**LAPORAN PRAKTIKUM  
PEMROGRAMAN BERORIENTASI OBJEK**

**MODUL II**

**“REVIEW STRUKTUR KONTROL”**



**Oleh:**

**MUHAMMAD RAGIEL PRASTYO**

**2311102183**

**S1IF-11-02**

**S1 TEKNIK INFORMATIKA**

**TELKOM UNIVERSITY PURWOKERTO**

**2024**

1. **DASAR TEORI**

**2.1 Struktur Program Go**

Dalam kerangka program yang ditulis dalam bahasa pemrograman Go, program utama selalu mempunyai dua komponen berikut:

* Package main merupakan penanda bahwa file ini berisi program utama
* func main() berisi kode utama dari sebuah program Go Komentar, bukan bagian dari kode program dan dapat ditulis dimana saja di dalam program:
* satu baris teks yang diawali dengan garis miring ganda (//) s.d. akhir baris, atau
* beberapa baris teks yang dimulai dengan pasangan karakter '/*' dan di akhiri dengan '*/'.

1. **Koding, Kompilasi, dan Eksekusi Go**

**Koding**

* Tidak berbeda dengan penulisan program sumber dalam bahasa lain, program Go harus dibuat dengan menggunakan penyuntingan teks dan disimpan dalam format teks, bukan dalam format dokumen (doc, docx, atau lainnya).
* Setiap program go disimpan dalam file teks dengan ekstensi \*.go, dengan nama bebas. Sebaiknya nama file adalah nama untuk program tersebut.
* Setiap satu program lengkap Go disimpan dalam satu folder tersendiri. Nama folder merupakan nama program tersebut. Karena itu secara prinsip, satu program Go dapat dipecah dalam beberapa file dengan esktensi \*.go selama disimpan dalam folder yang sama.

**Kompilasi**

Beberapa bahasa pemrograman dibangun untuk digunakan sebagai interpreter, sedangkan yang lain dimaksudkan untuk dikompilasi. Interpreter akan membaca setiap baris instruksi dan kemudian langsung melakukannya, hanya untuk memeriksa apakah penulisan program secara keseluruhan sudah benar atau belum. Setelah memeriksa program sumber secara keseluruhan, compiler mengubahnya menjadi program eksekutabel. Ini memastikan bahwa konsistensi penulisan, seperti penggunaan tipe data, telah dipastikan sebelum program dieksekusi. Program juga dapat dioptimalkan karena dibuat menjadi eksekutabel lebih awal.

Go berfungsi sebagai kompilator. Saat mengkompilasi dan mengeksekusi

program dalam bahasa Go, berikut adalah beberapa sesi yang biasa dilakukan:

* Panggil shell atau terminal (program/utiliti cmd.exe di windows).
* Masuk ke dalam (cd) folder program (normalnya ada di C:\Users\go\src\ atau yang sejenisnya).
* Kemudian panggil perintah go build atau go build file.go untuk mengkompilasi file.go.
* Jika gagal, akan muncul pesan eror yang sesuai, pelajari dengan baik pesan tersebut, perbaiki teks program sumber, kemudian ulangi proses build-nya.
* Jika berhasil, maka pada folder tersebut akan dibuat program dengan nama yang sama dan diakhiri dengan .exe (untuk windows).
* Panggil progra, eksekutabel tersebut dari terminal yang sama. Jangan memanggil program tersebut dengan mengklik eksekutabel tersebut dari folder karena program kalian hanya berbasis teks, bukan/belum dirancang dengan tampilan windows.

**Catatan!!!**

Semua proses terkait bahasa Go dilakukan melalui utilitas go. Beberapa opsi dengan utilitas go:

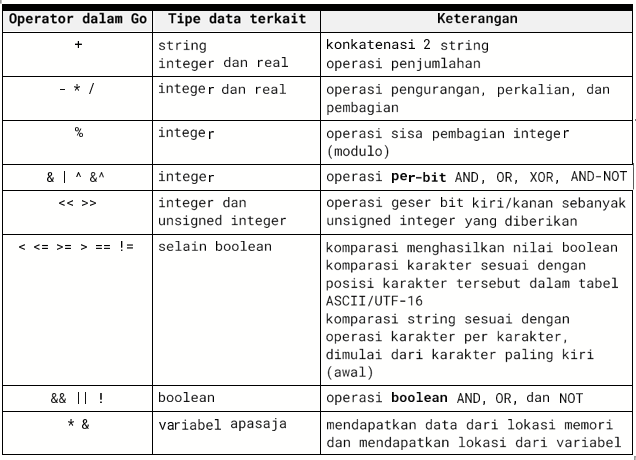
* go build: mengkompilasi program sumber yang ada dalam folder menjadi sebuah program.
* go build file.go: mengkompilasi program sumber file.go saja.
* go fmt: membaca semua program sumber dalam folder dan mereformat penulisannya agar sesuai dengan standar penulisan program sumber Go.
* go clean: membersihkan file-file dalam folder sehingga tersisa program sumbernya saja.

**2.2 Tipe Data dan Instruksi Dasar**

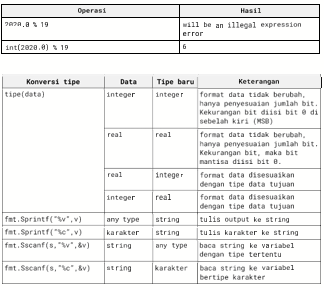
1. **Data dan Variabel**

Variabel adalah nama dari suatu lokasi di memori, yang data dengan tipe tertentu dapat disimpan

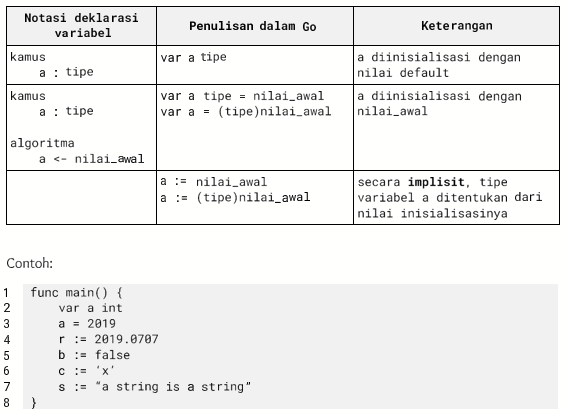
* Nama variabel dimulai dengan huruf dan dapat diikuti dengan sejumlah huruf, angka, atau garis bawah.
* Tipe data yang umum tersedia adalah integer, real, boolean, karakter, dan string. Lihat tabel berikut ini untuk variasi tipe data yang disediakan dalam bahasa Go.
* Nilai data yang tersimpan dalam variabel dapat diperoleh dengan menyebutkan langsung nama variabelnnya contoh: menyebutkan nama found akan mengambil nilai tersimpan dalam memori untuk variabel found, pastinya.
* Informasi alamat atau lokasi dari variabel dapat diperoleh dengan menambahkan prefiks & di depan nama variabel tersebut contoh: &found akan mendapatkan alamat memori untuk menyimpan data pada found.
* Jika variabel berisi alamat memori, prefiks \* pada variabel tersebut akan memberikan nilai yang tersimpan dalam memori yang lokasinya disimpan dalam variabel tersebut contoh: \*mem akan mendapatkan data di memori yang alamatnya tersimpan di mem, karenanya \*(&found) akan mendapatkan data dari lokasi memori variabel found berada, alias sama saja dengan menyebutkan langsung found 8=).
* Operasi yang dapat dilakukan terhadap tipe data diatas adalah



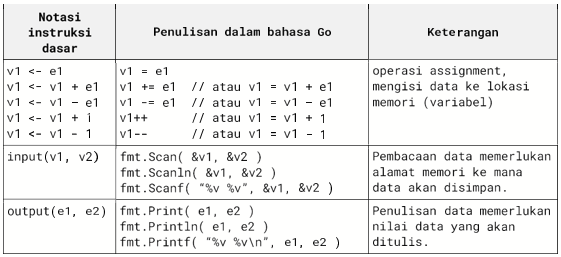
* Bahasa Go menganut kesesuaian tipe data yang ketat. Tipe data yang berbeda tidak boleh dicampur dalam satu ekspresi, bahkan tipe data masih yang sejenis, misalnya masih sama-sama integer (int dan int32). Untuk menyesuaikan tipe data, ada beberapa cara yang dapat dilakukan:
* Casting, tipe (data), mengubah tipe dari data yang diberikan ke tipe yang diinginkan.
* Memanfaatkan fungsi Sprint dan Sscan dari paket fmt.
* Memanfaatkan fungsi-fungsi dalam paket strconv, seperti Atol, Itoa, dan ParseBool. Lihat lampiran berikut untuk contoh penggunaan.



* Variabel harus dideklarasikan dulu sebelum digunakan. Variabel juga harus diinisialisasi dulu (diisi data) agar nilai yang tersimpan diketahui dengan jelas dan eksekusi algoritma menjadi terprediksi. Dalam bahasa Go, variabel yang tidak diinisialisasi lebih dulu otomatis diisi dengan nilai default yang ekuivalen dengan bit 0.
* Nilai 0 untuk bilangan integer.
* 0.0E+0 untuk bilangan real
* false untuk boolean
* karakter NUL (lihat tabel ASCII) untuk karakter
* ''''(string kosong, string dengan panjang 0) untuk string
* nill untuk alamat memori



1. **Instruksi Dasar**



1. **Konstanta Simbolik**

Konstanta dapat diberi nama untuk memudahkan mengingat maksud dan manfaat dari nilai yang diberi nama tersebut.

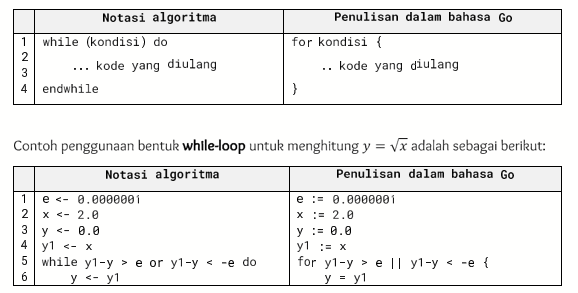


**2.3 Struktur Kontrol Perulangan**

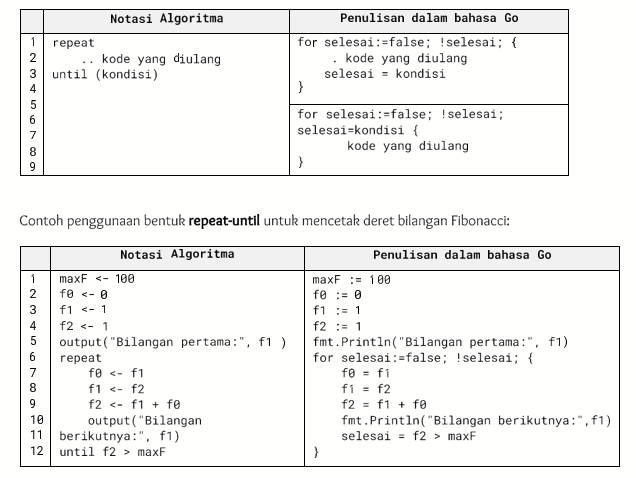
Go hanya mempunyai kata kunci for untuk semua jenis perulangan yang kita pelajari dalam kondisi notasi algoritma. Dua bentuk yang kita gunakan disini adalah struktur while-loop dan repeat-until. Dalam konsep pemrograman terstruktur, setiap rancangan algoritma harus memenuhi syarat satu pintu masuk dan satu pintu keluar. Karena itu tidaklah diperkenankan untuk membuat program sumber yang mempunyai struktur loop yang mempunyai pintu keluar lebih dari satu, seperti:

* Satu pintu keluar dari kondisi for dan satu lagi dari instruksi if-break.
* Atau mempunyai isntruksi if-break yang lebih dari satu.

1. Bentuk While-Loop Bentuk While-Loop memastikan setiap kali memasuki loop, ada kondisi yang harus terpenuhi (benar/true). Ini juga berarti saat keluar dari loop, maka nilai kondisi tersebut pasti salah/false!



1. Bentuk Repeat-Until Bentuk repeat-until di perulangan dilakukan terus menerus sampai kondisi keluar terpenuhi. Artinya selama kondisi belum terpenuhi (salah/false) maka perulangan akan terus dilakukan. Pada saat keluar dari loop maka nilai kondisi pasti benar/true!

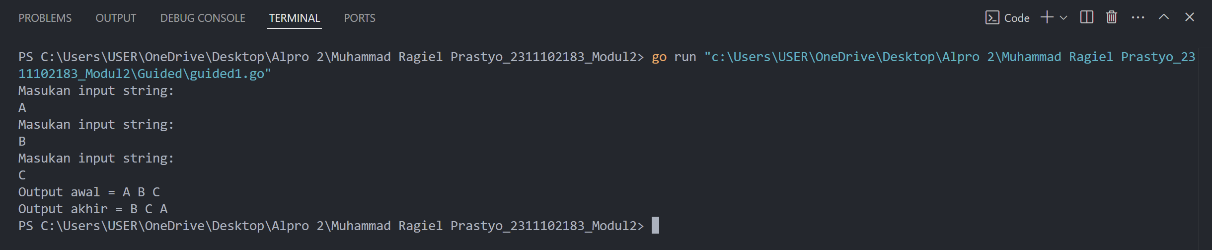


1. **GUIDED**
   * + 1. Telusuri program berikut dengan cara mengkompilasi dan mengeksekusi program. Silahkan masukkan data yang sesuai sebanyak yang diminta program. Perhatikan keluaran yang diperoleh. Coba terangkan apa sebenarnya yang dilakukan program tersebut?

*Source Code*

|  |
| --- |
| package main  import "fmt"  func main() {      var (          satu, dua, tiga string          temp string      )      fmt.Println("Masukan input string: ")      fmt.Scanln(&satu)      fmt.Println("Masukan input string: ")      fmt.Scanln(&dua)      fmt.Println("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)  } |

*Screenshot Program*



**Penjelasan:**

String yang dimasukkan oleh pengguna diminta oleh program untuk ditampilkan sebelum dan sesudah penukaran. Program mengambil input setelah mendeklarasikan variabel dan menampilkan "Output awal". Kemudian, nilai diputar: satu ke dua, dua ke tiga, dan nilai awal satu ke tiga menggunakan variabel sementara temp. Akhirnya, hasil rotasi ditampilkan dalam "Output akhir". Misalnya input A B C akan menghasilkan output akhir B C A.

* + - 1. Tahun kabisat adalah tahun yang habis dibagi 400 atau habis dibagi 4 tetapi tidak habis dibagi 100. Buatlah sebuah program yang menerima input sebuah bilangan bulat dan memeriksa apakah bilangan tersebut merupakan tahun kabisat (true) atau bukan (false). (Contoh input/output, teks bergaris bawah adalah input dari user)

*Source Code*

|  |
| --- |
| package main  import "fmt"  func main() {      var year int      fmt.Print("Masukkan tahun: ")      fmt.Scan(&year)      if (year%400 == 0) || (year%4 == 0 && year%100 != 0) {          fmt.Println("Kabisat:", true)      } else {          fmt.Println("Kabisat:", false)      }  } |

*Screenshot Program*

**

**Penjelasan:**

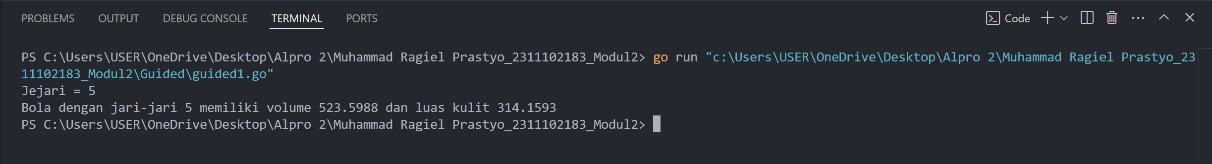
Untuk menentukan apakah suatu tahun adalah tahun kabisat, program di atas ditulis dalam bahasa Go. Untuk menyimpan data, program mendefinisikan variabel tahun dan meminta pengguna memasukkan tahun tersebut. Setelah itu, program memeriksa apakah tahun tersebut kabisat dengan dua kondisi: jika tahun tersebut dibagi 400, atau jika tahun tersebut dibagi 4 tetapi tidak habis dibagi 100. Program menampilkan output Kabisat “true” jika salah satu kondisi terpenuhi. Jika tidak, program menampilkan output Kabisat “false”. Oleh karena itu, program ini menunjukkan tahun kabisat***.***

* + - 1. Buat program Bola yang menerima input jari-jari suatu bola (bilangan bulat). Tampilkan volume dan luas kulit bola. volume bola = 4/3 phi r3 dan luas bola = 4 phi r2 (phi = 3.14) (contoh input/output, teks bergaris bawah adalah input dari user)

*Source Code*

|  |
| --- |
| package main  import (          "fmt"          "math"  )  func main() {          var jariJari int          fmt.Print("Jejari = ")          fmt.Scan(&jariJari)          volume := (4.0 / 3.0) \* math.Pi \* float64(jariJari\*jariJari\*jariJari)          luas := 4 \* math.Pi \* float64(jariJari\*jariJari)          fmt.Printf("Bola dengan jari-jari %d memiliki volume %.4f dan luas kulit %.4f\n", jariJari, volume, luas)  } |

*Screenshot Program*

**

**Penjelasan:**

Program ini ditulis dalam bahasa Go dan menggunakan jari-jari pengguna untuk menghitung volume dan luas kulit bola. Setelah menerima input jari-jari, program menghitung volume dengan rumus 4/3 πr3 dan luas dengan rumus 4 πr2. Hasilnya ditampilkan dalam format yang jelas, menunjukkan jari-jari, volume, dan luas kulit bola.

1. **UNGUIDED**
   * + 1. Buatlah sebuah program yang menerima input berupa warna dari ke 4 gelas reaksi sebanyak 5 kali percobaan. Kemudian program akan menampilkan true apabila urutan warna sesuai dengan informasi yang diberikan pada paragraf sebelumnya, dan false untuk urutan warna lainnya.

*Source Code*

|  |
| --- |
| // Muhammad Ragiel Prastyo  // IF-11-02  // 2311102183  package main  import (      "fmt"      "strings"  )  func main() {      var ujiKimia = [5][4]string{}      var warna = [4]string{"merah", "kuning", "hijau", "ungu"}      var beda bool = true      for i := 0; i < 5; i++ {          fmt.Print("Percobaan ", i+1, ": ")          fmt.Scan(&ujiKimia[i][0], &ujiKimia[i][1], &ujiKimia[i][2], &ujiKimia[i][3])      }      for i := 0; i < 5; i++ {          for j := 0; j < 4; j++ {              beda = strings.ToLower(ujiKimia[i][j]) == warna[j]              if !beda {                  i = 5                  break              }          }      }      fmt.Println("Berhasil : ", beda)  } |

*Screenshot Program*



**Penjelan:**

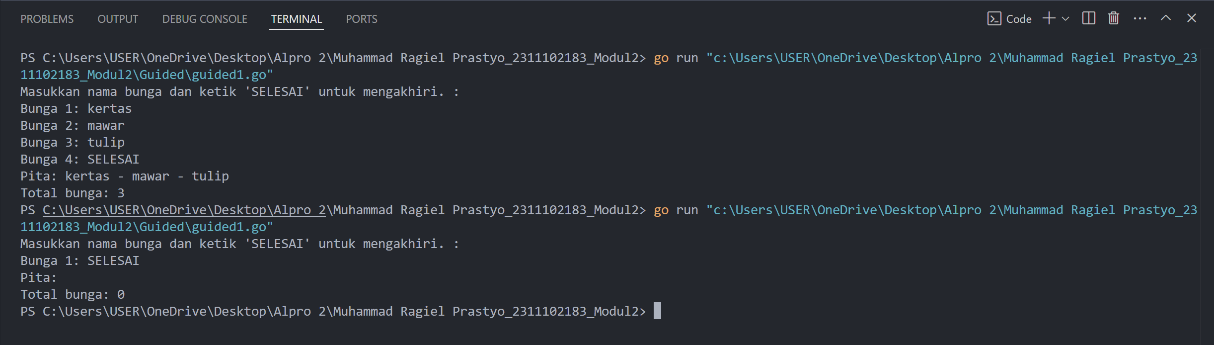
Program ini dibuat dalam bahasa Go untuk menilai hasil eksperimen kimia berdasarkan warna yang dimasukkan pengguna. Setelah menentukan array yang akan menyimpan hasil eksperimen dan warna yang diharapkan (merah, kuning, hijau, dan ungu), program meminta pengguna memasukkan data untuk lima eksperimen. Selanjutnya, program memeriksa warna yang dimasukkan dan yang diharapkan. Jika semua warna cocok, variabel perbedaan tetap “benar”. Jika tidak, nilainya menjadi “salah”. Hasil akhir menunjukkan apakah percobaan tersebut berhasil atau tidak.

* + - 1. Buatlah sebuah program yang menerima input sebuah bilangan bulat positif (dan tidak nol) N, kemudian program akan meminta input berupa nama bunga secara berulang sebanyak N kali dan nama tersebut disimpan ke dalam pita.

*Source Code*

|  |
| --- |
| // Muhammad Ragiel Prastyo  // IF-11-02  // 2311102183  package main  import (      "bufio"      "fmt"      "os"      "strings"  )  func main() {      var pita string      var TotalBunga int      scanner := bufio.NewScanner(os.Stdin)      fmt.Print("Masukkan nama bunga dan ketik 'SELESAI' untuk mengakhiri. :\n")      for {          TotalBunga++          fmt.Printf("Bunga %d: ", TotalBunga)          scanner.Scan()          input := scanner.Text()          if strings.ToUpper(input) == "SELESAI" {              TotalBunga--              break          }          if pita == "" {              pita = input          } else {              pita += " - " + input          }      }      fmt.Println("Pita:", pita)      fmt.Println("Total bunga:", TotalBunga )  } |

*Screenshot Program*



**Penjelasan:**