



Go

3-Structure de controle

J. Vlasak

But du cours

- Savoir manipuler les structures de controle



Structure conditionnelle

- Ressemble qu code C, mais pas de parenthèse

```
n:= rand.Intn(10)

if n == 0 {
    fmt.Println("valeur la plus basse")
} else {
    fmt.Println("autre valeur")
}
```

```
n:= rand.Intn(10)

if n == 0 {
    fmt.Println("valeur la plus basse")
}
```

```
n := rand.Intn(10)

if n == 0 {
    fmt.Println("valeur la plus basse")
} else if n > 5 {
    fmt.Println("valeur de 6 à 9")
} else {
    fmt.Println("valeur de 1 à 4")
}
```

Structure conditionnelle

- Possible d'écrire

```
if n := rand.Intn(10); n == 0 {  
    fmt.Println("valeur la plus basse", n)  
} else if n > 5 {  
    fmt.Println("valeur de 6 à 9", n)  
} else {  
    fmt.Println("valeur de 1 à 4", n)  
}
```

- ATTENTION : n n'est visible que dans les blocs if.
- Très utile lors du traitement des erreurs



Structure conditionnelle

- Possible d'écrire

```
if n := rand.Intn(10); n == 0 {  
    fmt.Println("valeur la plus basse", n)  
} else if n > 5 {  
    fmt.Println("valeur de 6 à 9", n)  
} else {  
    fmt.Println("valeur de 1 à 4", n)  
}
```

- ATTENTION : n n'est visible que dans les blocs if.
- Très utile lors du traitement des erreurs



Switch

- Possibilité de remplacer par un switch

```
n := rand.Intn(10)

switch n {
case 0:
    fmt.Println("valeur la plus basse")
case 1, 2, 3, 4:
    fmt.Println(" valeur de 1 a 4")
case 5, 6, 7, 8, 9:
    fmt.Println("valeur de 6 à 9")
}
```

```
switch n := rand.Intn(10); n {
case 0:
    fmt.Println("valeur la plus basse")
case 1, 2, 3, 4:
    fmt.Println(" valeur de 1 a 4")
case 5, 6, 7, 8, 9:
    fmt.Println("valeur de 6 à 9")
}
```

Go 3-Structure de controle

7

Boucle, 4 possibilités

■ Complète, style -C

Condition d'arrêt

Déclaration/initialisation

```
for i := 0; i < 10; i++ {  
    fmt.Println(i)  
}
```

Dernière instruction de la boucle

■ ATTENTION : i n'est visible que dans les blocs if.



Boucle, 4 possibilités

■ Condition d'arrêt

```
i := 0

for i < 10 {
    fmt.Println(i)
    i++
}
```

■ ATTENTION : i est visible dans le bloc père de if



Boucle, 4 possibilités

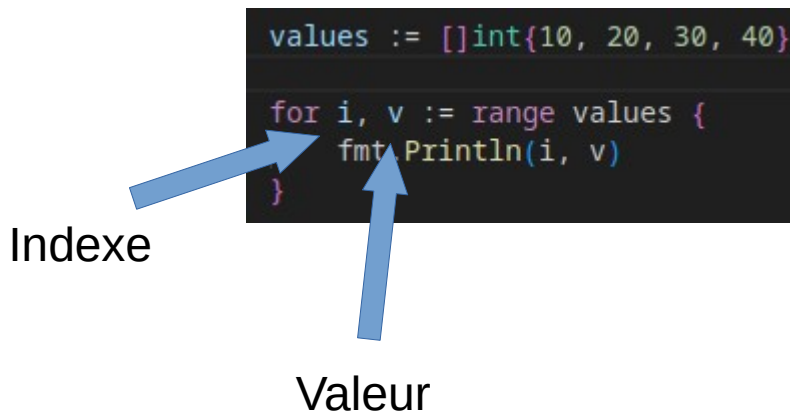
- Boucle infini
- Utilise pour attendre un événement, une requête http

```
for {  
    |   fmt.Println("infini")  
}
```



Boucle, 4 possibilités

■ For ... range



The diagram illustrates the 'For ... range' loop in Go. It features a code block with the following text:

```
values := []int{10, 20, 30, 40}

for i, v := range values {
    fmt.Println(i, v)
}
```

Two blue arrows point from labels to the code. The label 'Indexe' (Index) has an arrow pointing to the variable 'i' in the loop header. The label 'Valeur' (Value) has an arrow pointing to the variable 'v' in the loop header.

Switch

- Possibilité de remplacer par un switch

```
n := rand.Intn(10)

switch n {
case 0:
    fmt.Println("valeur la plus basse")
case 1, 2, 3, 4:
    fmt.Println(" valeur de 1 a 4")
case 5, 6, 7, 8, 9:
    fmt.Println("valeur de 6 à 9")
}
```

```
switch n := rand.Intn(10); n {
case 0:
    fmt.Println("valeur la plus basse")
case 1, 2, 3, 4:
    fmt.Println(" valeur de 1 a 4")
case 5, 6, 7, 8, 9:
    fmt.Println("valeur de 6 à 9")
}
```

Go 3-Structure de controle

