LES TABLEAUX DANS LE LANGAGE DE PROGRAMMATION GO

Pourquoi les tableaux

Imaginez que vous avez besoin de déclarer 100 variables de type int, est-ce que vous allez écrire 100 fois cette ligne ?

```
var var1 int
var var1 int
var var2 int
...
var var100 int
```

La réponse est **NON**. Ça sera long et fastidieux, rappelez vous aussi que vous êtes un bon flemmard. Donc impossible pour vous de passer votre temps à déclarer plusieurs variables !

La solution dans ce cas c'est de créer un tableau de type int. Un tableau (array en anglais) est une structure de données qui permet de stocker une collection séquentielle de taille fixe de variables du même type.

Dans ce chapitre nous allons voir que des **tableaux statiques**, c'est-à-dire des tableaux dont la taille est connue pendant la déclaration, nous verrons les tableaux dynamiques dans un chapitre dédié aux `Slice()`.

Initialisation des tableaux

Voici comment on déclare un tableau statique

```
var nomTableau [taille] type
```

Où `taille` est un nombre entier qui correspond au nombre de variables que le tableau est capable de stocker suivit du type de variable qu'il peut supporter.

Revenons à notre exemple où il fallait déclarer 100 variables de type int, en tableau statique ça donnera quelque chose comme ça :

```
var tableauInt [100]int
```

On peut déclarer n'importe quel type de tableau (int, float64 etc ...) et les variables dans votre tableau auront la même valeur par défaut que d'habitude.

Exemple :

```
package main

import (
    "fmt"
)

func main() {
    var tableauInt [10]int
    var tableauFloat [10]float32
    var tableauString [10]string
    var tableauBool [10]bool
    fmt.Println("Valeur par défaut de la variable tableauInt :", tableauInt)
    fmt.Println("Valeur par défaut de la variable tableauFloat :", tableauFloat)
    fmt.Println("Valeur par défaut de la variable tableauString :", tableauString)
    fmt.Println("Valeur par défaut de la variable tableauBool :", tableauBool)
}
```

Résultat :

```
Valeur par défaut de la variable tableauInt : [0 0 0 0 0 0 0 0 0 0]
Valeur par défaut de la variable tableauFloat : [0 0 0 0 0 0 0 0 0 0]
Valeur par défaut de la variable tableauString : [          ]
Valeur par défaut de la variable tableauBool : [false false false false false false f
```

Vous pouvez aussi surcharger les valeurs par défaut de votre tableau grâce à des accolades.

```
package main

import (
    "fmt"
)

func main() {
    var tableau1 = [5]int{1, 2, 3, 4, 5}
    fmt.Println("la taille de mon tableau1 :", len(tableau1))
    fmt.Println("les valeurs de mon tableau1 :", tableau1)
}
```

> **Information**
>
> la fonction `len()` retourne la taille de votre tableau

Résultat :

```
la taille de mon tableau1 : 5
les valeurs de mon tableau1 : [1 2 3 4 5]
```

Accéder aux éléments du tableau

Depuis l'index du tableau

les variables dans votre tableau sont accessibles depuis leur index (numéro de la case).

> **Attention**
>
> Le premier index commence toujours à la case 0 !

Voici un tableau de type string qui stocke les jours de la semaine :

```
package main

import (
    "fmt"
)

func main() {
    var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"

    fmt.Println("jours[0] =", jours[0])
    fmt.Println("jours[1] =", jours[1])
    fmt.Println("jours[2] =", jours[2])
    fmt.Println("jours[3] =", jours[3])
    fmt.Println("jours[4] =", jours[4])
    fmt.Println("jours[5] =", jours[5])
    fmt.Println("jours[6] =", jours[6])
}
```

Résultat :

```
jours[0] = lundi
jours[1] = mardi
jours[2] = mercredi
jours[3] = jeudi
jours[4] = vendredi
jours[5] = samedi
jours[6] = dimanche
```

On peut très vite se tromper en essayant d'accéder à la dernière valeur de notre tableau comme ci-dessous :

```
package main

import (
    "fmt"
)

func main() {
    var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"

    fmt.Println("jours[7] =", jours[7]) // erreur
}
```

Erreur :

```
invalid array index 7 (out of bounds for 7-element array)
```

Sauf que ça ne fonctionne pas car votre compilateur vous explique que vous être hors limites de votre tableau, l'index que vous avez rentré est non valide. Rappelez-vous que le **premier index** est toujours égal à 0, il faut donc prendre en considération le 0 quand vous calculez la taille de votre tableau, ici la valeur du dernier index est égale à 6 et non à 7.

Voici un moyen plus simple pour récupérer automatiquement la dernière valeur de votre tableau en utilisant la fonction `len()` :

```
package main

import (
    "fmt"
)

func main() {
    var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"

    fmt.Println("Dernier jour =", jours[len(jours)-1]) // taille du tableau - 1 = der
}
```

Résultat :

```
Dernier jour = dimanche
```

Boucle sur un tableau

Vous pouvez aussi parcourir la totalité de vos variables grâce à la **boucle** `for` en utilisant le mot-clé `range`.

le mot-clé `range` retourne à la fois l'index et la valeur de l'element itéré du tableau

```
package main
```

```
import (
 "fmt"
)

func main() {
 var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "

    // récupération de l'index et de la valeur
 for index, j := range jours {
  fmt.Println(j, "est le jour numéro", (index + 1), "de la semaine")
 }
}
```

Résultat :

```
lundi est le jour numéro 1 de la semaine
mardi est le jour numéro 2 de la semaine
mercredi est le jour numéro 3 de la semaine
jeudi est le jour numéro 4 de la semaine
vendredi est le jour numéro 5 de la semaine
samedi est le jour numéro 6 de la semaine
dimanche est le jour numéro 7 de la semaine
```

Boucle for len

Il est concevable aussi d'utiliser une boucle `for` classique en récupérant la taille du tableau avec la fonction `len()`

```
package main

import (
 "fmt"
)

func main() {
 var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "

 for i := 0; i < len(jours); i++ {
  fmt.Println(jours[i], "est le jour numéro", (i + 1), "de la semaine")
 }
}
```

Résultat :

```
lundi est le jour numéro 1 de la semaine
mardi est le jour numéro 2 de la semaine
mercredi est le jour numéro 3 de la semaine
jeudi est le jour numéro 4 de la semaine
vendredi est le jour numéro 5 de la semaine
samedi est le jour numéro 6 de la semaine
dimanche est le jour numéro 7 de la semaine
```

Récupération rapide et condensée

Il est possible comme sur le langage de programmation python de récupérer des valeurs de notre tableau grâce à l'opérateur `:`. Voici quelques cas d'utilisation :

- Récupérer tous les éléments de notre tableau :

```
package main

import (
 "fmt"
)

func main() {
 var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi", "
 fmt.Println(jours[:]) // on récupère tous les éléments
}
```

Résultat :

```
[lundi mardi mercredi jeudi vendredi samedi dimanche]
```

- Récupérer les trois premiers éléments de notre tableau :

```
package main

import (
    "fmt"
)

func main() {
    var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"
    fmt.Println(jours[:3]) // trois premiers éléments
```

```
}
```

Résultat :

```
[lundi mardi mercredi]
```

- Récupérer tous les résultats à partir du troisième index :

```
package main

import (
    "fmt"
)

func main() {
    var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"
    fmt.Println(jours[3:])
}
```

Résultat :

```
[jeudi vendredi samedi dimanche]
```

- Récupérer les résultats du premier jusqu'au troisième index (exclus) :

```
package main

import (
    "fmt"
)

func main() {
    var jours = [7]string{"lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"
    fmt.Println(jours[1:3])
}
```

Résultat :

```
[mardi mercredi]
```

Changer les valeurs des éléments du tableau

Il est permis de modifier la valeur d'une variable dans un tableau en récupérant l'index de l'élément en question

```
monTableau [indexDeLElement] = nouvelleValeur
```

Exemple :

```
package main

import "fmt"

func main() {
    var jours = [5]string{"Hatim", "Robert", "Inconnu", "Ahmed", "Inconnu"}

    jours[0] = "Alex" // on remplace le premier element (ici Hatim) par Alex
    fmt.Println(jours)

    jours = replaceByHatim(jours)
    fmt.Println(jours)
}

/*
    J'utilise une fonction pour vous montrer qu'il est
    possible de prendre en paramètre un tableau
    mais aussi de retourner un tableau dans une fonction
*/
func replaceByHatim(jours [5]string) [5]string {
    for i, jour := range jours {
        if jour == "Inconnu" {
            jours[i] = "Hatim" // Remplacer "Inconnu" par "Hatim"
        }
    }
    return jours
}
```

Résultat :

```
[Alex Robert Inconnu Ahmed Inconnu]
[Alex Robert Hatim Ahmed Hatim]
```

Tableau à deux dimensions

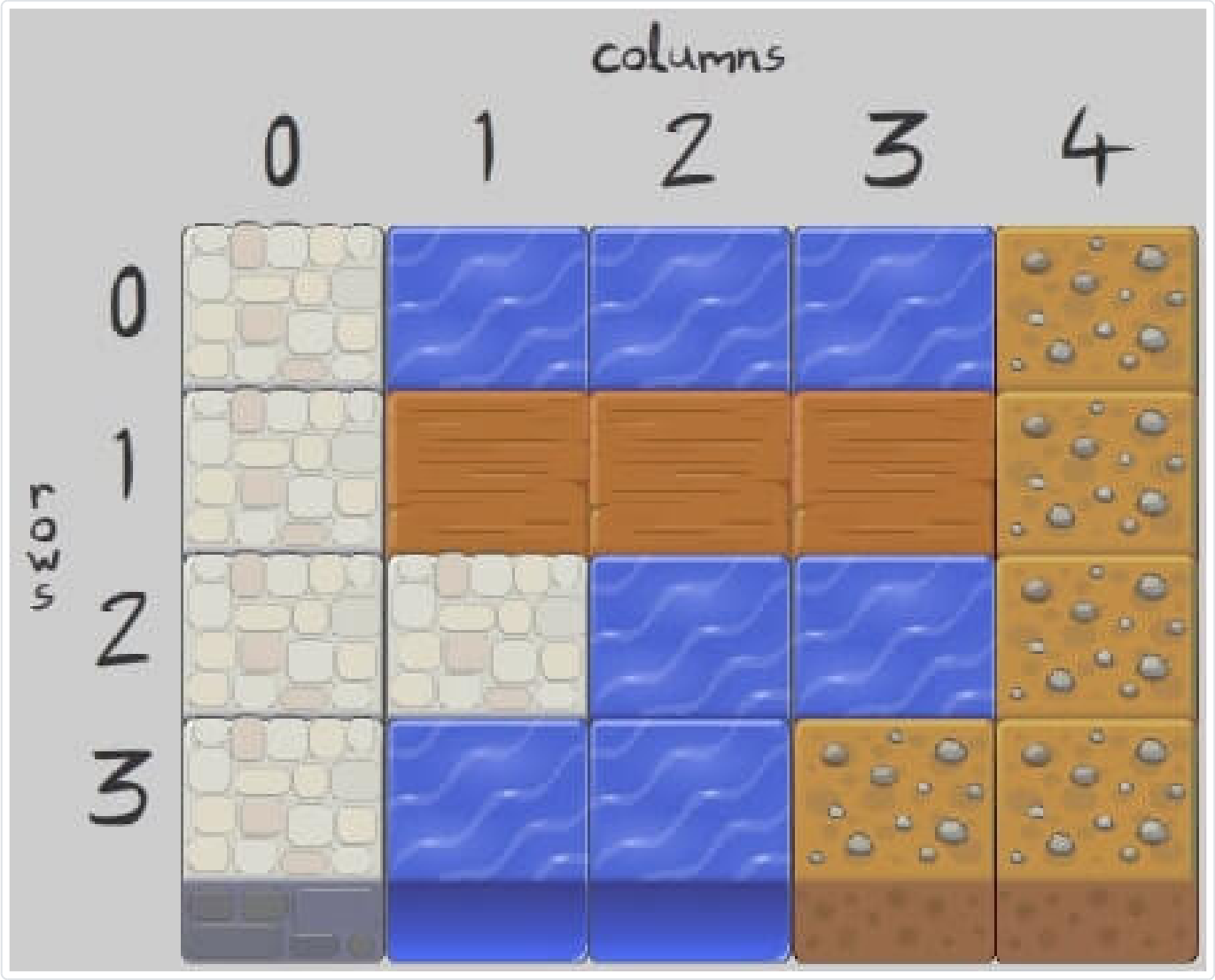
Initialisation des tableaux à deux dimensions

Un tableau à deux dimensions n'est rien d'autre qu'un tableau qui contient d'autres tableaux (des tableaux dans un tableau).

Un tableau à deux dimensions est composé de :

- `lignes` : correspond au nombre des sous tableaux inclus dans le tableau principal.
- `colonnes` : correspond au nombre d'éléments dans les sous tableaux.

Un tableau à deux dimensions peut par exemple être utilisé pour la construction d'une map dans un jeu 2D.



« Map d'un jeu construite depuis un tableau à double dimensions »

Exemple :

```
package main

import "fmt"

func main() {
 const (
  maxLigne   int = 3 // 3 sous tableaux
  maxColonne int = 4 // 4 éléments pour chaque sous tableau
 )

    var tableau [maxLigne][maxColonne]int // Création d'un tableau à double dimension

 fmt.Println(tableau)
}
```

Résultat :

```
[[0 0 0 0] [0 0 0 0] [0 0 0 0]]
```

Modifier les valeurs des tableaux à deux dimensions

Pour modifier la valeur d'un tableau à deux dimensions, il nous faut :

- `index de la ligne` : permet de récupérer un tableau bien spécifique dans le tableau principal (ex : s'il est égal à 0 alors on récupère le premier tableau du tableau principal).
- `index de la colonne` : récupérer un élément depuis tableau spécifique.

```
package main

import "fmt"

func main() {
 const (
  maxLigne   int = 3
  maxColonne int = 3
 )

 var doubleTableau [maxLigne][maxColonne]int

 fmt.Println(doubleTableau)

 fmt.Println("----------------------")

 //modification de la ligne 3, colonne 2
 doubleTableau[2][1] = 5 // modification du 2eme élément du 3eme tableau
 fmt.Println(doubleTableau)

 chaine := "Hello"
 fmt.Println(chaine)
}
```

Résultat :

```
[[0 0 0] [0 0 0] [0 0 0]]
```

```
----------------------
[[0 0 0] [0 0 0] [0 5 0]]
```