Travaux pratiques sur Maple :: Fiche n° 6

PROGRAMMATION CONDITIONNELLE ET ITÉRATIVE EN INTERACTIF

1. Un test à plusieurs alternatives est programmé généralement comme suit

```
if EXPRESSION BOOLÉENNE 1 then BLOC INSTRUCTIONS 1
        elif EXPRESSION BOOLÉENNE 2 then BLOC INSTRUCTIONS 2 ...
        elif EXPRESSION BOOLÉENNE N then BLOC INSTRUCTIONS N
        else BLOC INSTRUCTIONS N+1 end if;
```

N.B. **elif** est la contraction de **else if**

2. Une boucle **for** possède les deux syntaxes générales suivantes

```
for COMPTEUR from DÉBUT by PAS to FIN while EXPRESSION BOOLÉENNE
                                    do BLOC INSTRUCTIONS end do;
for COMPTEUR in LISTE OU ENSEMBLE while EXPRESSION BOOLÉENNE
                                    do BLOC INSTRUCTIONS end do;
```

N.B. En cas d'absence de **from** DÉBUT et/ou **by** PAS, Maple les prend comme étant 1, par défaut.

3. Une boucle **while** possède la syntaxe suivante

```
while EXPRESSION BOOLÉENNE do BLOC INSTRUCTIONS end do;
```

Exercice 1

Pour obtenir la factorielle d'un nombre entier, on peut utiliser simplement le point d'exclamation. Une deuxième méthode consiste à appeler la fonction **factorial** de Maple.

UNE TROISIÈME MÉTHODE : calculer 17 ! sans effectuer de calcul direct et sans utiliser de fonctions prédéfinies Maple. Faire usage d'une boucle **for**.

Exercice 2

Calculer à l'aide d'un programme interactif, la moyenne arithmétique des 17 premiers nombres premiers. Pour cela, utiliser une fonction Maple et une seule, celle qui permet d'obtenir les nombres premiers. Afficher le résultat en format décimal.

Exercice 3

Afficher avec trois chiffres significatifs, les éléments de la subdivision de l'intervalle [–3, 3], de pas 3/10. Utiliser une boucle **for**.

Exercice 4

On se propose dans cet exercice, d'afficher à l'écran, la liste des 20 premiers entiers naturels divisibles par 6 ou par 7.

I. ON DEVRA IMPÉRATIVEMENT SUIVRE LES ÉTAPES SUIVANTES.

1- Déterminer la séquence des 20 premiers entiers divisibles par 3. Utiliser pour cela une boucle **for** avec l'option **while**. La séquence construite doit être initialisée à vide.

2- Transformer la séquence précédente en une liste.

3- Déduire de la question précédente la liste des 20 premiers entiers naturels divisibles par 6.

4- Déterminer comme précédemment, la liste des 20 premiers entiers naturels divisibles par 7.

5- Fusionner en une seule, les listes obtenues en 3° et en 4°.

6- Ordonner la liste obtenue en 5°. En déduire la liste demandée. N'oublier pas de supprimer les opérandes répétés.

II. RETROUVER DIRECTEMENT LA LISTE DEMANDÉE.

Exercice 5

L'objet de cet exercice est de calculer la somme des logarithmes népériens d'entiers naturels positifs pris dans une liste d'entiers relatifs, donnée.

Exemple : dans le cas où L := [0, –1, 4, –3, 2, 0, 7, –1], calculer ln(4) + ln(2) + ln(7).

Suivre les instructions suivantes.

1. Utiliser une boucle **`for`** … **`in`** … pour un calcul itératif de la somme des logarithmes.
2. Tester la 'positivité' de chaque opérande k, de la liste, et en cas de réponse négative, passer à l'itération suivante. Ce passage est réalisé par le mot clé **`next`**.

Exercice 6

Le but de cet exercice est de calculer la moyenne d'une unité d'enseignement E, constituée de trois matières M1, M2 et M3, de même coefficient. Suivre les instructions suivantes.

1. Les notes des matières sont saisies dans une liste L. Introduire des notes correctes comprises entre 0 et 20, quitte à placer des balises par la suite.
2. Utiliser une boucle **`for`** … **`in`** …
3. Au cas où une note traitée est inférieure à 5, sortir de la boucle en affichant le message d'erreur suivant : Vous n'avez pas de chance ; votre unité E n'et pas compensable.

La sortie de la boucle est réalisée par le mot clé **`break`** et l'affichage du message d'erreur est programmé comme suit :

```
ERROR(`chaine de caractères`);
```

Exercice 7

Déterminer **le plus petit** entier naturel k, tel que la somme des carrés des entiers naturels de 1 à k dépasse 70 000.

Exercice 8

Soit à calculer la somme des 50 000 premiers entiers naturels, et ce, de trois façons différentes, puis à chronométrer le temps écoulé dans chaque opération.

MÉTHODE 1 : à l'aide de la fonction **`sum`**.

MÉTHODE 2 : à l'aide de la fonction **`add`**.

MÉTHODE 3 : à l'aide d'une boucle **`for`**.

`Pour` chronométrer le temps écoulé et l'afficher en secondes, placer au début du programme,

```
restart : d := time() :
```

puis à la fin,

```
(time()-d)*'s' ;
```

Exercice 9

Rédiger un programme interactif permettant de résoudre une équation trinômiale du second degré et de retourner le message : pas de solution, une ou deux solutions avec les valeurs éventuelles des solutions.

Exercice 10 Appolonius de Perge est un mathématicien et astronome grec qui a vécu vers la fin du 3ème et début du 2ème siècle avant J.C. Il est auteur d'un célèbre ouvrage sur les coniques. On lui attribue un problème de construction de cercles tangents à trois cercles donnés (appelés cercles d'Appolonius.)

Maple dispose d'une fonction nommée **Appolonius** faisant partie de la bibliothèque **geometry** permettant de construire ces cercles.

- Commencer par charger en mémoire, la bibliothèque **geometry**. Construire ensuite trois cercles plans C1, C2 et C3. Utiliser pour cela la fonction **circle** du package geometry.
- Appeler ensuite la fonction Appolonius, avec les trois cercles en arguments, assigner le résultat à un identificateur de variable et faire suivre d'un **draw**, comme ceci

```
> App := Appolonius(C1,C2,C3) :
  for k do draw([C1,C2,C3,App[k]],color=[blue,blue,blue,red]) end do;
```

Noter bien que Maple retourne les différents cercles d'Appolonius sous forme d'une liste.

VOICI UN PROGRAMME INTERACTIF MAPLE PERMETTANT DE REPRODUIRE LES CERCLES D'APPOLONIUS.

```
> with(geometry):
  circle(C1,[point(P1,6,3),3],[x,y]):
  circle(C2,[point(P2,-3,0),2],[x,y]):
  circle(C3,[point(P3,0,7),1.5],[x,y]):
  App:=Appolonius(C1,C2,C3):
  for k do draw([C1,C2,C3,App[k]],color=[red,red,red,green]) end do;
```