2 Pengantar Pemrograman Go

Objektif :

- **Mengetahui apa itu Go**
- **Mengenal lingkungan kerja Go dan struktur program Go**
- **Mengetahui tipe data dan variabel di Go**
- **Mampu membuat program, mengkompile dan menjalankan menggunakan sederhana**

2.1. Apa itu Go ?

Dimulai pada tahun 2009 oleh Google dan dirilis versi 1.0 pada tahun 2012, Go adalah bahasa pemrograman yang dikompilasi. Go dikompilasi secara statis (seperti C, C++, C #, Java). Go dikompilasi sangat cepat dan memiliki beberapa kesamaan dengan C. Go pada awalnya dikembangkan pada platform Linux.

Mengembangkan Program Go

Saat ini tidak ada IDE terbaik untuk Go, baik pada sistem operasi Windows, Linux atau Mac OS X . Ada dua free IDE yang dapat digunakan untuk membuat program Go, yaitu:

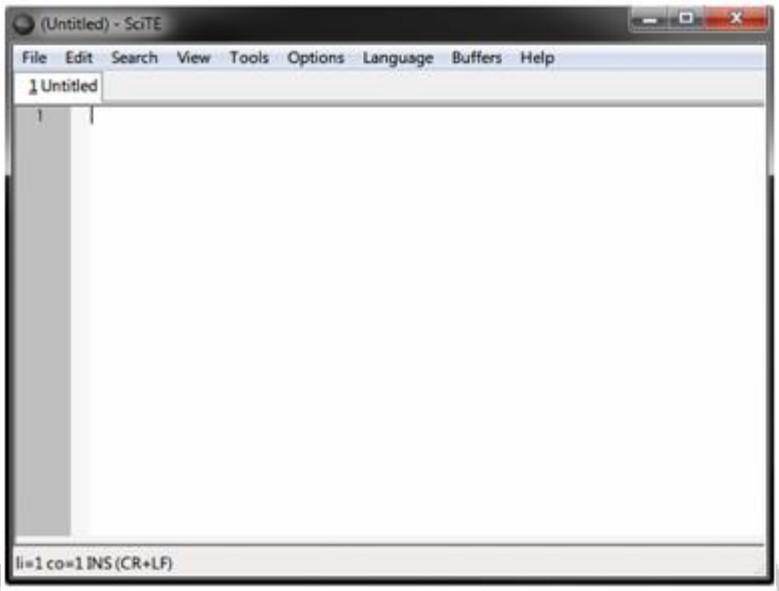
1. Golangide, yaitu sebuah IDE open source yang ditulis dalam C++.
2. Plugin Eclipse (dapat digunakan pada sistem operasi Windows atau Mac OS X, tapi tidak untuk Linux) yang disebut Goclipse dengan sintaks highlighting, autocomplete, pelaporan kesalahan di Eclipse.

Untuk pengguna Windows (dan Ubuntu di bawah Wine) ada versi komersialnya yaitu Bahasa IDE Zeus Go, seharga sekitar $ 99.

Teks Editor

Tool utama seorang programmer dalam menulis perangkat lunak adalah teks editor. Teks editor mirip dengan program word processing (MS Word, OpenOffice, ...) tetapi tidak seperti program tersebut, teks editor tidak mempunyai format (tidak ada bold, italic, ...), teks editor adalah plain text. Untuk membuat instalasi perangkat lunak lebih mudah, installer tersedia di http://www.golang-book.com/. Installler akan menginstall tool Go, setupa variabel environmental dan menginstall teks editor.

Untuk Windows, installer akan menginstall Scite teks editor. Pilih Start – All Program – Go – Scite. Makan akan tampak tampilan seperti berikut :



Gambar 2.1. Teks editor Scite

Teks editor berisi teks area berwarna putih dimana user dapat mengetikkan program disitu. Bagian kiri dari teks area, akan terlihat line number. Bagian bawah dari window akan tampak status bar yang menampilkan informasi mengenai file dan lokasi kursor saat ini (saat ini ada di baris 1 kolom 1).

2.2. Program Go Sederhana

Program pertama yang biasanya dibuat pada beberapa bahasa pemrograman lain disebut program 'Hello World', program sederhana yang akan mencetak "Hello World". Pembuatan program Hello World di Go:

```
package main

import "fmt"

// this is a comment

func main() {
    fmt.Println("Hello World")
}
```

Berinama program diatas hello.go

Kompilasi dan Menjalankan Hello World di Go

Jika menggunakan My Eclipse atau goclipse, klik panah hijau untuk menjalankannya atau jika melalui dari baris perintah (command line terminal di Linux), kompile dan jalankan dengan mengetikkan perintah :

```
go run hello.go
```

Mengapa Go begitu cepat ?

Tidak seperti C atau C++, Go menggunakan paket yang bukan file header sehingga tidak harus menyertakan banyak file. Beberapa mirip dengan fitur Pascal seperti deklarasi variabel. Program Pascal dapat dikompilasi dalam satu tahap. Go memiliki keuntungan dari bahasa yang dikompilasi dengan "edit/compile/run".

Membaca Program Go

Program Go dibaca dari atas ke bawah dan dari kiri ke kana, baris pertama tertulis :

```
package main
```

Ini dikenal sebagai ‘package declaration’. Setiap program Go harus dimulai dengan sebuah paker deklarasi. Package adalah cara Go mengorganisasikan code. Ada dua tipe program Go : executable dan libraries. Aplikasi executable adalah jenis program yang dapat dijalankan secara langsung dari terminal (pada windows berekstension .exe). Libraries adalah koleksi dari code yang dipaket bersama-sama sehingga dapat digunakan di program lain.

Baris berikutnya adalah baris kosong. Komputer merepresentasikan baris baru dengan karakter khusus (atau beberapa karakter). Baris baru, spasi dan tab dikenal dengan white space (karena kita tidak dapat melihatnya). Go tidak peduli dengan whitespace, penggunaan dalam program hanya untuk memudahkan membacanya. Kemudian berikutnya adalah :

```
import "fmt"
```

Keyword import adalah bagaimana kita memasukkan code dari paket lain untuk digunakan pada program kta. Paket fmt (format) adalah untuk pemformatan input dan output. fmt ada dalam tanda “. Package ini memberikan fungsi input dan output yang sama dengan scanf dan printf di C. Penggunaan “ seperti ini dikenal sebagai literal string yang merupakan tipe dari ekspresi. Di Go, string direpresentasikan sebagai kumpulan karakter (huruf, angka, simbol, ...) dengan panjang tertentu. Karekter “ sendiri bukan merupakan bagian dari string.

Baris yang dimulai dengan // adalah sebuah komentar. Komentar diabaikan oleh Go compiler. Komentar di Go sama seperti komentar di C++ dan C99. Satu baris menggunakan / / dan untuk komentar yang lebih dari satu baris di mulai dengan /* dan diakhiri dengan */.

```
// Sebuah komentar baris tunggal di Go
/ * Komentar Go ini
    tersebar di
    tiga baris * /
```

Setelah itu lihar deklarasi fungsi :

```
func main() {
    fmt.Println("Hello World")
}
```

Fungsi adalah bangunan blok-blok dari program Go, yang mempunyai input, output dan sederetan langkah yang disebut statement yang akan dijalankan. Semua fungsi dimulai dengan keyword func diikuti dengan nama fungsi (dalam hal ini main), deretan parameter atau tanpa paremeter yang ditulis dalam tanda ( ), Fungsi main diatas tidak mempunya parameter, tidak mengembalikan apapun dan hanya mempunyai satu statement. Nama main adalah khusus karena fungsi ini yang akan dipanggil saat program dijalankan. Bagian akhir dari baris program diatas adalah :

```
func main() {
    fmt.Println("Hello World")
}
```

Statement ini terdiri dari beberapa komponen. Pertama kita mengakses fungsi lain dalam paket fmt yang disebut Println (yang berarti Print Line). Kemudian buat string baru yang berisi Hello World.

2.3. Tipe Data di Go

Tipe data mengkategorikan seperangkat nilai yang terkait, menentukan operasi yang dapat dilakukan dan mendefinisikan bagaimana disimpan. Go mempunyai beberapa tipe data, yaitu :

2.3.1.NUMBER

Go mempunyau beberapa tipe data berbeda untuk merepresentasikan bilangan. Umumnya bilangan dibagi menjadi dua jenis berbeda yaitu integer dan floating point.

2.3.1.1. Integer

Integer adalah bilangan tanpa komponen desimal (..., -3, -2, -1, 0, 1, 2, 3,...). Tidak seperti sistem bilangan basis 10 yang kita digunakan untuk merepresentasikan bilangan, komputer menggunakan sistem bilangan basis 2.

Sistem kita terbentuk dari 10 digit berbeda. Jika ingin merepresentasikan bilangan lebih besar makan gunakan 2 (kemudian 3, 4, 5, ...) digit yang diletakkan setelahnya. Sebagai contoh setelah 9 adalah 10, bilangan setelah 99 adalah 100 dan seterusnya. Komputer melakukan hal yang sama, tapi komputer hanya mempunyai dua digit yang berisi 10, ... 0, 1, 10, 11, 100, 101, 110, 111 dan seterusnya. Perbedaan lain antara sistem bilangan yang kita gunakan dengan komputer adalah bahwa semua tipe integer mempunyai ukuran tertentu. Integer hanya mempunyai ruang untuk jumlah digit tertentu. Empat bit integer hanya menampung : 0000, 0001, 0010, 0011, 0100.

Tipe integer di Go adalah : uint8, uint32, uint64, int8, int16 dan int64. 8, 16, 32 dan 64 menyatakan berapa bit yang digunakan. Uint bermakna “unsigned integer”, sementara int artinya “signe integer”. Unsigned integer hanya berisi bilangan positif (atau nol). Ada dua tipe alias : byte yang sama dengan uint8 dan rune yang sama dengan int32. Byte adalah unit pengukuran umum yang digunakan komputer (1 byte = 8 bit, 1024 byte = 1 kilobyte, 1024 kilobyte = 1 megabyte, ...) dan karena itu tipe data byte di Go sering digunaan dalam pendefinisian tipe data lain. Ada 3 tipe machine dependent integer yaitu : uint, int dan uintptr, yang ukurannya bergantung pada tipe dari arsitektur yang digunakan. Biasanya jika bekerja dengan integer, gunakan tipe int.

2.3.1.2. Floating Point Number

Floating point number adalah bilangan yang berisi komponen desimal (real number) (1.234, 123.4, 0.00001234, 12340000). Beberapa hal yang perlu diingat adalah :

1. Floating point number adalah inexact. Sebuah bilangan dekat ke nilai yang kita inginkan tapi tidak sepenuhnya sama. Sebagai contoh hasil komputasi 1.01 – 0.99 adalah 0.20000000000000018.
2. Seperti integer, floating point mempunyai ukuran tertentu (32 atau 64 bit). Dengan menggunakan ukurang yang lebih besar, ketelitian/presisi floating point akan bertambah.
3. Ada beberapa nilai lain dari number yang direpresentasikan : “not a number” (NaN, untuk sesuatu seperti 0/0) serta positif dan negatif tak terbatas.

Go mempunyai dua tipe floating point : float32 dan float64 (sering juga disebut sebagai single dan double precision) sebagai dua tipe tambahan untuk merepresentasikan bilangan kompleks (bilangan dengan bagian imajiner) : complex64 dan complex128. Umumnya digunakan float64 saat bekerja dengan floating point. Contoh : Buat program menggunakan angka dengan nama main.go seperti dibawah ini :

```
package main

import "fmt"

func main() {
    fmt.Println("1 + 1 =", 1 + 1)
}
```

Jika program dijalankan akan terlihat :

```
$ go run main.go
1 + 1 = 2
```

Program berisi baris package, baris import yang sama, deklarasi fungsi dan penggunaan fungsi println yang sama. Program ini mencetak string 1 + 1 = diikuti dengan hasil dari ekspresi 1 + 1. Ekspresi ini terdiri dari tiga bagian : numeric literal 1 (bertipe int), operator + (merepresentasikan penjumlahan) dan literal numerik lain yaitu 1. Coba dengan cara yang sama menggunakan floating point :

```
    fmt.Println("1 + 1 =", 1.0 + 1.0)
```

Penggunaan .0 menyatakan pada Go bahwa tipe datanya adalah floating point. Go mempunyai beberapa operator :

+	addition
-	subtraction
*	multiplication
/	division
%	remainder

2.3.2. STRING

Deretan karakter dengan panjang tertentu digunakan untuk merepresentasikan teks. String di Go terbentuk dari individual byte, biasanya satu untuk setiap karakter. String literal dapat dibuat menggunakan double quote “Hello World” atau back ticks ‘Hello World’. Perbedaannya adalah string double quote tidak dapat berisi newline dan memungkinkan spesial escape. Sebagai contoh \n menggantikan dengan newline dan \t menggantikan tab karakter.

```
package main

import "fmt"

func main() {
    fmt.Println(len("Hello World"))
    fmt.Println("Hello World"[1])
    fmt.Println("Hello " + "World")
}
```

Hal yang perlu diingat :

1. Space dianggap satu karakterm sehingga panjang string pada baris tiga adalah 11 bukan 10
2. String diindex ulai dari 0 bukan 1. [1] berarti elemen ke-2 bukan elemen ke-1.
3. Penggabungan menggunakan simbol yang sama seperti penjumlahan. Go compiler mencari tahu apa yang dilakukan berdasarkan pada tipe dari argumen. Karena kedua sisi dari + adalah string, maka compiler melakukan proses penggabungan bukan penjumlahan.

String dapat dideklarasikan dengan beberapa cara :

```
var b String
b = " string "
a : = " Halo , di sini adalah "
fmt.Println (a , b )
```

Array

Array adalah urutan nomor elemen dari satu jenis tipe data. Misalnya 10 int atau 5 string. Ukurannya tetap pada saat kompilasi dan tidak bisa berubah . Sintaksnya adalah [size] type data. Contoh:

```
var (
   f [10] int
   g [5] string
 )

 f [0] = 8
 g [1] = "Test"
 fmt.Println (f[0], g[1])
```

Outputnya :

```
8 Test
```

Array dimulai dari 0, jadi array f dalam contoh memiliki elemen f[0] .. f[9]. Seperti halnya dengan C dan C + + , sebuah array dua dimensi dinyatakan dengan :

```
 var (
  t [20] [5] int
 )

 t [19] [4] = 9
 fmt.Println (t[19] [4])
```

Menghasilkan output :

```
9
```

2.3.3. BOOLEAN

Nilai boolean adalah spesial 1 bit tipe integer digunakan untuk merepresentasikan true dan false (atau on dan off). Tiga operator logika digunakan dengan nilai boolean.

&&	and
||	or
!	not

Berikut adalah contoh program yang menunjukkan bagaimana boolean digunakan :

```
func main() {
    fmt.Println(true && true)
    fmt.Println(true && false)
    fmt.Println(true || true)
    fmt.Println(true || false)
    fmt.Println(!true)
}
```

Menjalankan program diatas, akan menghasilkan output :

```
$ go run main.go
true
false
true
true
false
```

Biasanya digunakan tabel kebenaran untuk mendefinisikan bagaimana operator-operator ini bekerja :

Expression	Value
true && true	true
true && false	false
false && true	false
false && false	false

Expression	Value
true || true	true
true || false	true
false || true	true
false || false	false

Expression	Value
!true	false
!false	true

2.4. Variabel di Go

Sebuah variabel adalah lokasi penyimpanan dengan tipe tertentu dan dihubungkan dengan nama variabel. Ubah program pada bab 2 menggunakan variabel :

```
package main

import "fmt"

func main() {
    var x string = "Hello World"
    fmt.Println(x)
}
```

String literal dari program asli tampil di program tapi dikirim langsung ke fungsi Println yang ditetapkan pada sebuah variabel. Variabel Go dibuat dengan menggunakan keyword var kemudian ditentukan nama variabelnya (x), tipe data (string) dan akhirnya tetapkan sebuah nilai ke variabel (Hello World). Langkah akhir adalah opsional merupakan cara alternatif untuk menulis program seperti berikut :

```
package main

import "fmt"

func main() {
    var x string
    x = "Hello World"
    fmt.Println(x)
}
```

Variabel di Go mirip dengan variabel di aljabar tapi ada beberapa perbedaan :

1. Ketika melihat simbol =, kita mempunyai kecenderungan untuk membacanya sebagi “x sama dengan string Hello Workd”. Tidak ada yang salah dengan pembacaan seperti itu, tetapi akan lebih baik jika dibaca sebagai “x mengambil string Hello World” atau “x adalah string Hello World”. Perbedaan ini penting, karena variabel tidak dapat nilainya selama jalannya program. Coba untuk menjalankan program berikut :

```
package main

import "fmt"

func main() {
    var x string
    x = "first"
    fmt.Println(x)
    x = "second"
    fmt.Println(x)
}
```

Faktanya, program dapat ditulis dengan cara lain :

```
var x string
x = "first "
fmt.Println(x)
x = x + "second"
fmt.Println(x)
```

Program ini akan masuk akal jika kita membaca program sebagai sederetan perintah. Saat melihat x = x + ”second” kita akan membacanya sebagai “gabungkan nilai dari variabel x dan string literal second ke variabel x”. Disisi kanan dari =, dijalankan lebih dulu dan hasilnya dimasukkan ke sisi kiri dari tanda =. Bentuk x=x+y adalah umum pada program. Go mempunyai statement penugasan khusus +=. Kita telah menulis x = x + ”second” sebagai x += “second” dan keduanya adalah sama.

2. Perbedaan lain antara Go dan aljabar adalah penggunaan simbol yang berbeda untuk equality yaitu = =. Dua tanda sama dengan. = = adalah operaotr seperti + dan mengembalikan nilai boolean. Sebagai contoh :

```
var x string = "hello"
var y string = "world"
fmt.Println(x == y)
```

Program akan mencetak false karen hello tidak sama dengan world, disisi lain :

```
var x string = "hello"
var y string = "hello"
fmt.Println(x == y)
```

Program akan mencetak true karena dua string adalah sama. Pembuatan variabel baru dengan nilai awal adalah hal biasa pada Go, Go mendukung statement lebh ringkas untuk itu :

```
x := "Hello World"
```

: sebelum = dan tidak ada tipe yang ditentukan. Tipe data tidak dibutuhkan karena Go compiler dapat menyimpulkan tipe data berdasarkan pada nilai literal yang ditetapkan pada variabel. (x didefinisikan bertipe string karena berisi literal string). Compiler juga dapat menyimpulkan dengan statement var :

```
var x = "Hello World"
```

Hal yang sama bekerja untuk tipe data lain :

```
x := 5
fmt.Println(x)
```

Biasanya bentuk lebih pendek ini digunakan.

2.4.1. Bagaimana Memberi Nama Sebuah Variabel

Penamaan variabel adalah hal penting dari pengembangan perangkat lunak. Nama diawali dengan huruf dan dapat berisi huruf, bulangan atau simbol _ (underscore). Go compiler tidak peduli dengan nama variabel jadi berilah nama yang mempunyai makna bagi programmer (atau orang lain). Ambil nama yang secara tepat menjelaskan tujuan dari variabel.

```
    x := "Max"
    fmt.Println("My dog's name is", x)
```

Pada kasus diatas, x bukan merupakan variabel yang baik, akan lebih baik jika :

```
    name := "Max"
    fmt.Println("My dog's name is", name)
```

Atau bahkan :

```
    dogsName := "Max"
    fmt.Println("My dog's name is", dogsName)
```

Pada kasus ini, digunakan cara khusus untuk merepresentasikan multiple word dengan nama variabel yang dikenal sebagai lower camel (disebut juga mixed case, camel back atau hump back). Huruf pertama daRI kata pertama adalah huruf kecil, huruf pertama dari kata yang mengikutinya adalah huruf besar dan seluruh huruf lain adalah huruf kecil.

4.2. Scope

Kembali lihat program di awal bab :

```
package main

import "fmt"

func main() {
    var x string = "Hello World"
    fmt.Println(x)
}
```

Cara lain untuk menulis program ini adalah :

```
package main

import "fmt"

var x string = "Hello World"

func main() {
    fmt.Println(x)
}
```

Variabel dipindah keluar fungsi main, artinya bahwa fungsi lain dapat mengakses variabel tersebut :

```
var x string = "Hello World"

func main() {
    fmt.Println(x)
}

func f() {
    fmt.Println(x)
}
```

Sekarang fungsi f mengakses variabel x, sebagai gantinya ditulis program :

```
func main() {
    var x string = "Hello World"
    fmt.Println(x)
}

func f() {
    fmt.Println(x)
}
```

Jika dijalankan, akan tampak kesalahan :

```
.\main.go:11: undefined: x
```

Compiler memberitahukan bahwa variabel x didalam fungsi f tidak ada. Variabel ini hanya ada didalam fungsi main. Tempat yang memungkinkan untuk menggunakan variabel x disebut scope dari variabel. Pada dasarnya berarti variabel ada dalam tanda { } terdekat (sebuah blok) , tapi tidak diluar tanda { }.

2.4.3. Konstanta

Go juga mendukung konstanta. Konstanta pada dasarnya adalah variabel yang nilainya nantinya tidak dapat berubah. Konstanta dibuat dengan cara yang sama seperti membuat variabel.

```
package main

import "fmt"

func main() {
    const x string = "Hello World"
    fmt.Println(x)
}
```

```
    const x string = "Hello World"
    x = "Some other string"
```

Menghasilkan kesalahan pada saat kompilasi :

```
.\main.go:7: cannot assign to x
```

Konstanta adalah cara yang bagus untuk menggunakan kembali nilai di program tanpa menuliskannya setiap saat, contoh Pi pada package math didefinisikan sebagai konstanta. Konstata dapat dideklasikan dengan beberapa cara :

const a = 0

const b = 4.1

const c = "Yo"

2. 4.4. Mendefinisikan Multiple Variabel

Go juga mempunyai cara singkat untuk mendefinisikan multiple variabel

```
var (
    a = 5
    b = 10
    c = 15
)
```

Menggunakan keyword var (atau const) diikuti dengan () dengan setiap variabel di setiap barisnya.

Kurung Normal (dan) dapat digunakan untuk deklarasi dan dapat digunakan seperti contoh di bawah untuk membuat blok var dan deklarasi const.

```
var (
  int
  b float32
  c string = "Hello"
)
const (
  d = 9
  e = "Mission:"
)
fmt.Println ( a)
fmt.Println ( b )
fmt.Println ( c )
fmt.Println ( d )
fmt.Println ( e
)
```

Menghasilkan output :

```
0
0
Hello
9
Mission:
```

2.4.5. Contoh Program

Berikut adalah contoh program yang mengambil bilangan yang didapat dari masukan oleh user :

```
package main

import "fmt"

func main() {
     fmt.Print("Enter a number: ")
     var input float64
     fmt.Scanf("%f", &input)

     output := input * 2

     fmt.Println(output)
}
```

Disini digunakan fungsi lain dari package fmt untuk membaca input dari user (Scanf).