ditentukan atau pengguna memasukkan kata kunci "SELESAI". Saat pengguna diminta memasukkan nama bunga, program menampilkan nomor urut bunga yang sedang diminta, mulai dari satu. Jika pengguna memasukkan "SELESAI" dalam format apa pun (menggunakan `strings.ToUpper` untuk mengabaikan kasus huruf), loop akan berhenti. Nama-nama bunga yang dimasukkan kemudian digabungkan menjadi satu string dengan pemisah " - " di antara setiap bunga, sehingga hasil akhir akan terlihat seperti daftar bunga. Setelah semua input selesai, program menampilkan hasil gabungan bunga dalam variabel `pita` dan jumlah total bunga yang telah dimasukkan oleh pengguna. Program ini memberikan antarmuka yang sederhana dan interaktif bagi pengguna untuk mengumpulkan dan melihat daftar bunga.

Unguided 5

```
package main

import (
    "fmt"
    "math"
)

func main() {
    for {
        var kantong1, kantong2 float64

        // Meminta input berat belanjaan dari user
        fmt.Print("Masukan berat belanjaan di kedua kantong: ")
        fmt.Scan(&kantong1, &kantong2)

        // Hentikan jika salah satu berat negatif
        if kantong1 < 0 || kantong2 < 0 {
            fmt.Println("Proses selesai.")
            break
        }

        // Hentikan jika total berat lebih dari 150 kg
        if kantong1+kantong2 > 150 {
            fmt.Println("Proses selesai.")
            break
        }

        // Cek apakah sepeda motor akan oleng (selisih
        >= 9 kg)
        oleng := math.Abs(kantong1-kantong2) >= 9
        fmt.Printf("Sepeda motor Pak Andi akan oleng:
        %v\n", oleng)

        // Hentikan jika salah satu kantong mencapai 9
        kg atau lebih
        if kantong1 >= 9 || kantong2 >= 9 {
            fmt.Println("Proses selesai.")
            break
        }
    }
}
```

Screenshoot Output

```
go run /tmp/1wfZNze0Ir.go
Masukan berat belanjaan di kedua kantong: 5 10
Sepeda motor Pak Andi akan oleng: false
Proses selesai.
```

Deskripsi Program

Kode di atas adalah program dalam bahasa pemrograman Go (Golang) yang bertujuan untuk membantu pengguna dalam memantau berat belanjaan di dua kantong dan dampaknya terhadap stabilitas sepeda motor saat berkendara. Program ini menggunakan loop `for` tak terbatas yang meminta pengguna untuk memasukkan berat belanjaan di kedua kantong, yang disimpan dalam variabel `kantong1` dan `kantong2`. Jika pengguna memasukkan berat negatif untuk salah satu kantong, atau jika total berat kedua kantong melebihi 150 kg, program akan menghentikan proses dan menampilkan pesan "Proses selesai". Selanjutnya, program memeriksa apakah ada perbedaan berat antara kedua kantong yang membuat sepeda motor berpotensi oleng, yaitu jika selisih beratnya lebih besar atau sama dengan 9 kg. Hasil pemeriksaan ini ditampilkan dalam bentuk boolean (true atau false). Jika salah satu kantong memiliki berat 9 kg atau lebih, program juga akan menghentikan proses. Dengan demikian, program ini memberikan umpan balik kepada pengguna mengenai stabilitas sepeda motor berdasarkan berat belanjaan yang dibawa dan menetapkan batasan untuk keselamatan berkendara.

Unguided 6

```
package main

import (
    "fmt"
)

// Fungsi untuk menghitung f(K) sesuai rumus
func calculateF(K float64) float64 {
    return (4*K + 2) * (4*K + 2) / ((4*K + 1) * (4*K + 3))
}

// Fungsi untuk menghampiri akar kuadrat dari 2
func approximateSqrt2(iterations int) float64 {
    sqrt2Approx := 1.0
    for i := 0; i < iterations; i++ {
        sqrt2Approx = (sqrt2Approx + 2/sqrt2Approx) / 2
    }
    return sqrt2Approx
}

func main() {
    // Menampilkan pesan untuk memasukkan nilai K
    var K float64
    fmt.Print("Masukkan nilai K: ")
    fmt.Scanln(&K)

    // Menghitung nilai f(K)
    fK := calculateF(K)

    // Menampilkan hasil perhitungan f(K)
    fmt.Println("Nilai f(K) =", fK)

    // Menghitung hampiran akar kuadrat dari 2
    sqrt2Approx := approximateSqrt2(10)

    // Menampilkan informasi tentang akar kuadrat dari 2
    fmt.Println("\n√2 merupakan bilangan irasional. Meskipun demikian, nilai tersebut dapat dihamperi")
    fmt.Println("dengan rumus berikut:")
    fmt.Println("√2 ≈", sqrt2Approx)
}
```

Screenshoot Output

```
Masukkan nilai K: 10
Nilai f(K) = 1.0005672149744753

√2 merupakan bilangan irasional. Meskipun demikian, nilai tersebut
    dapat dihamperi
dengan rumus berikut:
√2 ≈ 1.414213562373095
|
```

Deskripsi Program

Program dalam bahasa Go (Golang) ini bertujuan untuk menghitung nilai fungsi $f(K)$ berdasarkan input dari pengguna dan menghampiri nilai akar kuadrat dari 2.

Program dimulai dengan mendeklarasikan package `'main'`, yang merupakan konvensi dalam Go untuk menandai bahwa ini adalah program yang dapat dieksekusi. Selanjutnya, program mengimpor package `'fmt'`, yang diperlukan untuk melakukan input dan output format. Terdapat dua fungsi utama dalam program ini: `'calculateF'` dan `'approximateSqrt2'`.

Fungsi `'calculateF'` menerima parameter `'K'` bertipe `'float64'` dan menghitung nilai $f(K)$ berdasarkan rumus yang ditentukan. Fungsi ini mengembalikan hasil perhitungan tersebut. Fungsi kedua, `'approximateSqrt2'`, dirancang untuk menghampiri nilai akar kuadrat dari 2 dengan menggunakan metode Newton-Raphson. Fungsi ini menerima parameter `'iterations'`, yang menentukan berapa kali iterasi akan dilakukan. Di dalam fungsi ini, variabel `'sqrt2Approx'` diinisialisasi dengan nilai 1.0, dan kemudian diperbarui dalam loop selama jumlah iterasi yang ditentukan.

Dalam fungsi `'main'`, program meminta pengguna untuk memasukkan nilai K dengan menggunakan `'fmt.Print'` dan `'fmt.Scanln'`. Nilai K yang dimasukkan kemudian digunakan untuk menghitung nilai $f(K)$ dengan memanggil fungsi `'calculateF'`. Hasil perhitungan tersebut kemudian ditampilkan ke layar. Setelah itu, program menghitung hampiran akar kuadrat dari 2 dengan memanggil fungsi `'approximateSqrt2'` dengan argumen 10 untuk melakukan 10 iterasi. Hasil dari hampiran akar kuadrat ditampilkan kepada pengguna, diikuti dengan penjelasan bahwa akar kuadrat dari 2 adalah bilangan irasional, tetapi dapat dihamperi menggunakan metode yang telah dijelaskan.

Dengan struktur yang jelas dan penggunaan fungsi yang terpisah, program ini memudahkan pembacaan dan pemeliharaan kode, serta memberikan hasil yang sesuai dengan tujuan awal.