Cours Apl 09 - Ecrivez vos fonctions et programmes.

Comme nous l'avons entrevu au chapitre 3, Apl vous permet d'écrire vos propres fonctions. Vos fonctions appelleront des fonctions primitives apl et elles pourront également s'appeler entre elles.

Comme les fonctions primitives, les fonctions utilisateurs admettent 0, 1 ou 2 arguments et rendent (ou pas) un résultat.

La première ligne d'une fonction se présente généralement comme suit :

```
r ← g Fonction d;loc1;loc2
```

- r est le nom de la variable à laquelle est affecté le résultat qui sera rendu par la fonction. Cette variable n'est vue que depuis la fonction ou depuis celles qu'elle appellerait elle même.
Elle est invisible pour la fonction appelante. Elle est donc dite "locale".

- g et d sont les noms des variables portant les arguments gauche et droit.
Ces 2 variables sont également locales.

- loc1; loc2 : Il est fortement recommandé de localiser les variables dont vous aurez besoin dans vos fonction afin qu'une fonction appelée ne vienne pas modifier accidentellement une variable dans une fonction appelante.
Pour ce faire, il suffit de déclarer les noms des locales séparés par des points-virgules.

Un exemple sera plus parlant.
Ecrivons 2 fonctions, l'une étant appelée par l'autre.
Nouvelles fonctions et signes Apl :

- ***** : Puissance
- ⍝ : Commentaire. Tout ce qui est saisi à droite n'est pas pris en compte dans le programme.

Pour éditer (saisir, modifier) une fonction, vous devez la saisir dans l'éditeur. Pour cela, vous avez plusieur solutions :
- Tapez)ed nom_fonction
- Tapez le nom de la fonction puis
  - double cliquez dessus
  - placez le curseur dessus et faites Shift Entrée.

Pour sortir de l'éditeur en validant votre saisie, il vous suffit de faire Escape. Pour sortir sans prendre en compte vos modifications, il faut faire Shift Escape.
Vous pouvez bien sûr aussi passer par les menus.

- Notre première fonction sera la fonction **Arrondir**.
Elle admettra à gauche le nombre de chiffres après la virgule et à droite l'objet numérique dont les éléments doivent être arrondis. En voici le listing :

```
r ← g Arrondir d;tmp
⍝ Arrondir d à g chiffres après la virgule
tmp ← d × 10 * g     ⍝ Décaler la virgule de g rangs à droite
tmp ← ⌊ .5 + tmp     ⍝ Ajouter 0.5 et arrondir à l'entier inférieur
r ← tmp ÷ 10 * g     ⍝ Ramener la virgule à gauche
⍝ En 1 seule ligne : r ← (⌊ .5 + d × 10 * g)  ÷ 10 * g
```

Une fois votre saisie terminée, il ne vous reste plus qu'à tester le résultat :

```
      2 Arrondir 2 2 ⍴ 88.322 .899 5 7.3
88.32    .9
 5      7.3
```

La deuxième fonction est la fonction **Repartion**
Elle admet une variable numérique en argument droit et rend un objet de même structure où les valeurs sont remplacées par leur poids en % dans l'objet.

```
r ← Repartition d
⍝ Répartition des valeurs de d
r ← 100 × d ÷ +/,d
```

De droite à gauche :
- on vectorise d, puis on en calcule la somme
- on divise chaque élément de d par cette somme et on multiplie le tout par 100 pour avoir les valeurs en %.

Testons notre fonction :

```
      Repartition 2 2 ⍴ 10 10 30 20
14.28571429 14.28571429
42.85714286 28.57142857
```

Maintenant, nous voulons afficher ce résultat avec seulement 2 chiffres après la virgule :

```
      2 Arrondir Repartition 2 2 ⍴ 10 10 30 20
14.29   14.29
42.86   28.57
```

Si vous avez tapé ces exemples, n'oubliez pas de sauvegarder votre travail avec la commande) save

Travaux pratiques :

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

1. Si vous ne l'avez pas déjà fait, écrivez et testez les 2 fonctions du cours.

2. Ecrivez la fonction Totmat qui admet une matrice numérique en argument droit et rend cette même matrice avec à droite le total des lignes et en bas le total des colonnes.

3. Ecrivez la fonction ColleV qui colle la matrice d sous la matrice g en ajoutant les éventuelles colonnes de zéros ou de blancs à droite de la matrice la plus étroite.

4. Ecrivez la fonction ColleH qui colle la matrice d à droite de la matrice g en ajoutant les éventuelles colonnes de zéros ou de blancs sous la matrice la moins haute.

5. Sauvez votre travail :
)save

Solutions :

1. Si vous ne l'avez pas déjà fait, écrivez et testez les 2 fonctions du cours.

Voir cours.

2. Ecrivez la fonction Totmat qui admet une matrice numérique en argument droit et rend cette même matrice avec à droite le total des lignes et en bas le total des colonnes.

```
 r←Totmat d
⍝ Ajouter le total par lignes et colonnes à la matrice d
 r←d,+/d          ⍝ Totaux lignes
 r←r,[1]+/[1]r    ⍝ Totaux colonnes
```

3. Ecrivez la fonction ColleV qui colle la matrice d sous la matrice g en ajoutant les éventuelles colonnes de zéros ou de blancs à droite de la matrice la plus étroite.

```
 r←g ColleV d;dim
⍝ Coller la matrice d sous la matrice g
 dim←⌈/(¯1↑⍴g),¯1↑⍴d  ⍝ Chercher la plus grande largeur
 r←(dim↑[2]g),[1]dim↑[2]d
```

4. Ecrivez la fonction ColleH qui colle la matrice d à droite de la matrice g en ajoutant les éventuelles colonnes de zéros ou de blancs sous la matrice la moins haute.

```
 r←g ColleH d;dim
⍝ Coller la matrice d à droite de la matrice g
 dim←⌈/(1↑⍴g),1↑⍴d    ⍝ Chercher la plus grande hauteur
 r←(dim↑[1]g),dim↑[1]d
```