Cours Apl 04 : Premières fonctions primitives et manipulations d'objets.

- L'affectation : ←

Cette fonction permet d'affecter la valeur de l'argument droit à l'objet de gauche.
Exemple : pour donner la valeur 35 à la variable dpt on procède comme suit :
dpt ← 35
L'affectation étant une fonction comme les autres, on peut la placer où bon nous semble, et plusieurs fois si nécessaire, sur la même ligne.
Exemple :
total ← +/liste ← 2 3 5

Attention !!!
Si plusieurs instructions sont placées sur la même ligne, apl commence par effectuer le groupe le plus à droite, puis en passe le résultat à la fonction immédiatement à sa gauche.
Dans ce cas, Apl a
- d'abord affecté 2 3 5 à liste
- puis il en a calculé la somme "+/"
- enfin, " ← " l'a affectée à total.

De même, il est courant d'initialiser plusieurs variables en une ligne.
Ainsi au lieu d'écrire :
v1 ← 0
v2 ← 0
v3 ← 0

on écrira :
v1 ← v2 ← v3 ← 0

- Affectation en modification :

Cette syntaxe est utilisée lorsqu'on réaffecte à une variable son contenu modifié.
Variable fonction ← argument

Exemple : au lieu d'écrire a ← a × 2
on peut écrire a × ← 2
ce qui signifie : a reçoit lui même multiplié par 2.

- Tailles et dimensions des variables :

Comme nous l'avons déjà aperçu au chapitre 3, la principale fonction de gestion des tailles est le ⍴(rhô).
En mode monadique, elle rend la taille de l'objet, ou plus exactement le vecteur de ses dimensions.
Par exemple : ⍴ 'coucou'
rend 6, qui est un vecteur de 1 élément.
Ainsi, ⍴⍴ 'coucou'
rendra 1
Il est donc très facile de savoir quel type d'objet on manipule (scalaire, vecteur, matrice , ...).

En mode dyadique, la fonction ⍴ rend un objet dont la dimension est définie à gauche, et dont le contenu est défini à droite.
Exemple : 2 3 ⍴ 1 2 3 4 5 6
rend cette matrice numérique :
1 2 3
4 5 6

Affectons cette expression à m1 :
m1 ← 2 3 ⍴ 1 2 3 4 5 6

D'après vous, quel sera le résultat de
(⍴ m1)⍴ 0

On obtient une nouvelle matrice de même taille que m1 remplie de 0.
Pour constituer l'objet de dimension G (argument gauche), ← boucle sur D pour remplir le nouvel objet.

Ainsi, pour constituer cette matrice :
abcd
abcd
abcd
il suffit d'écrire : 3 4 ⍴ 'abcd'

Si le rhô permet d'allonger les objets, il permet également de les raccourcir.
Ainsi
3 ⍴ 1 2 3 4 5 donne
1 2 3

- Fonctions voisines : take et drop

Le ↑ (take) peut également augmenter ou diminuer les tailles des objets mais ne peut pas modifier leur nombre de dimensions.
Par exemple
2 3 ⍴ 4 5 6 produit cette matrice :
4 5 6
4 5 6

En revanche
2 3 ↑ 4 5 6 produira un "Rank Error". En effet, 4 5 6 est un vecteur (objet de rang 1) et 2 3 indique les dimensions d'une matrice. Or, take ne peut pas changer le rang d'un objet, mais seulement ses dimensions, sans en changer le nombre.
Soit Mat1 ← 4 4 ⍴ 1 2 3
Mat vaut :
1 2 3 1
2 3 1 2
3 1 2 3
1 2 3 1

2 3 ↑ Mat1 donne
1 2 3
2 3 1

Dans cet exemple, on ne garde que les 2 premières lignes et les 3 premières colonnes de Mat1.

Si on signe négativement l'argument gauche, take prend alors les n derniers éléments.
Exemple :
  ¯3 ↑ 1 2 3 4 5 rend
3 4 5

- **↓ (Drop),** quant à lui, abandonne le nombre d'élément indiqué à gauche.
Par exemple
  1 1 ↓ Mat1 rend :
3 1 2
1 2 3
2 3 1
on a abandonné la 1ère ligne et la 1ère colonne.
On obtiendrait le même résultat avec
-3 -3 ↑ Mat1

Travaux pratiques :

0. Chargez votre Ws de travaux pratiques :
)load c:\Mes documents\pratique-apl

1. Affichez le vecteur Monvec créé précédemment.
A partir de Monvec et en utilisant la fonction ⍴ affichez cette suite de nombres : 10 20 30 40 10 20 30 40

2. Affichez une matrice de 4 lignes et 6 colonnes constituée à partir des éléments de Monvec.
Recommencez en affectant le résultat à Mnum2.

3. Additionnez Mnum1 et Mnum2.
Pourquoi cela ne fonctionne-t-il pas ?
Additionnez Mnum1 avec les 3 premières lignes et les 4 premières colonnes de Mnum2.
- additionnez Mnum1 avec les 3 dernières lignes et les 4 dernières colonnes de Mnum2
- additionnez Mnum1 avec les 3 premières lignes et les 4 dernières colonnes de Mnum2
- additionnez Mnum1 avec les 3 dernières lignes et les 4 premières colonnes de Mnum2

4. Affichez la taille de Mnum1, puis celle de Mnum2.

5. Affichez le premier chiffre en haut à gauche de Mnum1

6. Affichez le dernier chiffre en bas à droite de Mnum1

7. Affichez la somme des chiffres de la première et de la dernière colonne de Mnum2

8. Affichez un vecteur de 3 éléments (3 chiffres identiques) à partir de la somme du premier élément de Mnum1 et du dernier de Mnum2.

9. Sans saisir aucun 0, affichez cette suite de nombres : 4 5 0 0 0

10. Comme pour le point précédent, affichez 0 0 0 4 5

11. Sauvez votre travail :
)save

Solutions :

1. Affichez le vecteur Monvec créé précédemment.

```
Monvec
```

A partir de Monvec et en utilisant la fonction ⍴ affichez cette suite de nombres : 10 20 30 40 10 20 30 40

```
8 ⍴ Monvec
```

ou

```
(2x ⍴ Monvec) ⍴Monvec
```

2. Affichez une matrice de 4 lignes et 6 colonnes constituée à partir des éléments de Monvec.

```
4 6 ⍴ Monvec
```

Recommencez en affectant le résultat à Mnum2.

```
Mnum2← 4 6 ⍴ Monvec
```

3. Additionnez Mnum1 et Mnum2.

```
Mnum1 + Mnum2
```

LENGTH ERROR

Pourquoi cela ne fonctionne-t-il pas ?

Parce que ces 2 objets n'ont pas la même dimension, ce qui est une condition indispensable pour en additionner les éléments.

Additionnez Mnum1 avec les 3 premières lignes et les 4 premières colonnes de Mnum2.

```
Mnum1+3 4↑Mnum2
```

- additionnez Mnum1 avec les 3 dernières lignes et les 4 dernières colonnes de Mnum2

```
Mnum1+¯3 ¯4↑Mnum2
```

- additionnez Mnum1 avec les 3 premières lignes et les 4 dernières colonnes de Mnum2

```
Mnum1+3 ¯4↑Mnum2
```

- additionnez Mnum1 avec les 3 dernières lignes et les 4 premières colonnes de Mnum2

```
Mnum1+¯3 4↑Mnum2
```

4. Affichez la taille de Mnum1, puis celle de Mnum2.

```
⍴Mnum1
⍴Mnum2
```

5. Affichez le premier chiffre en haut à gauche de Mnum1

```
1 1↑Mnum1
```

6. Affichez le dernier chiffre en bas à droite de Mnum1

```
¯1 ¯1↑Mnum1
```

7. Affichez la somme des chiffres de la première et de la dernière colonne de Mnum2

```
(4 1↑Mnum2)+4 ¯1↑Mnum2
```

8. Affichez un vecteur de 3 éléments (3 chiffres identiques) à partir de la somme du premier élément de Mnum1 et du dernier de Mnum2.

```
3⍴(1 1↑Mnum1)+¯1 ¯1↑Mnum2
```

9. Sans saisir aucun 0, affichez cette suite de nombres : 4 5 0 0 0

```
5 ↑ 4 5
```

10. Comme pour le point précédent, affichez 0 0 0 4 5

```
¯5 ↑ 4 5
```