**LAPORAN PRAKTIKUM   
ALGORITMA DAN PEMROGRAMAN 2**

**MODUL IV**

**PROSEDUR**

**Sebuah gambar berisi logo, simbol, Grafis, Font

Deskripsi dibuat secara otomatis**

**Disusun Oleh :**

**Fariz Ilham / 2311102275**

**IF-11-06**

**Dosen Pengampu :**

**Abednego Dwi Septiadi**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

1. **DASAR TEORI**

* Definisi Procedure

Prosedur adalah potongan instruksi program yang dikelompokkan menjadi satu kesatuan untuk mengurangi kerumitan dan mengorganisasi kode dalam program yang besar. Prosedur tidak mengembalikan nilai, namun memberikan efek atau dampak langsung pada program saat dipanggil.

Suatu subprogram disebut “prosedur” apabila:

1. Tidak ada deklarasi tipe nilai yang dikembalikan.

2. Tidak terdapat kata kunci `return` di dalamnya.

Prosedur berfungsi seperti instruksi dasar dalam program, misalnya penugasan (assignment) atau penggunaan pustaka untuk input/output. Contoh dari penggunaan prosedur seperti operasi `cetak`, `hitung`, atau `mulai`, yang mewakili tindakan spesifik.

* Cara Memanggil Prosedur

Memanggil prosedur dilakukan dengan cara menyebutkan nama prosedur yang diinginkan, diikuti dengan parameter jika diperlukan. Prosedur akan dieksekusi sesuai dengan instruksi di dalamnya, tanpa harus mengembalikan nilai.

Contoh cara memanggil prosedur:

\*\*NamaProsedur(parameter1, parameter2, ...);\*\*

Setelah dipanggil, prosedur akan melaksanakan tugasnya dan memberikan efek langsung dalam program, seperti mencetak hasil, melakukan perhitungan, atau menjalankan operasi lainnya.

1. **GUIDED**

**Soal Studi Case**

1. Buatlah sebuah program beserta prosedur yang digunakan untuk menampilkan suatu pesan error, warning atau informasi berdasarkan masukan dari user.

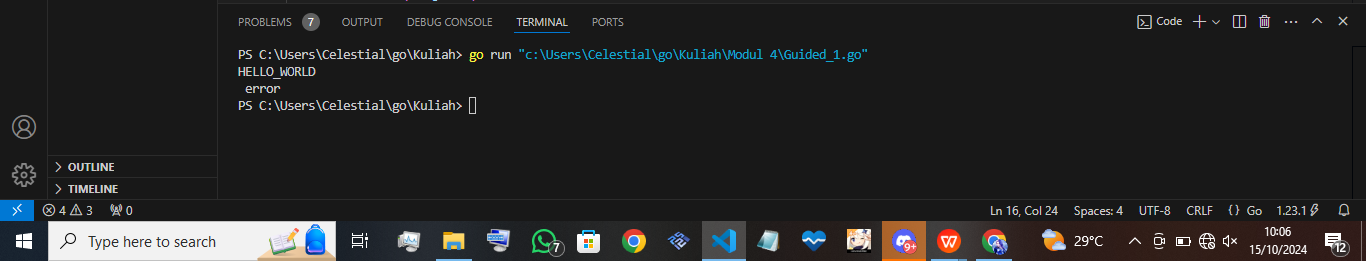
**Masukan** terdiri dari sebuah bilangan bulat flag (0 s.d. 2) dan sebuah string pesan M.

**Keluaran** berupa string pesan M beserta jenis pesannya, yaitu error, warning atau informasi berdasarkan nilai flag 0, 1 dan 2 secara berturut-turut

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func main() {  var bilangan int  var pesan string  fmt.Scan(&bilangan, &pesan)  cetakpesan(pesan, bilangan)  }  func cetakpesan(M string, flag int) {  var jenis string = ""  if flag == 0 {  jenis = "error"  } else if flag == 1 {  jenis = "warning"  } else if flag == 2 {  jenis = "informasi"  }  fmt.Println(M, jenis)  } |

**Screenshoot Output**

****

**Deskripsi Program**

Program ini merupakan program sederhana yang ditulis dalam bahasa Go, di mana pengguna dapat memasukkan dua input: sebuah bilangan bulat dan sebuah string (pesan). Program ini akan mencetak pesan tersebut dengan tipe status tertentu, seperti "error", "warning", atau "informasi", tergantung pada nilai bilangan yang dimasukkan.

* Algoritma

1. Input Data:

Program menunggu pengguna untuk memasukkan dua input: bilangan bulat `bilangan` dan string `pesan`.

2. Pemanggilan Fungsi `cetakpesan`:

Setelah input diterima, program memanggil fungsi `cetakpesan` dengan `pesan` dan `bilangan` sebagai argumennya.

3. Fungsi `cetakpesan`:

* Fungsi ini menerima dua parameter: `M` yang merupakan pesan dalam bentuk string, dan `flag` yang merupakan bilangan bulat.
* Fungsi memeriksa nilai dari `flag`:
* Jika `flag == 0`, maka akan dianggap sebagai "error".
* Jika `flag == 1`, maka dianggap sebagai "warning".
* Jika `flag == 2`, maka dianggap sebagai "informasi".
* Fungsi kemudian mencetak pesan (`M`) beserta jenis statusnya berdasarkan nilai `flag`.

4. Output:

Program mencetak pesan yang diikuti oleh jenis status (error, warning, atau informasi).

* Cara Kerja Program

1. Program memulai dengan mendeklarasikan variabel `bilangan` dan `pesan`.

2. Fungsi `fmt.Scan` digunakan untuk menerima input dari pengguna. Pengguna memasukkan bilangan bulat (`bilangan`) dan string (`pesan`).

3. Program kemudian memanggil fungsi `cetakpesan`, yang bertanggung jawab untuk menentukan jenis pesan berdasarkan nilai `bilangan` yang dimasukkan.

4. Berdasarkan nilai `bilangan`:

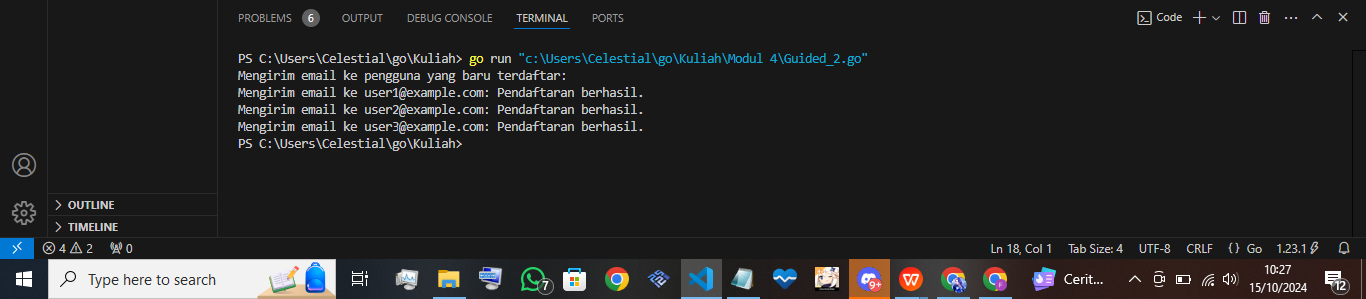
* - 0: Pesan akan dicetak dengan jenis "error".
* - 1: Pesan akan dicetak dengan jenis "warning".
* - 2: Pesan akan dicetak dengan jenis "informasi".

1. Buatlah sebuah program yang dapat mengirim beberapa email pemberitahuan ke pengguna yang baru terdaftar. Email tersebut berisi pesan bahwa pendaftaran telah berhasil. Setiap email akan ditampilkan sebagai output dengan pesan yang sudah diformat.

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func sendEmailNotification(email string) {  fmt.Printf("Mengirim email ke %s: Pendaftaran berhasil.\n", email)  }  func main() {  //daftar email pengguna baru  emails := []string{"user1@example.com", "user2@example.com", "user3@example.com"}  // Mengirim email ke pemberitahuan ke setiap pengguna  fmt.Println("Mengirim email ke pengguna yang baru terdaftar:")  for \_, email := range emails {  sendEmailNotification(email)  }  } |

**Screenshoot Output**

****

**Deskripsi Program**

Program ini mengimplementasikan sebuah skenario di mana beberapa email pemberitahuan dikirim ke pengguna yang baru terdaftar. Email tersebut berisi pesan bahwa pendaftaran telah berhasil. Setiap email akan ditampilkan sebagai output dengan pesan yang sudah diformat.

* Algoritma

1. Fungsi `sendEmailNotification`:

Fungsi ini menerima satu parameter berupa string (`email`), yaitu alamat email pengguna.

Di dalam fungsi, digunakan `fmt.Printf` untuk mencetak pesan berupa "Mengirim email ke [alamat email]: Pendaftaran berhasil."

2. Fungsi `main`:

* Pada fungsi `main`, terdapat sebuah slice `emails` yang berisi daftar email pengguna baru.
* Program mencetak pesan "Mengirim email ke pengguna yang baru terdaftar:".
* Kemudian, program menggunakan loop `for` untuk mengiterasi setiap alamat email di dalam slice `emails`, memanggil fungsi `sendEmailNotification` untuk setiap email.
* Cara Kerja Program

1. Daftar Email:

Program mendeklarasikan sebuah slice yang berisi daftar tiga email pengguna baru.

2. Perulangan `for`:

Program melakukan iterasi pada slice `emails` menggunakan perulangan `for`, dengan setiap iterasi memanggil fungsi `sendEmailNotification` dan mengirimkan alamat email saat itu sebagai argumen.

3. Mengirim Notifikasi Email:

Pada setiap iterasi, fungsi `sendEmailNotification` dipanggil, dan pesan "Mengirim email ke [alamat email]: Pendaftaran berhasil." dicetak ke layar.

* Output dari program ini adalah sebagai berikut:

Mengirim email ke pengguna yang baru terdaftar:

Mengirim email ke user1@example.com: Pendaftaran berhasil.

Mengirim email ke user2@example.com: Pendaftaran berhasil.

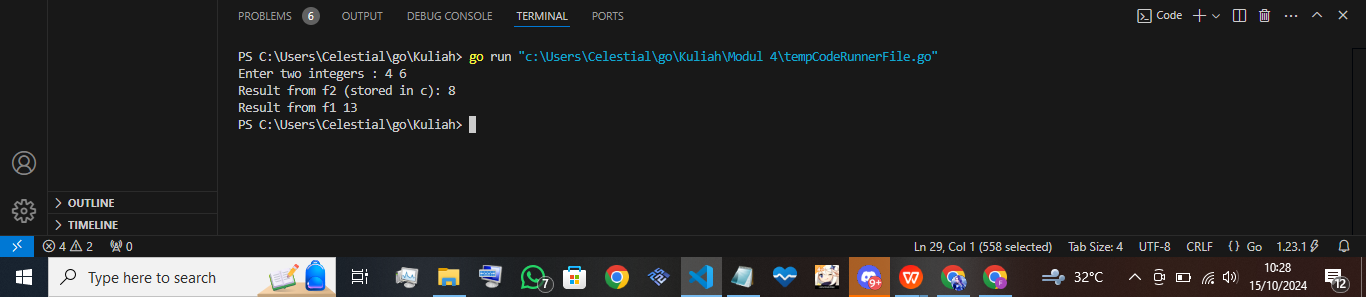
Mengirim email ke user3@example.com: Pendaftaran berhasil.

1. Buatlah program untuk menghitung nilai berdasarkan dua bilangan bulat (`x` dan `y`). Kedua fungsi tersebut melakukan operasi aritmatika yang serupa, namun perbedaan utamanya terletak pada bagaimana mereka menangani hasil (apakah menggunakan \*\*pass by value\*\* atau \*\*pass by reference\*\*).

**Sourcecode**

|  |
| --- |
| package main  import "fmt"  func f1(x, y int) float64 { //pass by value  var hasil float64  hasil = float64(2\*x) - 0.5\*float64(y) + 3.0  return hasil  }  func f2(x, y int, hasil \*float64) { //pass by referenc  \*hasil = float64(2\*x) - 0.5\*float64(y) + 3.0 //pass by referenc  }  func main() {  var a, b int  var c float64  //Take input for a and b  fmt.Print("Enter two integers : ")  fmt.Scan(&a, &b)  f2(a, b, &c)  fmt.Println("Result from f2 (stored in c):", c)  resultF1 := f1(b, a)  fmt.Println("Result from f1", resultF1)  } |

**Screenshoot Output**

****

**Deskripsi Program**

Program ini mendefinisikan dua fungsi, `f1` dan `f2`, untuk menghitung nilai berdasarkan dua bilangan bulat (`x` dan `y`). Kedua fungsi tersebut melakukan operasi aritmatika yang serupa, namun perbedaan utamanya terletak pada bagaimana mereka menangani hasil (apakah menggunakan \*\*pass by value\*\* atau \*\*pass by reference\*\*). Program ini juga meminta input dari pengguna dan menampilkan hasil dari kedua fungsi tersebut.

* Algoritma

1. Fungsi `f1(x, y int) float64`:

* Fungsi ini menerima dua bilangan bulat `x` dan `y` sebagai parameter, dan mengembalikan hasilnya sebagai nilai `float64`.
* Operasi yang dilakukan dalam fungsi adalah:

hasil = float64(2\*x) - 0.5\*float64(y) + 3.0

* Karena fungsi ini menggunakan \*\*pass by value\*\*, nilai-nilai `x` dan `y` disalin dan perubahannya hanya terjadi di dalam ruang lingkup fungsi tersebut, tanpa mempengaruhi variabel asli di luar fungsi.

2. Fungsi `f2(x, y int, hasil \*float64)`:

* Fungsi ini menerima dua bilangan bulat `x` dan `y`, serta pointer ke variabel `float64` (`hasil`).
* Operasi yang dilakukan sama dengan `f1`, tetapi kali ini hasil disimpan dalam variabel yang ditunjukkan oleh pointer `hasil`.
* Menggunakan \*\*pass by reference\*\* memungkinkan fungsi ini untuk mengubah langsung variabel yang ada di luar ruang lingkup fungsi tersebut, karena parameter `hasil` berupa pointer.

3. Fungsi `main`:

* Program meminta input dua bilangan bulat dari pengguna.
* Fungsi `f2` dipanggil terlebih dahulu, menggunakan alamat dari variabel `c` untuk menyimpan hasilnya.
* Kemudian, fungsi `f1` dipanggil, dan hasilnya disimpan dalam variabel `resultF1` dan dicetak.
* Cara Kerja Program

1. Input Pengguna:

Program meminta dua bilangan bulat dari pengguna dan menyimpannya dalam variabel `a` dan `b`.

2. Pemanggilan Fungsi `f2`:

Fungsi `f2` dipanggil dengan parameter `a`, `b`, dan referensi dari variabel `c`. Hasil operasi aritmatika disimpan langsung di variabel `c`.

3. Pemanggilan Fungsi `f1`:

Fungsi `f1` dipanggil dengan parameter `b` dan `a`, dan hasil operasi disimpan di variabel `resultF1`.

4. Output:

Program mencetak hasil dari kedua fungsi, yang dihitung dengan dua cara yang berbeda: \*\*pass by reference\*\* dan \*\*pass by value\*\*.

* Contoh Eksekusi
* Enter two integers : 4 6
* Output program:

Result from f2 (stored in c): 9

Result from f1 10

* Penjelasan
* Pada pemanggilan \*\*`f2(a, b, &c)`\*\*, hasil disimpan dalam variabel `c` secara langsung karena menggunakan \*\*pass by reference\*\*. Nilai `c` berubah di luar fungsi.
* Pada pemanggilan \*\*`f1(b, a)`\*\*, hasil dikembalikan sebagai nilai baru dan disimpan dalam variabel `resultF1`. Ini menggunakan \*\*pass by value\*\*, sehingga tidak ada perubahan pada variabel di luar fungsi tersebut.

1. **UNGUIDED**

**Soal Studi Case**

1. Minggu ini, mahasiswa Fakultas Informatika mendapatkan tugas dari mata kuliah matematika diskrit untuk mempelajari kombinasi dan permutasi. Jonas salah seorang mahasiswa, iseng untuk mengimplementasikannya ke dalam suatu program. Oleh karena itu bersediakah kalian membantu Jonas? (tidak tentunya ya :p)

**Masukan** terdiri dari empat buah bilangan asli a, b, c, dan d yang dipisahkan oleh spast, dengan syarat a ≥ c dan b ≥ d.

**Keluaran** terdiri dari dua baris. Baris pertama adalah hasil permutasi dan kombinasi a terhadap c, sedangkan baris kedua adalah hasil permutasi dan kombinasi b terhadap d.

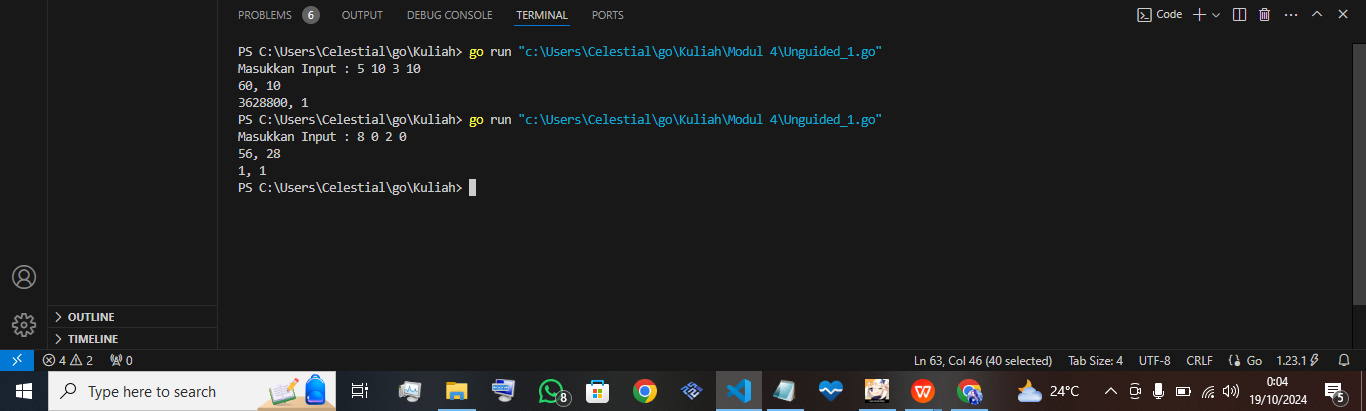
Catatan: permutasi (P) dan kombinasi (C) dari n terhadap r (n ≥ r) dapat dihitung dengan menggunakan persamaan berikut!

P(n,r) = n! / (n - r)!, sedangkan C(n,r) = n ! / r! (n - r)!

**Sourcecode**

|  |
| --- |
| package main  import (  "fmt"  )  func faktorial(n int, result \*int) {  if n < 0 {  \*result = 0  return  }  \*result = 1  for i := 1; i <= n; i++ {  \*result \*= i  }  }  func permutasi(n, r int, result \*int) {  if n < r {  \*result = 0  return  }  var nFact, nrFact int  faktorial(n, &nFact)  faktorial(n-r, &nrFact)  \*result = nFact / nrFact  }  func kombinasi(n, r int, result \*int) {  if n < r {  \*result = 0  return  }  var nFact, rFact, nrFact int  faktorial(n, &nFact)  faktorial(r, &rFact)  faktorial(n-r, &nrFact)  \*result = nFact / (rFact \* nrFact)  }  func main() {  var a, b, c, d int  var permA, combA, permB, combB int  fmt.Print("Masukkan Input : ")  \_, err := fmt.Scan(&a, &b, &c, &d)  if err != nil {  fmt.Print("Input tidak valid, silakan masukkan input.")  return  }  if a >= c && b >= d {  permutasi(a, c, &permA)  kombinasi(a, c, &combA)  fmt.Printf("%d, %d\n", permA, combA)  permutasi(b, d, &permB)  kombinasi(b, d, &combB)  fmt.Printf("%d, %d\n", permB, combB)  } else {  fmt.Println("Syarat tidak terpenuhi: a harus >= c dan b harus >= d")  }  } |

**Screenshoot Output**

****

**Deskripsi Program**

Program ini menghitung \*\*permutasi\*\* dan \*\*kombinasi\*\* untuk dua pasangan bilangan (`a, c` dan `b, d`). Pengguna memasukkan empat angka, dan program akan mengecek apakah syarat `a >= c` dan `b >= d` terpenuhi sebelum menghitung permutasi dan kombinasi. Perhitungan dilakukan dengan bantuan fungsi yang menghitung faktorial.

* Algoritma

1. Fungsi `faktorial`:

* Fungsi ini menerima dua parameter: bilangan bulat `n` dan pointer ke variabel `result` untuk menyimpan hasilnya.
* Jika `n` kurang dari 0, hasil akan diatur menjadi 0.
* Jika `n` >= 0, faktorial dari `n` dihitung menggunakan loop. Hasil perhitungannya disimpan di variabel `result`.

2. Fungsi `permutasi`:

* Fungsi ini menerima tiga parameter: bilangan bulat `n`, `r`, dan pointer ke variabel `result` untuk menyimpan hasilnya.
* Fungsi pertama-tama memeriksa apakah `n < r`. Jika ya, hasilnya adalah 0.
* Jika tidak, fungsi memanggil `faktorial` untuk menghitung faktorial dari `n` dan `(n - r)`, lalu menghitung permutasi menggunakan rumus: “ P (n,r ) = n! /(n - r)! ”
* Hasil disimpan dalam variabel `result`.

3. Fungsi `kombinasi`:

* - Fungsi ini menerima tiga parameter: bilangan bulat `n`, `r`, dan pointer ke variabel `result`.
* - Sama seperti fungsi `permutasi`, fungsi ini memeriksa apakah `n < r`. Jika ya, hasilnya adalah 0.
* - Fungsi kemudian memanggil `faktorial` untuk menghitung faktorial dari `n`, `r`, dan `(n - r)`, lalu menghitung kombinasi menggunakan rumus: “ P (n,r ) = n! / r! \* (n - r)! ”
* Hasil disimpan dalam variabel `result`.

4. Fungsi `main`:

* Program meminta pengguna memasukkan empat bilangan bulat: `a`, `b`, `c`, dan `d`.
* Program kemudian memeriksa apakah syarat `a >= c` dan `b >= d` terpenuhi. Jika ya, fungsi `permutasi` dan `kombinasi` dipanggil untuk menghitung nilai permutasi dan kombinasi dari `a, c` dan `b, d`, lalu hasilnya dicetak.
* Jika syarat tidak terpenuhi, program menampilkan pesan error.
* Cara Kerja Program

1. Input Pengguna:

* Program meminta pengguna memasukkan empat angka: `a`, `b`, `c`, dan `d`.
* Program akan mengecek apakah `a >= c` dan `b >= d` sebelum melakukan perhitungan.

2. Perhitungan Permutasi dan Kombinasi\*\*:

* Jika syarat `a >= c` dan `b >= d` terpenuhi, fungsi `permutasi` dan `kombinasi` dipanggil untuk menghitung nilai dari dua pasangan bilangan (`a, c` dan `b, d`).
* Hasil permutasi dan kombinasi untuk kedua pasangan akan dicetak.

3. Penanganan Error:

* Jika input tidak valid atau syarat `a >= c` dan `b >= d` tidak terpenuhi, program menampilkan pesan error.
* Contoh Eksekusi
* Masukkan Input: 5 6 3 2
* Output program:

60, 10

30, 15

* Penjelasan:
* Untuk pasangan `a = 5` dan `c = 3`:
* Permutasi: \( P(5, 3) = \frac{5!}{(5-3)!} = 60 \)
* Kombinasi: \( C(5, 3) = \frac{5!}{3!(5-3)!} = 10 \)

* Untuk pasangan `b = 6` dan `d = 2`:
* Permutasi: \( P(6, 2) = \frac{6!}{(6-2)!} = 30 \)
* Kombinasi: \( C(6, 2) = \frac{6!}{2!(6-2)!} = 15 \)
* Jika `a < c` atau `b < d`, program akan menampilkan:
* Syarat tidak terpenuhi: a harus >= c dan b harus >= d

1. Kompetisi pemrograman tingkat nasional berlangsung ketat. Setiap peserta diberikan 8 soal yang harus dapat diselesaikan dalam waktu 5 jam saja. Peserta yang berhasil menyelesaikan soal paling banyak dalam waktu paling singkat adalah pemenangnya.

Buat program **gema** yang mencari pemenang dari daftar peserta yang diberikan. Program harus dibuat modular, yaitu dengan membuat prosedur hitungSkor yang mengembalikan total soal dan total skor yang dikerjakan oleh seorang peserta, melalui parameter formal, Pembacaan nama peserta dilakukan di program utama, sedangkan waktu pengerjaan dibaca di dalam prosedur.

**prosedure** hitungskor (in/out soal, skor : integer)

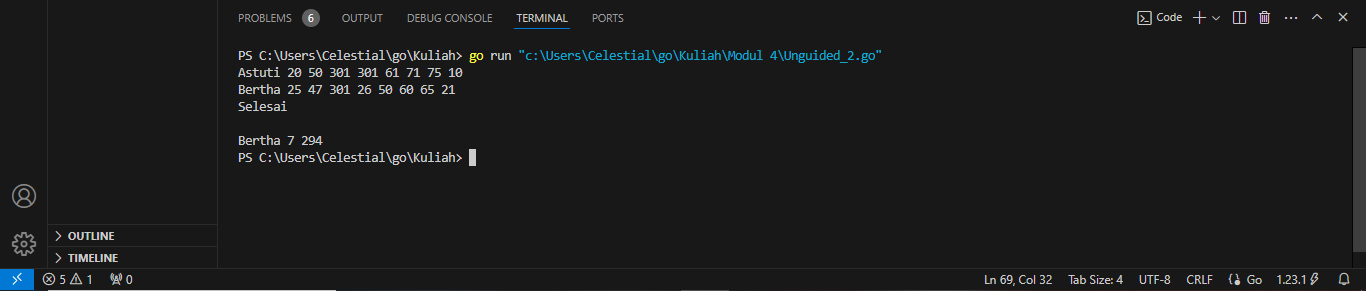
Setiap baris **masukan** dimulai dengan satu string nama peserta tersebut dikuti dengan adalah 8 integer yang menyatakan berapa lama (dalam menit) peserta tersebut menyelesaikan soal. Jika tidak berhasil atau tidak mengirimkan jawatan maka otomatis dianggap menyelesaikan dalam waktu 5 jam I menit (301 meilt).

Satu baris **keluaran** berisi nama pemenang, jumlah soal yang diseleseikan, dan nilai yang diperoleh. Nilai adalah total waktu yang dibutuhkan untuk menyelesaikan soal yang berhasil.

**Sourcecode**

|  |
| --- |
| package main  import (  "bufio"  "fmt"  "os"  "strconv"  "strings"  )  func hitungSkor(waktu []int, soal \*int, skor \*int, waktuTotal \*int) {  \*soal = 0  \*skor = 0  \*waktuTotal = 0  for \_, menit := range waktu {  if menit <= 300 {  \*soal++  \*skor += 300 - menit  \*waktuTotal += menit  }  }  }  func main() {  reader := bufio.NewReader(os.Stdin)  var namaPemenang string  var soalPemenang, waktuTotalPemenang int  selesai := false // Variabel untuk menandakan status selesai  for {  input, \_ := reader.ReadString('\n')  input = strings.TrimSpace(input)  // Cek apakah input adalah "Selesai"  if input == "Selesai" && !selesai {  selesai = true // Set status selesai  break  }  parts := strings.Fields(input)  nama := parts[0]  waktu := make([]int, len(parts)-1)  for i := 1; i < len(parts); i++ {  menit, err := strconv.Atoi(parts[i])  if err == nil {  waktu[i-1] = menit  }  }  var soal, skor, waktuTotal int  hitungSkor(waktu, &soal, &skor, &waktuTotal)  if (namaPemenang == "") || (skor > waktuTotalPemenang) || (skor == waktuTotalPemenang && waktuTotal < waktuTotalPemenang) {  namaPemenang = nama  soalPemenang = soal  waktuTotalPemenang = waktuTotal  }  }  if namaPemenang != "" {  fmt.Println()  fmt.Printf("%s %d %d\n", namaPemenang, soalPemenang, waktuTotalPemenang)  }  } |

**Screenshoot Output**

****

**Deskripsi Program**

Program ini digunakan untuk menghitung skor kompetisi, di mana peserta harus menyelesaikan soal dalam waktu tertentu. Program akan mengambil input berupa nama peserta dan waktu yang dihabiskan untuk menyelesaikan soal. Setelah input ‘selesai’ dimasukkan, program akan mengumumkan pemenang berdasarkan skor yang dihitung dan total waktu yang dihabiskan. Peserta dengan skor tertinggi dan waktu terendah akan menjadi pemenang.

* Algoritma

1. Fungsi `hitungSkor`:

* Fungsi ini menerima tiga parameter:
* `waktu`: slice berisi waktu (dalam menit) yang dihabiskan oleh peserta untuk setiap soal.
* `soal`: pointer ke variabel yang akan menyimpan jumlah soal yang dikerjakan.
* skor`: pointer ke variabel yang akan menyimpan total skor.
* `waktuTotal`: pointer ke variabel yang akan menyimpan total waktu yang dihabiskan.
* Inisialisasi `soal`, `skor`, dan `waktuTotal` menjadi 0.
* Fungsi melakukan iterasi pada slice `waktu`, dan untuk setiap waktu yang kurang dari atau sama dengan 300 menit:

2. Fungsi `main`:

* Menggunakan `bufio.NewReader` untuk membaca input dari pengguna.
* Menginisialisasi variabel untuk menyimpan nama pemenang, jumlah soal yang dikerjakan oleh pemenang, dan total waktu yang dihabiskan oleh pemenang.
* Menggunakan loop untuk membaca input:
* Input adalah nama peserta diikuti dengan waktu yang dihabiskan untuk setiap soal.
* Jika input adalah "Selesai", loop akan berhenti.

* Memanggil fungsi `hitungSkor` untuk menghitung jumlah soal, skor, dan waktu total.
* Membandingkan jumlah soal yang dikerjakan dan waktu peserta pertama dengan peserta yang lain
* Setelah semua input diproses, program mencetak nama pemenang, jumlah soal, dan waktu total yang dihabiskan.
* Cara Kerja Program

1. Input Pengguna:

* Program meminta pengguna untuk memasukkan nama peserta dan waktu yang dihabiskan untuk menyelesaikan soal.
* Input dapat dimasukkan berulang kali sampai pengguna mengetik "Selesai".

2. Menghitung Skor:

Program menghitung skor dan total waktu menggunakan fungsi `hitungSkor`, yang menjumlahkan skor berdasarkan waktu yang dihabiskan dan menghitung jumlah soal yang dikerjakan.

3. Menentukan Pemenang:

Setelah input selesai, program menentukan pemenang berdasarkan skor tertinggi dan waktu terendah.

4. Output:

Program mencetak nama pemenang, jumlah soal yang dikerjakan, dan total waktu yang dihabiskan.

1. Skiena dan Revilla dalam Programming Challenges mendefinisikan sebuah deret bilangan. Deret dimulai dengan sebuah bilangan bulat n. Jika bilangan n saat itu genap, maka suku berikutnya adalah 1/2n, tetapi jika ganjil maka suku berikutnya bernilai 3n+1. Rumus yang sama digunakan terus menerus untuk mencari suku berikutnya. Deret berakhir ketika suku terakhir bernilai 1. Sebagai contoh jika dimulai dengan n=22, maka deret bilangan yang diperoleh adalah:

22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1

Untuk suku awal sampai dengan 1000000, diketahul deret selalu mencapai suku dengan nilai 1.

Buat program sklena yang akan mencetak setiap suku dari deret yang dijelaskan di atas untuk nilai suku awal yang diberikan. Pencetakan deret harus dibuat dalam prosedur cetakDeret yang mempunyai 1 parameter formal, yaitu nilai dari suku awal.

prosedure cetak Deret(in n: integer)

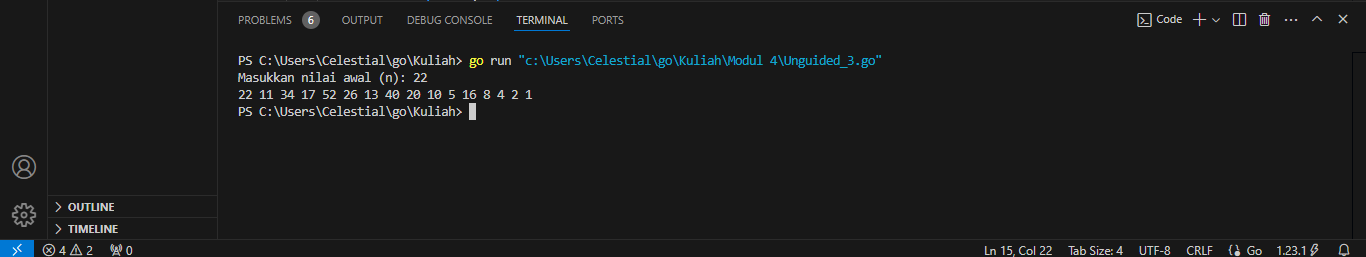
**Masukan** berupa satu bilangan integer positif yang lebih kecil dari 1000000.

**Keluaran** terdiri dari satu baris saja. Setiap sultu dari deret tersebut dicetak dalam baris yang dan dipisahkan oleh sebuah spasi.

**Sourcecode**

|  |
| --- |
| package main  import (  "fmt"  )  func cetakDeret(n int) {  if n <= 0 {  fmt.Println("Masukan harus berupa bilangan integer positif yang lebih besar dari 0.")  return  }  fmt.Printf("%d ", n)  for n != 1 {  if n%2 == 0 {  n = n / 2  } else {  n = 3\*n + 1  }  fmt.Printf("%d ", n)  }  fmt.Println()  }  func main() {  var n int  fmt.Print("Masukkan nilai awal (n): ")  \_, err := fmt.Scan(&n)  if err != nil {  fmt.Println("Error: Masukan tidak valid.")  return  }  if n > 1000000 {  fmt.Println("Error: Nilai maksimum yang diperbolehkan adalah 1000000.")  return  }  cetakDeret(n)  } |

**Screenshoot Output**

****

**Deskripsi Program**

Program ini menghitung dan mencetak deret berdasarkan dua aturan:

* Jika bilangan genap, bagi dengan 2.
* Jika bilangan ganjil, kalikan dengan 3 dan tambahkan 1.
* Proses ini diulang hingga mencapai angka 1.
* Algoritma

1. Fungsi `cetakDeret`:

* Menerima satu parameter `n`, yang merupakan bilangan bulat positif.
* Jika `n` kurang dari atau sama dengan 0, fungsi akan mencetak pesan kesalahan dan keluar.
* Program mencetak nilai `n`, lalu memulai loop yang berlanjut hingga `n` menjadi 1.
* Jika `n` genap (`n % 2 == 0`), nilai `n` dibagi dengan 2.
* Jika `n` ganjil (`n % 2 != 0`), nilai `n` diperbarui menjadi `3\*n + 1`.
* Setiap langkah akan mencetak nilai `n` yang baru hingga loop selesai.

2. Fungsi `main`:

* Menggunakan `fmt.Print` untuk meminta pengguna memasukkan nilai awal `n`.
* Program memeriksa apakah input valid:
* Jika input tidak valid (misalnya, bukan bilangan), program mencetak pesan kesalahan dan keluar.
* Jika `n` lebih dari 1.000.000, program juga akan mencetak pesan kesalahan dan keluar.
* Jika semua pemeriksaan validasi berhasil, fungsi `cetakDeret` dipanggil untuk menghasilkan dan mencetak deret berdasarkan nilai `n` yang diberikan.
* Cara Kerja Program

1. Input Pengguna:

Program meminta pengguna untuk memasukkan bilangan bulat positif sebagai nilai awal `n`.

2. Validasi Input:

Program memeriksa apakah input valid, yaitu:

* Harus berupa bilangan bulat positif.
* Tidak boleh lebih dari 1.000.000.

3. Menghitung Deret Collatz:

Jika input valid, program akan memanggil fungsi `cetakDeret`, yang menghasilkan deret Collatz berdasarkan nilai `n`.

4. Output:

Program mencetak deret angka dari nilai awal `n` hingga mencapai 1.