Feuille de TP n° 4

Structures répétitives : boucle `for`

Ce TP accompagne le chapitre 2 (Informatique et Algorithmique) : **Programmation en Scilab**. Créez le dossier `..\ECS1B_TPInfo\TP4\` et faites-en le répertoire courant de Scilab. Tous les scripts devront être sauvegardés dans ce dossier.

I Entrainement

En général, la construction d'une boucle `for` au sein d'un programme se base sur quatre étapes :

- On identifie les variables d'entrées (qui proviennent du début du programme ou qui sont fournies par l'utilisateur via la commande `input`).
- On définit des variables qui vont être modifiées lors de la boucle `for`. C'est l'étape d'initialisation.
- On écrit la boucle en tant que telle (`for`$\cdots$`end`). Elle utilise les variables d'entrées et, à chaque étape de la boucle, les variables définies à l'étape d'initialisation sont mises à jour.
- On renvoie les variables de sorties (affichées éventuellement dans une phrase via la commande `disp`) qui sont en général des fonctions des quantités calculées pendant la boucle.

• Étudier en détail l'exemple du chapitre 2 (calcul de la somme $\sum_{i=1}^{n} i^p$, n et p étant fournis par l'utilisateur).

• Écrire un script contentant le programme suivant :

```
n=input('Entrer un entier strictement positif : ')
q=input('Entrer un réel : ')
u=1; s=1;
for  i=1:n
     u=u*q;
     s=s+u;
end
disp('La somme des ' + string(q) + '^k pour k allant de 0 à ' + string(n)
         + ' est ' + string(s) + '.')
```

Analyser ce programme en détail : identifier les quatre étapes recensées en début de paragraphe et ajouter des commentaires au programme (on pourra s'aider d'un tableau comme dans le chapitre 2). Enregistrer-le sous le nom `SommeGeom.sce` et exécuter-le sans écho dans la console (tester-le avec plusieurs valeurs de `n` et `q`) et comparer avec la formule du cours.

• Écrire un script contentant le programme suivant :

```
for  j=1:10
     t=0;
     for  i=1:10
          t=t+j;
          disp(string(i)+'x'+string(j)+'='+string(t));
     end
     disp('########')
end
```

Après avoir identifié ce que fait ce programme, enregistrer-le avec nom approprié et exécuter-le sans écho dans la console. Ajouter des commentaires au code et modifier le programme pour qu'il affiche une phrase précisant ce qu'il fait. Enregistrer et exécuter-le sans écho à nouveau.

II Exercices

Commencez par taper `clear` *dans la console. Pour vous y retrouver plus facilement, veuillez écrire*

- *en commentaire au début de chaque programme le numéro du TP et de l'exercice concerné.*
- *sur cette feuille de TP, le nom que vous avez donné au programme (extension en* `.sce`*).*

Vous testerez dans la console Scilab tous les programmes que vous écrirez (sans oublier de les exécuter au préalable).

Exercice 1. Écrire un programme qui demande à l'utilisateur un entier naturel n et qui calcule $n!$ en utilisant une structure répétitive.

Exercice 2. Écrire un programme qui demande un entier naturel n et qui calcule $\displaystyle\sum_{1\leqslant i<j\leqslant n} \frac{1}{i+j}$.

Exercice 3. Pour tout $n \in \mathbb{N}$, posons $\displaystyle S_n = \sum_{1\leqslant i\leqslant k\leqslant n} (k^2 - i + 1)$

1) Calculer S_n en fonction de $n \in \mathbb{N}$.
2) Écrire un programme qui demande à l'utilisateur un entier naturel n et calcule S_n (sans utiliser la formule précédente).
3) Modifier le programme afin de faciliter la comparaison entre la valeur théorique de S_n (donnée par la formule) et la valeur calculée par le programme.

Exercice 4. Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par $u_0 \in \mathbb{R}$ et, pour tout $n \in \mathbb{N}$, $u_{n+1} = \cos\left(e^{u_n}\right) + \dfrac{u_n}{2}$. Écrire un programme qui demande à l'utilisateur un entier naturel n et un réel u_0 et qui calcule u_n.

Exercice 5. On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 \in \mathbb{R}_+^*$, $u_1 \in \mathbb{R}_+^*$ et, pour tout $n \in \mathbb{N}$, $u_{n+2} = \ln(u_{n+1} + \sqrt{n + u_n})$. Écrire un programme qui demande à l'utilisateur un entier naturel n et deux réels u_0, u_1 et qui calcule u_n.

Exercice 6. Soient $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ deux suites définies par $u_0 = -2$, $v_0 = 1$ et

$$\forall n \in \mathbb{N}, \qquad u_{n+1} = 5u_n + 4v_n \qquad \text{et} \qquad v_{n+1} = 4u_n + 5v_n.$$

Ecrire un programme qui prend en entrée un entier naturel n et qui calcule u_n et v_n. Comparer avec le résultat théorique de l'exercice 3 de la feuille d'exercice n° 5.

Exercice 7 (Algorithme de Hörner). Soient $x \in \mathbb{K}$ et $P = \displaystyle\sum_{k=0}^{n} a_k X^k \in \mathbb{K}[X]$. On souhaite demander à un ordinateur de calculer $P(x) = \displaystyle\sum_{k=0}^{p} a_k x^k$.

1) Implémenter en `Scilab` un algorithme naturel permettant de calculer $P(x)$.
2) Cette méthode naïve est assez lente : elle nécessite $1 + 2 + \cdots + n = \dfrac{n(n+1)}{2}$ multiplications et n additions. L'algorithme de Hörner consiste en l'écriture de $P(x)$ sous la forme

$$P(x) = a_0 + x(a_1 + x(a_2 + x(a_3 + \cdots + x(a_{n-1} + xa_n)))).$$

Implémenter en `Scilab` l'algorithme de Hörner. Combien nécessite-il d'additions et de multiplications ? Est-il plus rapide ?

Exercice 8. Pour tous $x \in \mathbb{R}$ et $n \in \mathbb{N}$, notons $\displaystyle S_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$.

1) Écrire un programme qui demande à l'utilisateur un entier naturel n et un réel x et qui calcule $S_n(x)$ (sans utiliser le symbole $\wedge$).
2) Comparer $S_n(x)$ et $\exp(x)$ pour plusieurs valeurs de $x \in \mathbb{R}$ et $n \in \mathbb{N}$. Commenter.

Exercice 9 (conjecture de Syracuse). La suite de Syracuse est la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 \in \mathbb{N}$ et, pour tout $n \in \mathbb{N}$,

$$u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si} \quad u_n \text{ est pair,} \\ 3u_n + 1 & \text{sinon.} \end{cases}$$

1) Caractériser la suite de Syracuse quand $u_0 = 1$.
2) Écrire un programme qui demande à l'utilisateur d'entrer deux entiers naturels n et u_0 et qui affiche les $n+1$ premiers termes $u_0, \ldots, u_n$ de la suite de Syracuse (on pourra par exemple retourner un vecteur contenant ces $n+1$ valeurs).
3) Tester le programme pour différentes valeurs. Que peut-on conjecturer ?

La conjecture de Syracuse n'est pas démontrée à ce jour... mais elle a été vérifiée par ordinateur au moins jusque l'entier $N < 1,25 \times 2^{62}$.

Exercice 10. Pour tout $n \in \mathbb{N}^*$, posons $S_n = \displaystyle\sum_{k=1}^{n} \frac{(-1)^{k+1}}{k}$. Nous montrerons dans le chapitre *Dérivées successives et formules de Taylor* que $(S_n)_{n\in\mathbb{N}^*}$ converge vers $\ln(2)$ et que, pour tout $n \in \mathbb{N}^*$, $|S_n - \ln(2)| \leqslant \dfrac{1}{n+1}$.

1) Déterminer un rang n tel que, $|S_n - \ln(2)| \leqslant 0,01$.
2) Écrire un programme qui détermine une valeur approchée de $\ln(2)$ à 10^{-2} près.