# *Outils informatiques*

# *3. Les listes de listes*

DESS TEXTE

# *Introduction*

- Nous avons vu comment manipuler des objets Perl par référence (ou adresse).

- Un cas particulier de structures employant les références sont les listes de listes.

- Ce sont des tableaux à deux dimensions,
exemple :

```
@LoL = ([1,2,3],["yannick","eric","joseph"],[4,5]);
```

Ici @LoL contient 3 références à des tableaux anonymes.

# *Plan du chapitre*

Chapitre 3 : manipulation de listes de listes

1. Déclaration et accès aux listes de listes
2. Ajout d’éléments à une liste de listes
3. Accès et affichage de certains éléments d’une liste de listes
4. Affectation par tranches

# 1. Déclaration et accès aux listes de listes

Rappel (chapitre 2) : deux façons de stocker une liste de listes :

1. au moyen d'une liste contenant des références sur des listes, exemple :

```
@LoL = (["homer","marge","bart"],["x","y","z"]);
```

2. au moyen d'un scalaire contenant l'adresse d'une liste (anonyme) contenant des références sur des listes,
exemple :

```
$refLoL = [["homer","marge","bart"],["x","y","z"]];
```

- Accès cas 1 : `$LoL[0][2];`
- Accès cas 2 : `$refLoL->[1][0];`

# 2. Ajout d'éléments à une liste de listes (1)

## Ajout de listes à une liste de listes

**Exemple 1** : création à partir d'un fichier

```
while($ligne=<FICH>) {
  chop($ligne);
  @tmp = split(/ /, $ligne);
  push(@LoL, [@tmp]);
}
```

**Exemple 2** : idem sans table temporaire

```
while($ligne=<FICH>) {
  chop($ligne);
  push(@LoL, [split(/ /, $ligne)]);
}
```

# 2. Ajout d'éléments à une liste de listes (2)

**Exemple 3** : création au moyen d'une fonction

```
for $i (1..10){
  @tmp = &fonction($i);
  $LoL[$i] = [@tmp];
}
```

**Exemple 4** : idem sans table temporaire

```
for ($i=1,$i<=10,$i++){
  $LoL[$i] = [&fonction($i)];
}
```

# 2. Ajout d’éléments à une liste de listes (3)

**Remarques :**

- attention aux "[ ]" (règle de création de référence anonyme) !
- il existe deux types d’assignation, l’assignation directe au moyen de `$LoL[$i]`, et l’assignation "non directe" au moyen de la fonction `push`.
- qu’en serait-il si nous avions utilisé une référence à une liste de liste `$refLoL` ?

# 2. Ajout d'éléments à une liste de listes (4)

**Ajout de colonnes à une (ou plusieurs) liste(s) de la liste de listes**
(Rappel : Liste de liste $\equiv$ table à 2 dimensions)

**Exemple 1** : ajout d'éléments à une *ligne* particulière

```
for ($y=7,$y<=10,$y++){
  $LoL[4][$y] = &fonction($y);
}
```

**Exemple 2** : ajout d'éléments à plusieurs *lignes*

```
for ($x=1,$x<=10,$x++){
  for ($y=7,$y<=10,$y++){
    $LoL[$x][$y] = &fonction($x,$y);
  }
}
```

# 2. Ajout d'éléments à une liste de listes (5)

**Exemple 3** : ajout d'un élément à plusieurs *lignes*

```
for $x (3,7,9){
  $LoL[$x][20] = &fonction2($x);
}
```

**Exemple 4** : ajout d'éléments à une *ligne* par assignation non directe

```
push (@{$LoL[0]},"wilma","betty");
```

Ajout de 2 éléments à la fin de la première *ligne* de notre liste de listes.

# *3. Accès et affichage de certains éléments d'une liste de listes (1)*

**Affichage de certains éléments de la liste de listes**

**Exemple 1** : affichage de l'élément d'adresse "ligne = i, colonne = j"

```
print $LoL[$i][$j];
```

**Exemple 2** : affichage de la ligne i

```
print @{$LoL[$i]};
```

**Remarque :** @LoL contient des références !

**Exemple 3** : affichage ligne par ligne via une boucle *foreach*

```
foreach $elt (@LoL) {
  print @{$elt};
}
```

# *3. Accès et affichage de certains éléments d'une liste de listes (2)*

**Exemple 4** : affichage ligne par ligne via une boucle *for*

```
for($i=0,$i<=$#LoL,$i++){
  print @{$LoL[$i]};
}
```

**Exemple 5** : affichage élément par élément via deux boucles *for*

```
for($i=0,$i<=$#LoL,$i++){
  for($j=0,$j<=$#{$LoL[$i]}){
  print $LoL[$i][$j];
}
```

**Rappel** : $#TAB contient l'indice du dernier élément du tableau @TAB.

# *4. Affectation par tranches (1)*

## a) Affectation par une tranche de ligne

**Exemple 1** : au moyen d'une boucle *for*

```
for($y=7,$y<13,$y++){
  push(@newL,$LoL[4][$y]);
}
```

**Exemple 2** : au moyen de la notation *[x..y]*

```
@newL = @{$LoL[4]}[7..12];
```

**Remarque** : ici @newL est une liste contenant les éléments d'indice 7 à 12 de la 5e ligne de @LoL.

# *4. Affectation par tranches (2)*

## b) Affectation par une tranche à deux dimensions

**Exemple 1** : au moyen de deux boucles *for* (assignation directe)

```
for($startx=$x=4;$x<=8;$x++){
  for($starty=$y=7,$y<13,$y++){
    @newLoL[$x-$startx][$y-$starty] = $LoL[$x][$y];
  }
}
```

**Exemple 2** : au moyen de la notation *[x..y]*

```
for($x=4;$x<=8;$x++){
  push(@newLoL , [@{$LoL[$x]}[7..12]]);
}
```

**Remarque** : ici @newLoL est une liste de listes contenant les éléments d'indice 7 à 12 des lignes 4 à 8 de @LoL.

# *Conclusion*

- Les listes de listes représentent une structure très utile et courante en Perl.

- Elles disposent de la notation simplifiée :
`$LoL[$Ligne][$Colonne]`
(au lieu de `${$LoL[$Ligne]}[$Colonne]`).

- Elles permettent notamment de travailler sur les mots d'un fichier texte (cf exercice).