LES CONDITIONS DANS LE LANGAGE DE PROGRAMMATION GO

Les prérequis

Maintenant que vous savez comment déclarer vos variables, il est temps de les tester avec des conditions !

Entrée utilisateur

Avant de vous montrer comment créer des conditions, je vais d'abord vous indiquer comment enregistrer une **entrée utilisateur** dans une variable (nous utiliserons les entrées utilisateur plus tard dans la suite de ce chapitre et dans les prochains chapitres)

Pour cela, on va utiliser la fonction `NewScanner()` de la bibliothèque `bufio`. Cette fonction permet de créer un nouveau scanner qui nous permettra de capturer l'entrée utilisateur.

Voici comment elle s'utilise :

```
package main

import (
    "bufio"
    "fmt"
    "os"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin) // création du scanner capturant une entrée
    fmt.Print("Entrez quelque chose : ")
    scanner.Scan()                        // lancement du scanner
    entreeUtilisateur := scanner.Text() // stockage du résultat du scanner dans une v
    fmt.Println(entreeUtilisateur)
```

Appartient à devopssec.fr - lien original ici

```
}
```

Résultat :

```
Entrez quelque chose : quelque chose
quelque chose
```

Convertir une string en entier

J'ai une autre chose à vous montrer avant de passer à la suite. Pour le moment nous sommes capables de capturer que des chaines de caractères, dans certains cas vous aurez besoin de capturer un type int à la place d'un type string pour faire par exemple des calculs.

Pour ce faire, il suffit de convertir le résultat de votre scanner en int, une fonction nommée `Atoi()` de la bibliothèque `strconv` est disponible pour répondre à ce besoin.

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Print("Entrez un nombre entier : ")
    scanner.Scan()

    nbr, _ := strconv.Atoi(scanner.Text()) // conversion du type string en int

    fmt.Printf("res : %d\n", (nbr + 6))
}
```

Résultat :

```
Entrez un nombre entier : 6
res : 12
```

Les conditions

Maintenant promis on s'attaque aux conditions .

if, else

Imaginons le scénario suivant : *"un videur de boîte de nuit fainéant décide d'automatiser l'entrée des clients avec le langage de programmation Go."*

Pour répondre à l'envie du videur, on va utiliser les conditions.

Pour créer une condition il suffit d'utiliser le mot-clé `if` (qui se traduit en français par "si") suivit de votre condition. Ensuite vous ouvrez une accolade `{` et fermez-la un peu plus loin `}`. Tout ce qui se trouve à l'intérieur de vos accolades sera exécuté uniquement si la condition est bonne.

Pour revenir à notre scénario, on va d'abords commencer par respecter la loi en refusant les mineurs.

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Print("Entrez votre age : ")
    scanner.Scan()
    age, err := strconv.Atoi(scanner.Text())
    if err != nil {
```

```
        // Si la conversion n'a pas fonctionné alors on affiche un message d'erreur e
        fmt.Println("On essaie de m'arnaquer ? allé Dehors ! Et la prochaine entrez u
        os.Exit(2) // on quitte le programmation
    }

    if age < 17 { // vérifier si l'utilisateur à au moins 18 ans
        fmt.Println("Sortez !")
    } else { // si ce n'est pas le cas alors on l'accepte pas
        fmt.Println("Entrez :)")
    }
}
```

Résultat :

```
Entrez votre age : 16
Sortez !
```

```
Entrez votre age : 25
Entrez :)
```

> **Information**
>
> `nil` est une valeur par défaut de plusieurs autres types de variables que nous découvrirons dans d'autres chapitres

La dernière partie avec le mot-clé `else` (qui se traduit en français par "autre") traite le cas où aucune des conditions n'a été remplie (elle est optionnelle).

Les opérateurs de comparaison

Dans notre ancien exemple nous avons utilisé l'opérateur de comparaison `<`. En effet il existe d'autres types d'opérateurs que je vais décrire à travers le tableau ci-dessous

Opérateur	Description	Exemple	Résultat

`==`	Compare deux valeurs et vérifie leur égalité	x==4	La condition est bonne si x est égal à 4
`<`	Vérifie qu'une variable est strictement inférieure à une valeur	x<4	La condition est bonne si x est strictement inférieure à 4
`<=`	Vérifie qu'une variable est inférieure ou égale à une valeur	x<=4	La condition est bonne si x est inférieure ou égale à 4
`>`	Vérifie qu'une variable est strictement supérieure à une valeur	x>4	La condition est bonne si x est strictement supérieure à 4
`>=`	Vérifie qu'une variable est supérieure ou égale à une valeur	>=4	La condition est bonne si x est supérieure ou égale à 4
`!=`	Vérifie qu'une variable est différente à une valeur	>=4	La condition est bonne si x est différent à 4

Attention

Ne confondez pas l'opérateur d'égalité `==` avec l'opérateur d'affectation `=`, le premier permet de comparer l'égalité de deux variables et le second d'affecter une valeur à une variable !

Les opérateurs logiques

Les opérateurs logiques vont nous permettre de vérifier si plusieurs conditions sont bonnes. Il existe trois types d'opérateurs logiques :

Opérateur	Signification	Description
`||`	OU	Vrai si au moins une des comparaisons est vraie
`&&`	ET	Vrai si toutes les comparaisons sont vraies
`!`	NON	Retourne faux si la comparaison est vraie et vraie si la comparaison est fausse

Appartient à devopssec.fr - lien original ici

else if

Suite du scénario : *"Le videur a laissé sa place à un nouveau videur très fainéant aussi (on ne change pas une équipe qui gagne ?) mais bon il est un peu spécial ..."*

Tant mieux car au moins avec lui on va pouvoir utiliser les opérateurs logiques vu précédemment !

Voici les caractéristiques du nouveau videur :

- Le nouveau videur n'aime pas le prénom Hatim ou hatim (je ne me sens pas visé)
- Le nouveau videur a tendance à changer d'humeur de temps en temps et quand il n'est pas content il décide alors de virer les personnes ayant 18 ans

Les opérateurs de comparaison ne vont pas nous suffire, il nous faut un moyen pour tester d'autres conditions quand la première condition n'est pas vrai. Ce cas est gérable avec le mot-clé `else if` (qui se traduit en français par "sinon si"). La syntaxe ressemble à celle du mot-clé `if`.

Dans cet exemple je vais utiliser la bibliothèque `math/rand` pour apporter de l'aléatoire à notre code

```
package main

import (
    "bufio"
    "fmt"
    "math/rand"
    "os"
    "strconv"
    "time"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)
```

Appartient à devopssec.fr - lien original ici

```
    fmt.Print("Entrez votre age : ")
    scanner.Scan()
    age, err := strconv.Atoi(scanner.Text())
    if err != nil {
        fmt.Println("On essaie de m'arnaquer ? allé Dehors ! Et la prochaine entrez u
        os.Exit(2)
    }

    fmt.Print("Entrez votre prenom : ")
    scanner.Scan()
    prenom := scanner.Text()

    /*
        On a besoin de changer la graine (générateur de nombres pseudo-aléatoires) à
        chaque exécution de programmation sinon on obtiendra le même nombre aléatoire
    */
    rand.Seed(time.Now().UnixNano())
    radomInt := rand.Intn(2) // retourne aléatoirement soit 1 soit 0
    radomInt2 := rand.Intn(2)

    if age < 18 {
        fmt.Println("Sortez !")
    } else if prenom == "Hatim" || prenom == "hatim" {
        fmt.Println("Ah c'est toi Hatim, dehors !")
    } else if age == 18 && radomInt == 1 { // si le client est chanceux et qu'il a 18
        fmt.Println("Hum vous avez de la chance je suis de bonne humeur, entrez !")
    } else if radomInt2 == 0 {
        fmt.Println("Vous n'avez vraiment pas de chance sortez !")
    } else {
        fmt.Println("Entrez :)")
    }

}
```

Résultat :

```
Entrez votre age : 16
Sortez !
```

```
Entrez votre age : 19
Entrez votre prenom : Hatim
Ah c'est toi Hatim, dehors !
```

```
Entrez votre age : 18
Entrez votre prenom : Fred
Hum vous avez de la chance je suis de bonne humeur, entrez !
```

Appartient à devopssec.fr - *lien original ici*

```
Entrez votre age : 18
Entrez votre prenom : Alex
Vous n'avez vraiment pas de chance sortez !
```

switch, case

Une instruction `switch` permet de tester l'égalité d'une variable par rapport à une liste de valeurs (idéal pour le choix dans un menu !).

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Print("Votre choix : ")
    scanner.Scan()
    choix, err := strconv.Atoi(scanner.Text())
    if err != nil {
        fmt.Println("Entrez un entier !")
        os.Exit(2)
    }

    switch choix { // la variable qu'on souhaite vérifier
        case 0, 1: // 1 ou 0
            println("George Boole !")
        case 7:
            println("William Van de Walle!")
        case 23:
            println("Pour certains, le nombre 23 est source de nombreux mystères, qu'
        case 42:
            println("la réponse à la question ultime du sens de la vie!")
        case 666:
            println("Quand le diable veut une âme, le mal devient séduisant.")
        default:
            println("Mauvais choix !") // Valeur par défaut
    }
}
```

le mot-clé `default` correspond au choix par défaut comme le mot-clé `else`.

Résultat :

```
Votre choix : 666
Quand le diable veut une âme, le mal devient séduisant
```

```
Votre choix : 7
William Van de Walle!
```

```
Votre choix : 1
George Boole !
```

```
Votre choix : 0
George Boole !
```

```
Votre choix : 50
Mauvais choix !
```

Voici un autre exemple qui nous permet de savoir si on est en week-end ou pas, dans cet exemple nous utiliserons les fonctions `Now().Weekday()` de la bibliothèque `time` :

```
package main

import (
    "fmt"
    "time"
)

func main() {
    switch time.Now().Weekday() {
    case time.Saturday:
        fmt.Println("Il est samedi")
    case time.Sunday:
        fmt.Println("Il est dimanche")
    default:
        fmt.Println("au boulot ! Le week-end est fini")
    }
}
```

Le switch dur à cuire

Sur d'autres langages de programmation il n'est pas toujours toléré d'utiliser des opérateurs logiques ou des opérateurs de conditions dans votre instruction `switch` . Sur Go s'est toléré, mais dans ce cas il ne faut pas rajouter votre variable après votre mot-clé `switch` (d'ailleurs vous être libre de vérifier n'importe quoi vu que le `switch` ne se base plus sur une variable).

```
package main

import (
    "bufio"
    "fmt"
    "os"
    "strconv"
    "time"
)

func main() {
    scanner := bufio.NewScanner(os.Stdin)
    fmt.Print("Votre choix : ")
    scanner.Scan()
    choix, err := strconv.Atoi(scanner.Text())
    if err != nil {
        fmt.Println("Entrez un entier !")
        os.Exit(2)
    }

    switch {
    case choix >= 2000:
        println("Ah un 2000 !")
    case choix >= 1939 && choix <= 1945:
        println("Triste année")
    case time.Now().Weekday() == time.Sunday:
        println("Dimanche !")
    default:
        println("Mauvais choix !") // Valeur par défaut
    }
}
```

```
Votre choix : 2000
Ah un 2000 !
```

```
Votre choix : 1942
Triste année
```

```
Votre choix : 0
Mauvais choix !
```