Constants & Operators

Constant

Constant (const) atau Konstanta merupakan jenis variable pada bahasa Go yang nilainya tidak dapat diubah. Contohnya jika kita memiliki nilai-nilai tetap seperti *PI,* kecepatan cahaya, luas lingkaran dan lain-lain yang merupakan nilai-niali tetap.

Maka kita bisa menyimpan nilai-nilai tersebut dengan variable *const*. Seperti yang kita lihat pada gambar pertama dibawah, kita mempunyai suatu variable *const* yang memiliki tipe data *string*. Perlu diingat disini bahwa ketika kita membuat variable dengan *const,* maka kita harus langsung memberikan nilai kepadanya. Karena jika tidak maka akan timbul error pada saat compile time seperti pada gambar kedua dibawah.

```go
package main

import "fmt"

func main() {
  const full_name string = "Airell Jordan"

  fmt.Printf("Hello %s", full_name)
}
```

```go
package main

import "fmt"

func main() {
  const full_name string

  fmt.Println(full_name)
}
```

Constant

Constant (const) atau Konstanta merupakan jenis variable pada bahasa Go yang nilainya tidak dapat diubah. Contohnya jika kita memiliki nilai-nilai tetap seperti *PI,* kecepatan cahaya, luas lingkaran dan lain-lain yang merupakan nilai-niali tetap.

Maka kita bisa menyimpan nilai-nilai tersebut dengan variable *const*. Seperti yang kita lihat pada gambar pertama dibawah, kita mempunyai suatu variable *const* yang memiliki tipe data *string*. Perlu diingat disini bahwa ketika kita membuat variable dengan *const,* maka kita harus langsung memberikan nilai kepadanya. Karena jika tidak maka akan timbul error pada saat compile time seperti pada gambar kedua dibawah.

```go
package main

import "fmt"

func main() {
  const full_name string = "Airell Jordan"

  fmt.Printf("Hello %s", full_name)
}
```

```go
package main

import "fmt"

func main() {
  const full_name string

  fmt.Println(full_name)
}
```

Operators

Pada bahasa Go, terdapat 3 jenis operator yang perlu kita ketahui yaitu operator aritmatika, operator logika dan operator perbandingan. Mari kita bahasa dari operator aritmatika terlebih dahulu

- Operator aritmatika

  Pada gambar dibawah ini merupakan daftar dari operator aritmatika yang dapat kita pakai

Operator	Description	Example
+	Adds two operands	A + B gives 30
-	Subtracts second operand from the first	A - B gives -10
*	Multiplies both operands	A * B gives 200
/	Divides the numerator by the denominator.	B / A gives 2
%	Modulus operator; gives the remainder after an integer division.	B % A gives 0
++	Increment operator. It increases the integer value by one.	A++ gives 11
--	Decrement operator. It decreases the integer value by one.	A-- gives 9

https://www.tutorialspoint.com/go/go_operators.htm

Operators (Arithmetic Operators)

Jika kita melihat pada gambar pertama disebelah kanan, kita telah membuat suatu variable bernama *value* yang dimana variable *value* menyimpan nilai dari hasil kalkulasi 2 + 2 * 3. Walaupun pada bahasa Go, syntax kita akan dibaca dari kiri ke kanan, namun karena pada kalkulasi tersebut ada simbol * yang dimana simbol tersebut merupakan simbol perkalian, maka Go akan melakukan perkalian tersebut terlebih dahulu kemudian baru melakukan pertambahannya. Kalkulasi pada gambar pertama akan menghasilkan angka 8. Lalu jika kita ingin agar pertambahannya dieksekusi terlebih dahulu, maka kita bisa mengelompokkan kalkulasi yang ini kita eksekusi terlebih dahulu menggunakan simbol () seperti pada gambar kedua. Jika kita jalankan gambar kedua pada terminal kita, maka akan menghasilkan angka 12.

```go
package main

import "fmt"

func main() {
  var value = 2 + 2*3
  fmt.Println(value)
}
```

```go
package main

import "fmt"

func main() {
  var value = (2 + 2) * 3
  fmt.Println(value)
}
```

Operators (Relational Operators)

- Operator perbandingan/relasional

Operator relasional atau perbandingan dapat kita pakai untuk memeriksa kebenaran suatu kondisi yang pada akhirnya akan menghasilkan nilai dengan tipe data *bool*. Gambar dibawah ini merupakan daftar dari operator relasional/perbandingan yang dapat kita pakai.

Operator	Description	Example
==	It checks if the values of two operands are equal or not; if yes, the condition becomes true.	(A == B) is not true.
!=	It checks if the values of two operands are equal or not; if the values are not equal, then the condition becomes true.	(A != B) is true.
>	It checks if the value of left operand is greater than the value of right operand; if yes, the condition becomes true.	(A > B) is not true.
<	It checks if the value of left operand is less than the value of the right operand; if yes, the condition becomes true.	(A < B) is true.
>=	It checks if the value of the left operand is greater than or equal to the value of the right operand; if yes, the condition becomes true.	(A >= B) is not true.
<=	It checks if the value of left operand is less than or equal to the value of right operand; if yes, the condition becomes true.	(A <= B) is true.

https://www.tutorialspoint.com/go/go_operators.htm

Operators (Relational Operators)

Jika kita perhatikan pada gambar disebelah kanan, kita telah membuat 4 variable yang dimana semuanya akan menghasilkan tipe data *bool* yang dihasilkan dari pengecekan kesamaan nilai menggunakan operator perbandingan.

Jika kita jalankan pada terminal kita, maka kita akan melihat hasilnya seperti pada gambar kedua. Variable bernama *firstCondition* akan menghasilkan nilai *true* karena memang angka 2 lebih kecil dari angka 3.

```go
package main

import "fmt"

func main() {
  var firstCondition bool = 2 < 3
  var secondConditiion bool = "joey" == "Joey"
  var thirdContition bool = 10 != 2.3
  var fourthCondition bool = 11 <= 11

  fmt.Println("first condition:", firstCondition)
  fmt.Println("second condition:", secondConditiion)
  fmt.Println("third condition:", thirdContition)
  fmt.Println("fourth condition:", fourthCondition)
}
```

```
first condition: true
second condition: false
third condition: true
fourth condition: true
```

Operators (Relational Operators)

Variable bernama *secondCondition* akan menghasilkan nilai *false* karena kata “joey” tidak sama dengan kata “Joey”.

Variable bernama *thirdCondition* akan menghasilkan nilai *true* karena angka 10 tidak sama dengan angka 2.3.

Dan yang terakhir variable bernama *fourthCondition* akan *meng*hasilkan nilai *true* karena angka 11 kurang dari atau sama dengan angka 11.

```go
package main

import "fmt"

func main() {
  var firstCondition bool = 2 < 3
  var secondConditiion bool = "joey" == "Joey"
  var thirdContition bool = 10 != 2.3
  var fourthCondition bool = 11 <= 11

  fmt.Println("first condition:", firstCondition)
  fmt.Println("second condition:", secondConditiion)
  fmt.Println("third condition:", thirdContition)
  fmt.Println("fourth condition:", fourthCondition)
}
```

```
first condition: true
second condition: false
third condition: true
fourth condition: true
```

Operators (Logical Operators)

- Operator logika

Operator logika ini digunakan untuk mengetahui benar atau tidaknya suatu kombinasi nilai dengan bertipe data *boo*. Gambar dibawah ini merupakan daftar dari operator logika yang dapat kita pakai.

Operator	Description	Example
&&	Called Logical AND operator. If both the operands are non-zero, then condition becomes true.	(A && B) is false.
||	Called Logical OR Operator. If any of the two operands is non-zero, then condition becomes true.	(A || B) is true.
!	Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.	!(A && B) is true.

https://www.tutorialspoint.com/go/go_operators.htm

Operators (Logical Operators)

Seperti yang kita lihat pada gambar disebelah kanan, terdapat sebuah variable bernama *right* dengan nilai *true* lalu terdapat sebuah variable bernama *wrong* dengan nilai false.

Lalu terdapat variable dengan nama *wrongAndRight* yang dimana kita melakukan pengecekan kombinasi tipe data bool antart variable *wrong* dengan *right*. Variable *wrongAndRight* akan menghasilkan nilai *false* karena jika kita memakai operator &&, maka kedua nilainya harus sama dengan *true*.

```go
import "fmt"

func main() {
  var right = true
  var wrong = false

  var wrongAndRight = wrong && right
  fmt.Printf("wrong && right \t(%t) \n", wrongAndRight)

  var wrongOrRight = wrong || right
  fmt.Printf("wrong || right \t(%t) \n", wrongOrRight)

  var wrongReverse = !wrong
  fmt.Printf("!wrong \t\t(%t) \n", wrongReverse)
}
```

```
left && right    (false)
left || right    (true)
!left            (true)
```

Operators (Logical Operators)

Kemudian terdapat sebuah variable dengan nama *wrongOrRight* yang menghasilkan nilai *true* karena jika kita memakai operator || dan ingin menghasilkan nilai *true,* maka salah satu dari nilainya boleh *false*.

Dan yang terakhir terdapat sebuah variable bernama *wrongReverse* yang menghasilkan nilai *true,* karena memang jika menggunakan operator dengan simbol !, maka operator tersebut akan membalikan nilai dari *bool* nya, misalkan nilainya *true* maka akan menjadi *false* dan begitu juga sebaliknya.

```go
import "fmt"

func main() {
  var right = true
  var wrong = false

  var wrongAndRight = wrong && right
  fmt.Printf("wrong && right \t(%t) \n", wrongAndRight)

  var wrongOrRight = wrong || right
  fmt.Printf("wrong || right \t(%t) \n", wrongOrRight)

  var wrongReverse = !wrong
  fmt.Printf("!wrong \t\t(%t) \n", wrongReverse)
}
```

```
left && right   (false)
left || right   (true)
!left           (true)
```