MATERI

# Hari 2 - Introduction Golang, Variable, Constant & Data Type

## Introduction Golang

**Golang** (atau biasa disebut dengan Go) adalah bahasa pemrograman baru yang dikembangkan di Google oleh Robert Griesemer, Rob Pike, dan Ken Thompson pada tahun 2007 dan mulai diperkenalkan ke publik tahun 2009.

Penciptaan bahasa Go didasari bahasa **C** dan **C++**, oleh karena itu gaya sintaksnya mirip.

### Kelebihan Go

Go memiliki kelebihan dibanding bahasa lainnya, beberapa di antaranya:

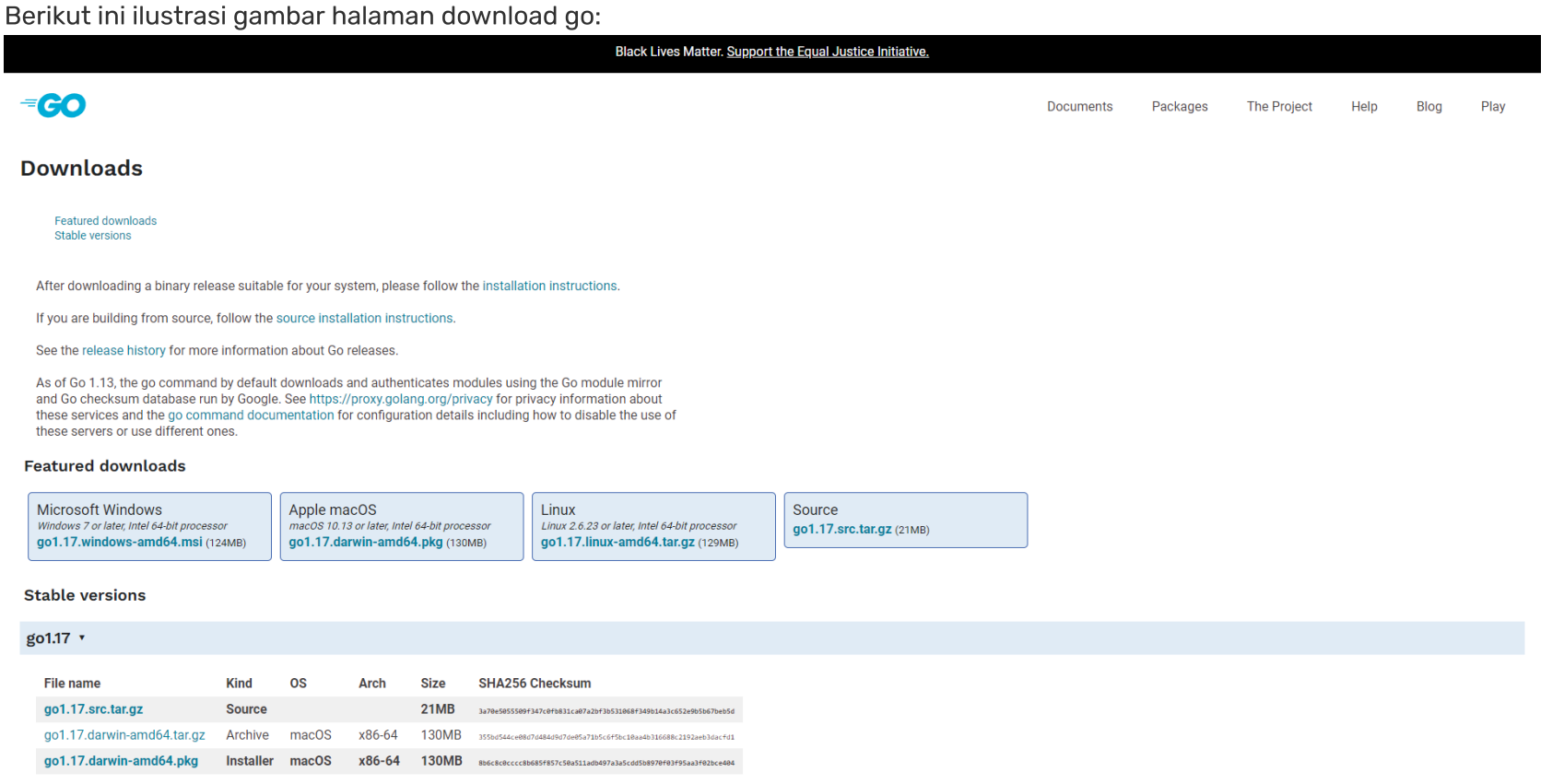
- Mendukung konkurensi di level bahasa dengan pengaplikasian cukup mudah
- Mendukung pemrosesan data dengan banyak prosesor dalam waktu yang bersamaan *(pararel processing)*
- Memiliki *garbage collector*
- Proses kompilasi sangat cepat
- Bukan bahasa pemrograman yang hirarkial, menjadikan developer tidak perlu *ribet* memikirkan segmen OOP-nya
- Package/modul yang disediakan terbilang lengkap. Karena bahasa ini open source, banyak sekali developer yang juga mengembangkan modul-modul lain yang bisa dimanfaatkan

## Instalasi Go

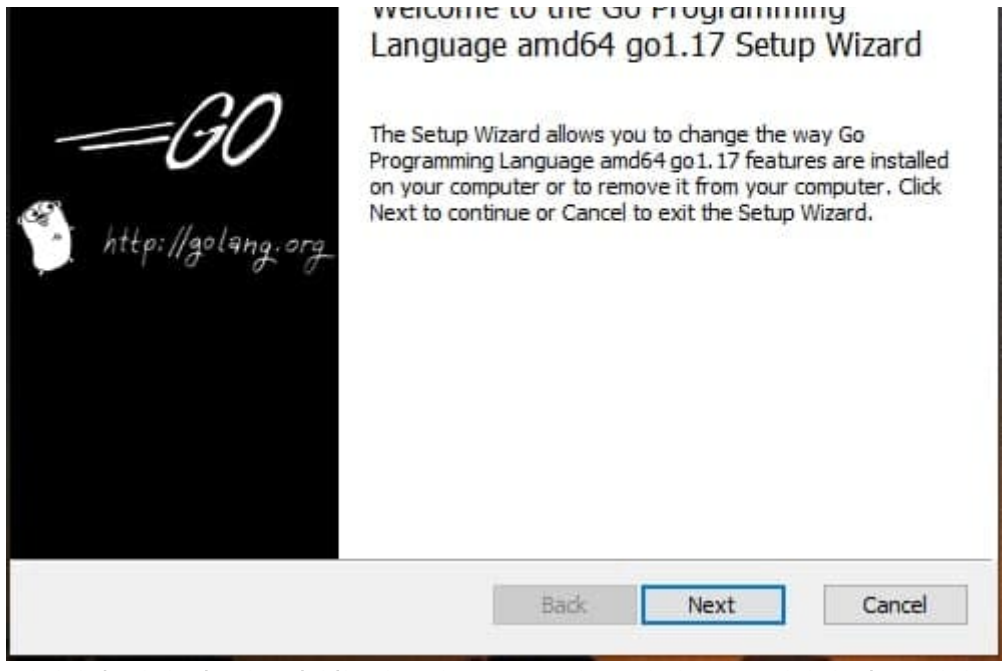
**a. Instalasi di Windows dan Mac**

Berikut ini cara untuk instalasi go ke dalam windows dan mac

1. Download terlebih dahulu file instalasi nya dari halaman https://golang.org/dl/
   Berikut ini ilustrasi gambar halaman download go:




2. Pilihlah sesuai requirement OS pada PC/Laptop anda misalnya Windows / macOS
3. Jika sudah terdownload, Jalankan file instalasi lalu ikuti instruksi instalasi,
   Berikut ini contohnya : ( **Klik next saja sampai selesai )**




4. setelah instalasi selesai, kita dapat mengecek apakah Go sudah terinstall dengan cara membuka cmd atau command prompt lalu ketikkan perintah `go version`.


## b. Instalasi di Linux

Berikut ini cara untuk instalasi Go ke dalam linux:

1. Unduh arsip *installer* dari https://golang.org/dl/, pilih installer untuk Linux yang sesuai dengan jenis bit komputer anda. Proses download bisa dilakukan lewat CLI, menggunakan `wget` atau `curl`.

```
$ wget https://golang.org/dl/go1.17.linux-amd64.tar.gz
```

2. Buka terminal, *extract* arsip tersebut ke `/usr/local`.

```
$ rm -rf /usr/local/go && tar -C /usr/local -xzf go1.17.linux-amd64.tar.gz
```

3. Tambahkan path binary Go ke `PATH` *environment variable*.

```
$ echo "export PATH=$PATH:/usr/local/go/bin" >> ~/.bashrc
$ source ~/.bashrc
```

4. Selanjutnya, eksekusi perintah berikut untuk mengetes apakah Go sudah terinstal dengan benar.

```
go version
```

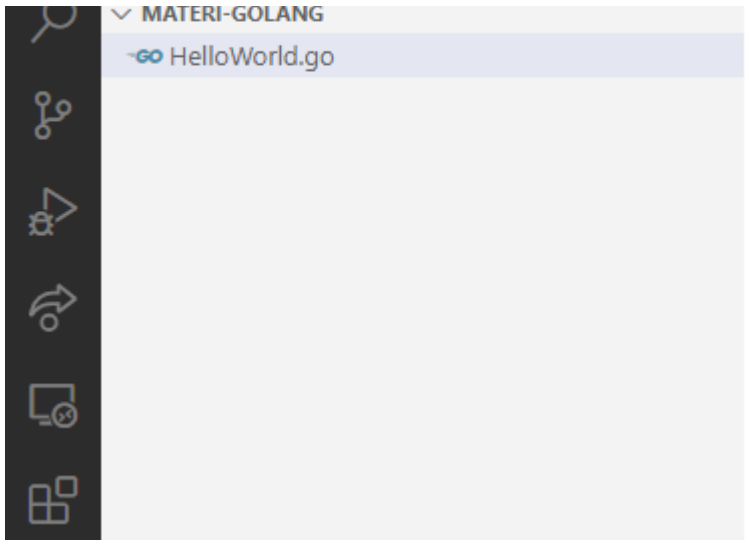
5. Jika output adalah sama dengan versi Go yang ter-*install*, menandakan proses instalasi berhasil.

---

# Membuat Program Go Pertama

---

Hal yang di lakukan pertama kali dalam membuat program go ialah membuat file berekstensi go, misal HelloWorld.go, anda bisa langsung membuka teks editor yang anda punya dan buat file berekstensi go tersebut:



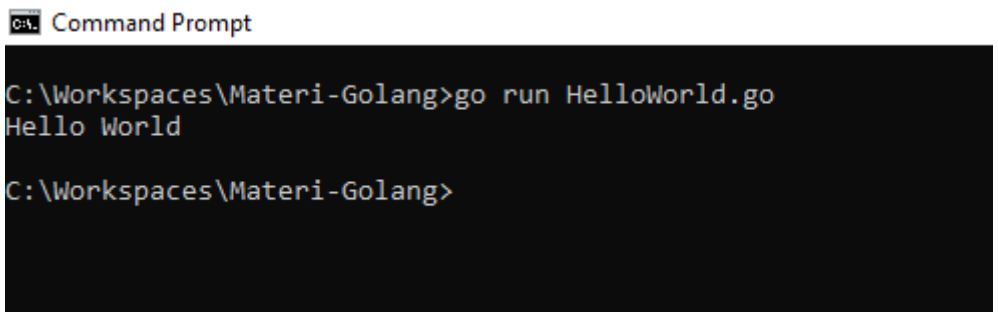

lalu buatlah kode seperti di bawah ini di dalam file HelloWorld.go tersebut:

```
package main

import (
        "fmt"
)

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

setelah itu buka terminal dimana lokasi file tersebut berada lalu jalankan perintah `go run HelloWorld.go` untuk menjalankan program go




---

# Go Modules

walaupun sebelumnya kita sudah bisa menjalankan program go pertama kita, itu contoh kurang ideal dalam sebuah project go, dalam project go setidaknya terdapat file go.mod yang diinisialisasi menggunakan go modules sintaks.

Go modules merupakan manajemen dependensi resmi untuk Go. Modules ini diperkenalkan pertama kali di `go1.11`, sebelum itu pengembangan project Go dilakukan dalam `GOPATH`. Modules digunakan untuk menginisialisasi sebuah project, sekaligus melakukan manajemen terhadap *3rd party* atau *library* lain yang dipergunakan. Modules penggunaannya adalah lewat CLI. Dan jika temen-temen sudah sukses meng-*install* Go, maka otomatis bisa mempergunakan Go Modules. Modules atau Module disini merupakan istilah untuk project ya. Jadi jangan bingung.

## Inisialisasi Project Menggunakan Go Modules

Command `go mod init` digunakan untuk menginisalisasi project baru.

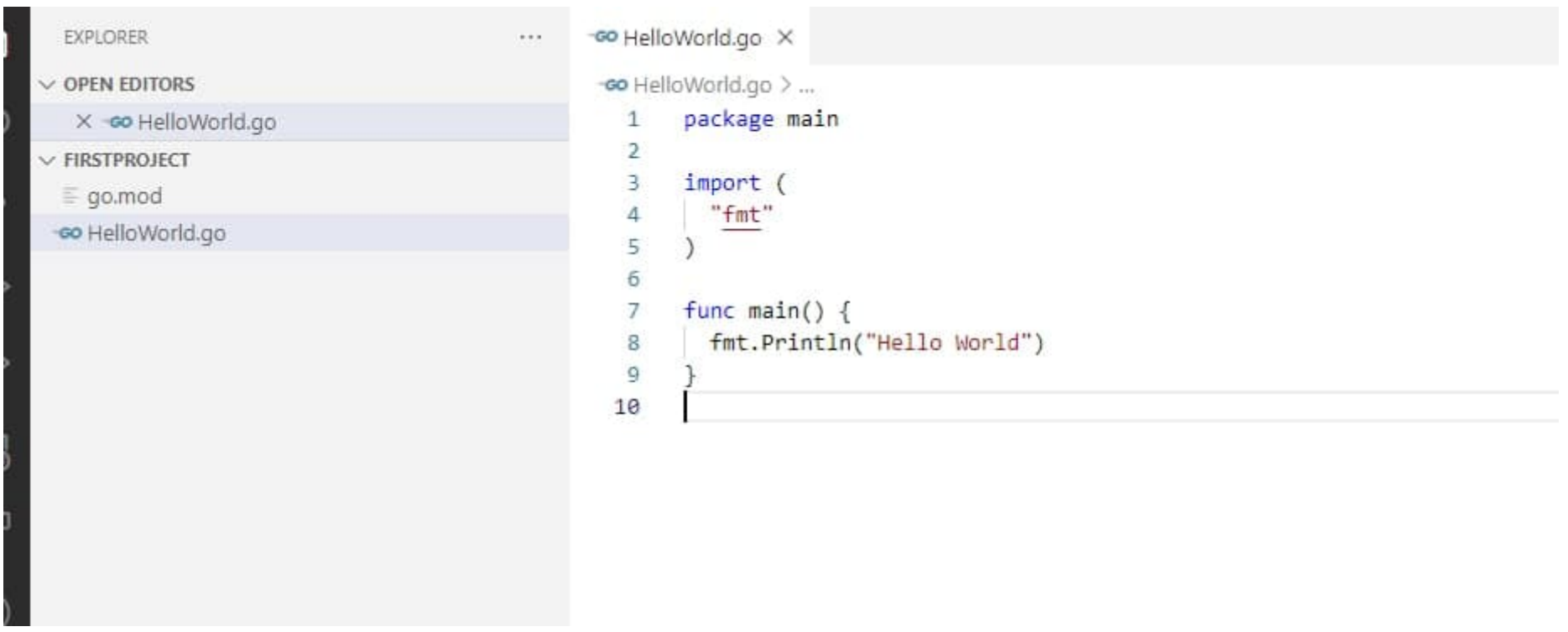
Skema penulisan command `go mod`:

```
go mod init <nama-project>
```

Mari kita praktekan disini saya akan menggunakan folder baru dengan nama firstProject di CLI dan menjalankan perintah `go mod init`.


setelah menjalankan perintah diatas maka akan dibuatkan sebuah file baru dengan nama go.mod

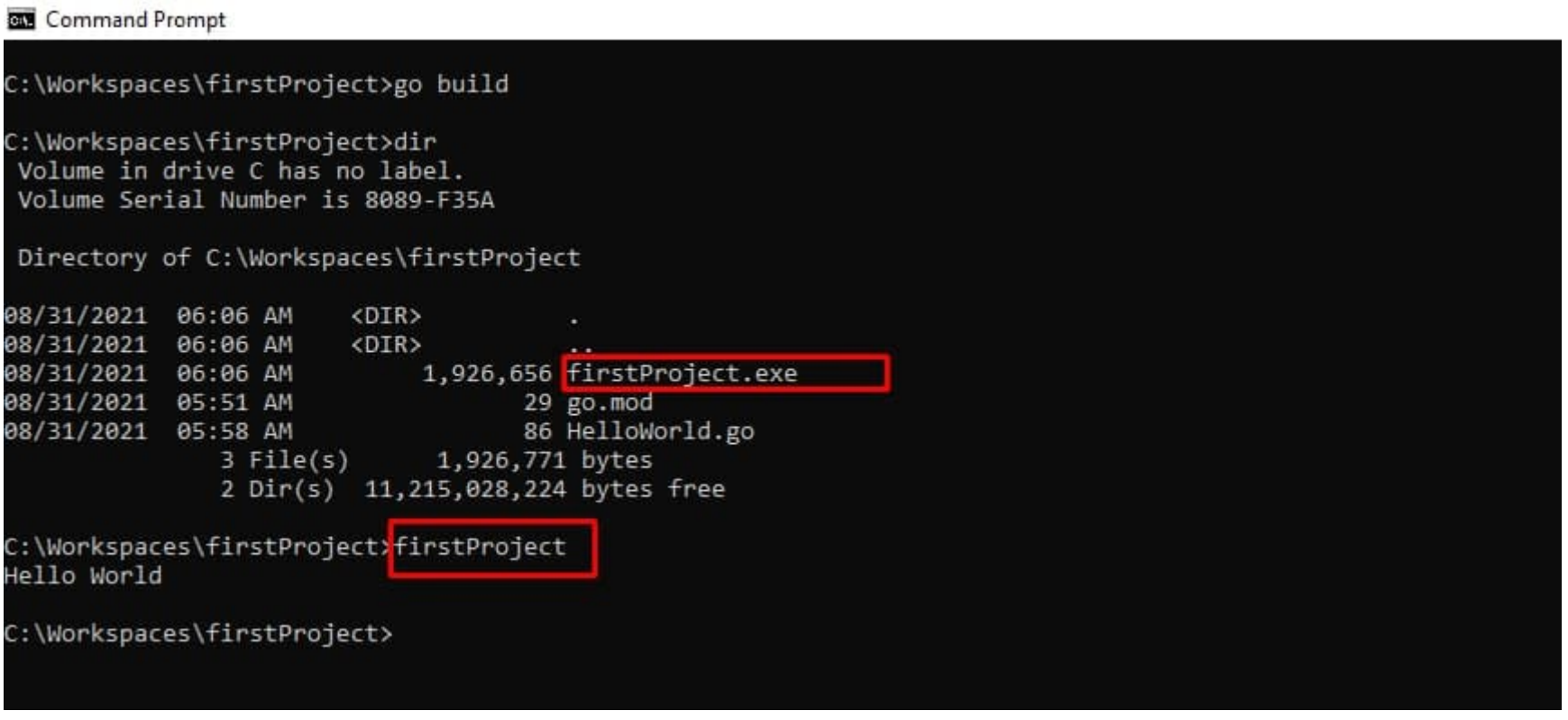

setelah itu kita coba buat file go lagi dan sintaksnya sama seperti sebelumnya yang menampilkan Hello World




untuk menjalankan programnya masih menggunakan sintaks `go run <nama-file>` seperti tadi, tetapi ada cara lainnya jika sudah menginisialisasi project dengan `go modules`, yaitu dengan menjalankan perintah `go build`.

`go build` adalah *Command* yang digunakan untuk mengkompilasi file program.Sebenarnya ketika eksekusi program menggunakan `go run`, terjadi proses kompilasi juga. File hasil kompilasi akan disimpan pada folder temporary untuk selanjutnya langsung dieksekusi.

Berbeda dengan `go build`, *command* ini menghasilkan file *executable* atau *binary* pada folder yang sedang aktif. Contohnya bisa dilihat di bawah ini:




pada contoh tersebut setelah menjalankan perintah go build, otomatis di buatkan file firstProject.exe (namanya mengikuti nama module di go.mod), dan ketika firstProject itu di akses langsung menjalankan programnya

---

# Variable dan Constant

---

## Variable

**Variable/variabel** adalah lokasi penyimpanan dan terkait nama simbolis yang berisi beberapa kuantitas yang diketahui atau tidak diketahui atau informasi, nilai. singkatnya variabel kita gunakan untuk menyimpan data. Di dalam go ada beberapa cara untuk mendeklarasikan sebuah variabel.

```
package main

import (
        "fmt"
)

func main() {
        var text = "Hello World"
        fmt.Println(text)

        text = "Hello World diubah"
        fmt.Println(text)
}
```

### 2. Deklarasi dengan var dan juga tipe data

```
package main

import (
        "fmt"
)

func main() {
        // contoh 1
        var text1 string
        text1 = "ini teks 1"
        fmt.Println(text1)

        // contoh 1
        var text2 string = "ini teks 2"
        text2 = "ini teks 1 diubah"
        fmt.Println(text2)
}
```

### 3. Deklarasi dengan menggunakan perantara ":="

```
package main

import (
        "fmt"
)

func main() {
        // contoh 1
        text1 := "ini teks 1"
        text1 = "ini teks 1 diubah"
        fmt.Println(text1)

        // contoh 1
        text2 := "ini teks 2"
        fmt.Println(text2)
}
```

## Constant

**Constant/konstanta** adalah jenis variabel yang nilainya tidak bisa diubah. Inisialisasi nilai hanya dilakukan sekali di awal, setelah itu variabel tidak bisa diubah nilainya.

berikut ini contoh penggunaan constant:

```
package main

import (
        "fmt"
)

func main() {
        const judul = "Ini Judul"
        fmt.Println(judul)
        // jika kode di bawah ini di uncomment maka akan error
        // judul = "judul di ubah"
}
```

# Tipe Data

Go mengenal beberapa jenis tipe data, diantaranya adalah tipe data numerik (desimal & non-desimal), string, dan boolean.

## Tipe Data Numerik Non-Desimal

Tipe data numerik non-desimal atau **non floating point** di Go ada beberapa jenis. Secara umum ada 2 tipe data kategori ini yang perlu diketahui.

- `uint`, tipe data untuk bilangan cacah (bilangan positif).
- `int`, tipe data untuk bilangan bulat (bilangan negatif dan positif).

Kedua tipe data di atas kemudian dibagi lagi menjadi beberapa jenis, dengan pembagian berdasarkan lebar cakupan nilainya, detailnya bisa dilihat di tabel berikut.

| Tipe data | Cakupan bilangan |
|---|---|
| `uint8` | 0 ↔ 255 |
| `uint16` | 0 ↔ 65535 |
| `uint32` | 0 ↔ 4294967295 |
| `uint64` | 0 ↔ 18446744073709551615 |
| `uint` | sama dengan `uint32` atau `uint64` (tergantung nilai) |
| `byte` | sama dengan `uint8` |
| `int8` | -128 ↔ 127 |
| `int16` | -32768 ↔ 32767 |
| `int32` | -2147483648 ↔ 2147483647 |
| `int64` | -9223372036854775808 ↔ 9223372036854775807 |
| `int` | sama dengan `int32` atau `int64` (tergantung nilai) |
| `rune` | sama dengan `int32` |

Dianjurkan untuk tidak sembarangan dalam menentukan tipe data variabel, sebisa mungkin tipe yang dipilih harus disesuaikan dengan nilainya, karena efeknya adalah ke alokasi memori variabel. Pemilihan tipe data yang tepat akan membuat pemakaian memori lebih optimal, tidak berlebihan.

```
var positiveNumber uint8 = 89
var negativeNumber = -1243423644

fmt.Printf("bilangan positif: %d\n", positiveNumber)
fmt.Printf("bilangan negatif: %d\n", negativeNumber)
```

Variabel `positiveNumber` bertipe `uint8` dengan nilai awal `89`. Sedangkan variabel `negativeNumber` dideklarasikan dengan nilai awal `-1243423644`. Compiler secara cerdas akan menentukan tipe data variabel tersebut sebagai `int32` (karena angka tersebut masuk ke cakupan tipe data `int32`).

Template `%d` pada `fmt.Printf()` digunakan untuk memformat data numerik non-desimal.

## Tipe Data Numerik Desimal

Tipe data numerik desimal yang perlu diketahui ada 2, `float32` dan `float64`. Perbedaan kedua tipe data tersebut berada di lebar cakupan nilai desimal yang bisa ditampung.

```
var decimalNumber = 2.62

fmt.Printf("bilangan desimal: %f\n", decimalNumber)
fmt.Printf("bilangan desimal: %.3f\n", decimalNumber)
```

Pada kode di atas, variabel `decimalNumber` akan memiliki tipe data `float32`, karena nilainya berada di cakupan tipe data tersebut.

```
bilangan desimal: 2.620000
bilangan desimal: 2.620
```

## Tipe Data `bool` (Boolean)

Tipe data `bool` berisikan hanya 2 variansi nilai, `true` dan `false`. Tipe data ini biasa dimanfaatkan dalam seleksi kondisi dan perulangan

```
var exist bool = true
fmt.Printf("exist? %t \n", exist)
```

Gunakan `%t` untuk memformat data `bool` menggunakan fungsi `fmt.Printf()`.

## Tipe Data `string`

Ciri khas dari tipe data string adalah nilainya di apit oleh tanda *quote* atau petik dua (`"`). Contoh penerapannya:

```
var message string = "Halo"
fmt.Printf("message: %s \n", message)
```

Selain menggunakan tanda quote, deklarasi string juga bisa dengan tanda *grave accent/backticks* (`` ` ``), tanda ini terletak di sebelah kiri tombol 1. Keistimewaan string yang dideklarasikan menggunakan backtics adalah membuat semua karakter didalamnya **tidak di escape**, termasuk `\n`, tanda petik dua dan tanda petik satu, baris baru, dan lainnya. Semua akan terdeteksi sebagai string.

```
var message = `Nama saya "John Wick".
Salam kenal.
Mari belajar "Golang".`

fmt.Println(message)
```

Ketika dijalankan, output akan muncul sama persisi sesuai nilai variabel `message` di atas. Tanda petik dua akan muncul, baris baru juga muncul, sama persis.

```
Nama saya "John Wick".
Salam kenal.
Mari belajar "Golang".
```

## Nilai `nil` & Zero Value

`nil` bukan merupakan tipe data, melainkan sebuah nilai. Variabel yang isi nilainya `nil` berarti memiliki nilai kosong.

Semua tipe data yang sudah dibahas di atas memiliki zero value (nilai default tipe data). Artinya meskipun variabel dideklarasikan dengan tanpa nilai awal, tetap akan ada nilai default-nya.

- Zero value dari `string` adalah `""` (string kosong).
- Zero value dari `bool` adalah `false`.
- Zero value dari tipe numerik non-desimal adalah `0`.
- Zero value dari tipe numerik desimal adalah `0.0`.

Zero value berbeda dengan `nil`. `Nil` adalah nilai kosong, benar-benar kosong. `nil` tidak bisa digunakan pada tipe data yang sudah dibahas di atas. Ada beberapa tipe data yang bisa di-set nilainya dengan `nil`, diantaranya:

- pointer
- tipe data fungsi
- slice
- `map`
- `channel`
- interface kosong atau `interface{}`

Nantinya kita akan sering bertemu dengan `nil` setelah masuk pada materi selanjutnya.

## Konversi Data Menggunakan Teknik Casting

Keyword tipe data bisa digunakan untuk casting, atau konversi antar tipe data. Cara penggunaannya adalah dengan menuliskan tipe data tujuan casting sebagai fungsi, lalu menyisipkan data yang akan dikonversi sebagai parameter fungsi tersebut.

```
var a float64 = float64(24)
fmt.Println(a) // 24

var b int32 = int32(24.00)
fmt.Println(b) // 24

var str = "Halo"
var c string = string(str[0])
fmt.Println(c) // H
```

# Package strings

Go menyediakan package `strings`, isinya banyak fungsi untuk keperluan pengolahan data string.

## 1. `strings.Index()`

Digunakan untuk mencari posisi indeks sebuah string (parameter kedua) dalam string (parameter pertama).

```
var index1 = strings.Index("ethan hunt", "ha")
fmt.Println(index1) // 2
```

String `"ha"` berada pada posisi ke `2` dalam string `"ethan hunt"` (indeks dimulai dari 0). Jika diketemukan dua substring, maka yang diambil adalah yang pertama, contoh:

```
var index2 = strings.Index("ethan hunt", "n")
fmt.Println(index2) // 4
```

String `"n"` berada pada indeks `4` dan `8`. Yang dikembalikan adalah yang paling kiri (paling kecil), yaitu `4`.

## 2. `strings.Replace()`

Fungsi ini digunakan untuk replace atau mengganti bagian dari string dengan string tertentu. Jumlah substring yang di-replace bisa ditentukan, apakah hanya 1 string pertama, 2 string, atau kesemuanya.

```
var text = "banana"
var find = "a"
var replaceWith = "o"

var newText1 = strings.Replace(text, find, replaceWith, 1)
fmt.Println(newText1) // "bonana"

var newText2 = strings.Replace(text, find, replaceWith, 2)
fmt.Println(newText2) // "bonona"

var newText3 = strings.Replace(text, find, replaceWith, -1)
fmt.Println(newText3) // "bonono"
```

Penjelasan:

1. Pada contoh di atas, substring `"a"` pada string `"banana"` akan di-replace dengan string `"o"`.
2. Pada `newText1`, hanya 1 huruf `o` saja yang tereplace karena maksimal substring yang ingin di-replace ditentukan 1.
3. Angka `-1` akan menjadikan proses replace berlaku pada semua substring. Contoh bisa dilihat pada `newText3`.

## 3.`strings.Repeat()`

Digunakan untuk mengulang string (parameter pertama) sebanyak data yang ditentukan (parameter kedua).

```
var str = strings.Repeat("na", 4)
fmt.Println(str) // "nananana"
```

Pada contoh di atas, string `"na"` diulang sebanyak 4 kali. Hasilnya adalah: `"nananana"`

## 4. `strings.ToLower()`

Mengubah huruf-huruf string menjadi huruf kecil.

```
var str = strings.ToLower("aLAy")
fmt.Println(str) // "alay"
```

## 5. `strings.ToUpper()`

Mengubah huruf-huruf string menjadi huruf besar.

```
var str = strings.ToUpper("eat!")
fmt.Println(str) // "EAT!"
```

jika ingin lebih banyak lagi fungsi-fungsi dari packages strings silahkan akses https://pkg.go.dev/strings

# Package strconv

Package `strconv` berisi banyak fungsi yang sangat membantu kita untuk melakukan konversi. Berikut merupakan beberapa fungsi yang dalam package tersebut.

• Fungsi `strconv.Atoi()`

```
package main

import "fmt"
import "strconv"

func main() {
    var str = "124"
    var num, err = strconv.Atoi(str)

    if err == nil {
        fmt.Println(num) // 124
    }
}
```

### • Fungsi `strconv.Itoa()`

Merupakan kebalikan dari `strconv.Atoi`, berguna untuk konversi `int` ke `string`.

```
var num = 124
var str = strconv.Itoa(num)

fmt.Println(str) // "124"
```

### • Fungsi `strconv.ParseInt()`

Digunakan untuk konversi `string` berbentuk numerik dengan basis tertentu ke tipe numerik non-desimal dengan lebar data bisa ditentukan.

Pada contoh berikut, string `"124"` dikonversi ke tipe numerik dengan ketentuan basis yang digunakan `10` dan lebar datanya mengikuti tipe `int64` (lihat parameter ketiga).

```
var str = "124"
var num= strconv.ParseInt(str, 10, 64)

fmt.Println(num) // 124
```

Contoh lainnya, string `"1010"` dikonversi ke basis 2 (biner) dengan tipe data hasil adalah `int8`.

```
var str = "1010"
var num= strconv.ParseInt(str, 2, 8)

fmt.Println(num) // 10
```

### • Fungsi `strconv.FormatInt()`

Berguna untuk konversi data numerik `int64` ke `string` dengan basis numerik bisa ditentukan sendiri.

```
var num = int64(24)
var str = strconv.FormatInt(num, 8)

fmt.Println(str) // 30
```

### • Fungsi `strconv.ParseFloat()`

Digunakan untuk konversi `string` ke numerik desimal dengan lebar data bisa ditentukan.

```
var str = "24.12"
var num, err = strconv.ParseFloat(str, 32)

fmt.Println(num) // 24.1200008392334
```

Pada contoh di atas, string `"24.12"` dikonversi ke float dengan lebar tipe data `float32`. Hasil konversi `strconv.ParseFloat` adalah sesuai dengan standar IEEE Standard for Floating-Point Arithmetic.

### • Fungsi `strconv.FormatFloat()`

Berguna untuk konversi data bertipe `float64` ke `string` dengan format eksponen, lebar digit desimal, dan lebar tipe data bisa ditentukan.

```
var num = float64(24.12)
var str = strconv.FormatFloat(num, 'f', 6, 64)

fmt.Println(str) // 24.120000
```

Pada kode di atas, Data `24.12` yang bertipe `float64` dikonversi ke string dengan format eksponen `f` atau tanpa eksponen, lebar digit desimal 6 digit, dan lebar tipe data `float64`.

Ada beberapa format eksponen yang bisa digunakan. Detailnya bisa dilihat di tabel berikut.

| Format Eksponen | Penjelasan |
|---|---|
| `b` | -ddddp±ddd, a, eksponen biner (basis 2) |
| `e` | -d.dddde±dd, a, eksponen desimal (basis 10) |

| | |
|---|---|
| f | -ddd.dddd, tanpa eksponen |
| g | Akan menggunakan format eksponen e untuk eksponen besar dan f untuk selainnya |
| G | Akan menggunakan format eksponen E untuk eksponen besar dan f untuk selainnya |

• Fungsi `strconv.ParseBool()`

Digunakan untuk konversi `string` ke `bool`.

```
var str = "true"
var bul, err = strconv.ParseBool(str)

fmt.Println(bul) // true
```

• Fungsi `strconv.FormatBool()`

Digunakan untuk konversi `bool` ke `string`.

```
var bul = true
var str = strconv.FormatBool(bul)

fmt.Println(str) // true
```

jika ingin lebih banyak lagi fungsi-fungsi dari packages strings silahkan akses https://pkg.go.dev/strconv

---

## Referensi Video:

- https://www.youtube.com/watch?v=nyGu8Xn5b3g (Programmer Zaman Now)
- https://www.youtube.com/watch?v=sDHZbPfrNoM (Programmer Zaman Now)
- https://www.youtube.com/watch?v=eZO2DMQepns (Programmer Zaman Now)
- https://www.youtube.com/watch?v=RgufJoZ4idA (Programmer Zaman Now)
- https://www.youtube.com/watch?v=xz-CBXIjfd8 (Programmer Zaman Now)
- https://www.youtube.com/watch?v=vbTDLECgpZU (Programmer Zaman Now)
- https://www.youtube.com/watch?v=AKrz7Gh3hw4 (Programmer Zaman Now)
- https://www.youtube.com/watch?v=03f3rCDxhH0 (Programmer Zaman Now)
- https://www.youtube.com/watch?v=a3ShRKaGcvw (Programmer Zaman Now)
- https://www.youtube.com/watch?v=pwIiUJL0mpI (Programmer Zaman Now)

## Referensi Tulisan:

- https://dasarpemrogramangolang.novalagung.com/1-berkenalan-dengan-golang.html
- https://dasarpemrogramangolang.novalagung.com/2-instalasi-golang.html
- https://dasarpemrogramangolang.novalagung.com/A-setup-go-project-dengan-go-modules.html
- https://dasarpemrogramangolang.novalagung.com/A-go-command.html
- https://dasarpemrogramangolang.novalagung.com/A-hello-world.html
- https://dasarpemrogramangolang.novalagung.com/A-variabel.html
- https://dasarpemrogramangolang.novalagung.com/A-tipe-data.html
- https://dasarpemrogramangolang.novalagung.com/A-konstanta.html
- https://dasarpemrogramangolang.novalagung.com/A-data-type-conversion.html
- https://dasarpemrogramangolang.novalagung.com/A-strings.html
- https://pkg.go.dev/strings
- https://pkg.go.dev/strconv