Introduction à la programmation *via* SCILAB

13 Octobre 2008

Note concernant l'aide

En plus de votre encadrant(e) favori(e), SCILAB fournit un bon moyen pour obtenir la réponse à vos questions sur certaines fonctions : la commande `help`. Par exemple pour obtenir de l'aide sur la fonction cosinus, il suffit de taper `help cos`. Quand vous rencontrez une fonction inconnue dans un sujet de TP, utilisez l'aide ...puis votre encadrant(e) !

Note concernant les commentaires

Il est parfois (souvent ?) utile de laisser des petits commentaires sur les bouts de code que l'on écrit. On appelle ça un commentaire. Pour SCILAB, tout ce qui suit deux "slash" `//` est un commentaire, et donc ignoré.

```
//ceci est un commentaire
a=1
//a=2
a == 2 // ce test va échouer
```

N'hésitez pas à semer des commentaires un peu partout dans vos programmes, ils seront utiles à la relecture !

1 Les tableaux

SCILAB use et abuse de la notion de matrice. Comme vous n'avez pas encore étudié les matrices (un peu quand même ?), on n'en parlera pas tout de suite. Par contre les tableaux, un sous ensemble des matrices, sont faciles à comprendre et très utilisés en programmation.

Si on garde l'image qu'une variable est une boite qui contient une valeur, un tableau est une boite de boites. On peut ranger pleins de boites (de même type !) dans un tableau. Un tableau est une suite d'éléments entre crochets `[]`.
Exemple 1: Définition de tableau

```
a=1 // tiens, une boite
b=2 // une autre boite !
c=3 // ça traine de partout !
d= [ a b c ] // hop on met tout dans une nouvelle boite
```

On peut aller plus vite et directement faire

```
d=[1 2 3]
// ou encore
d=[1,2,3]
```

Et on peut aller encore plus vite en précisant un pas

```
d=[1:1:3]    // créer toutes les valeurs de 1 à 3 en ajoutant 1 à chaque fois
d=[1:3]      // le pas par défaut est 1
e=[0:2:100]  // tous les nombres pairs de 0 à 100
```

On peut lire les éléments d'un tableau en utilisant leur indice, c'est à dire leur place dans le tableau. Le premier élément est a la première place !
Exemple 2: Tableau de chaine de caractères

```
noms=["Nicolas" "Segolene" "francois" "mr X"]
femme=noms(2)
myster=noms(4)
hommes=[ noms(1) noms(3)]
```

On peut aussi changer le contenu d'un tableau de cette façon !

```
noms(1)="président"
```

On peut faire beaucoup d'autres choses avec les tableaux, mais pas cette fois ! Enfin si on peut quand même faire quelques petits exercices !

1.1 Exercices

1.1.1 Trions un petit tableau

La commande `rand(1,n)` crée un tableau de n éléments aléatoires. Créer un tableau `petit_tableau` contenant 2 éléments pris au hasard.

Écrire un petit programme qui vérifie si le premier élément de ce tableau est plus petit que le deuxième.

Écrire un autre programme qui trie ces deux éléments par ordre croissant.

Maintenant jeter un coup d'œil a la fonction `sort` ...

1.1.2 question de taille

Écrire un programme qui vérifie si deux tableaux contiennent le même nombre d'éléments. La fonction `size` peut être utile.

1.1.3 Viens faire le mélange des couleurs

On a dit qu'on ne pouvait pas faire d'opérations entre "boites" de "types" différents. Par exemple on ne peut pas additionner un entier et une chaîne de caractères. Et pour les tableaux ? Quel serait le sens de

1. une somme de tableaux
2. une différence de tableaux
3. un produit de tableaux

4. le produit d'un tableau et d'une valeur
5. la somme d'un tableau et d'une valeur

Vérifiez vos idées sous SCILAB

2 Les polynômes

Nous avons déjà vu quelques types de bases comme les entiers, les réels, les booléens, les chaînes de caractères. SCILAB est capable de manipuler des types beaucoup plus complexes, comme les polynômes.

Pour décrire un polynôme, il faut d'abord créer le paramètre formel utilise dans le polynôme. En mathématiques, on utilise généralement la variable x a cet effet, mais SCILAB réserve `%z` et `%s` pour les polynômes.
Exemple 3: Cette commande définit un polynôme et l'enregistre dans la variable `u`

```
u=1+%z^2
```

Si on préfère utiliser une autre variable que `%s` ou `%z`. Dans ce cas il faut donner sa définition.
Exemple 4: Entre autres usages, la fonction `poly` le permet.

```
x=poly(0,'x') // définition de x
u=1+x^2       // c'est plus clair comme ça!
```

La fonction `poly` permet aussi de créer des polynômes a partir de leurs racines
Exemple 5:

```
v=poly([1 -1],'x')
```

ou a partir de leurs coefficients

```
w=poly([1 0 2],'x','coeff')
```

Pour évaluer un polynôme en une certaine valeur, il faut utiliser la fonction `horner`.
Exemple 6: Pour avoir la valeur du polynôme précédent pour $x = 2$, on utilisera

```
horner(w,2)
```

2.1 Exercices

Quelques petits exos d'application directe ...

2.1.1 Tout doux

Définir un polynôme `paul` ayant pour racines 1 2 3 4 5.

Combien vaut se polynôme en 0, en 1, en 100, en π et en $+\infty$?

Pour quelle(s) valeur(s) `paul` vaut-il 3 ?

Aide: il existe une fonction `roots`

2.1.2 Un poil moins doux !

Définir les polynômes $y1 = x$ et $y2 = x^2 - 4x + 5$.

Quels sont leurs points d'intersection ?

Aide: la fonction `roots` n'a pas disparu

2.1.3 Un poil plus dur

Définir les polynômes $u = 1 + z^2$ et $v = 1 + 3z^2$. Quelles sont leurs racines respectives ?

Calculer $w = u * v$. Quelles sont les racines de w ? Cela vous semble-t-il normal ?

Dériver les polynômes w, u, v, vérifier la formule de dérivation de produit de fonctions.

Aide: la fonction `derivat` fait bien ce qu'elle semble faire