TD Info n°3

PCSI 2 Lycée Pasteur

20 septembre 2007

Procédures

Comme vous avez sûrement pu le constater en faisant les exercices de la feuille de TD précédente, écrire des suites d'instruction un peu complexes sans avoir une structure adaptée va vite devenir pénible. Il serait beaucoup plus agréable de pouvoir décomposer le travail, notamment en définissant de nouvelles fonctions qui font une tâche dont on aura besoin plusieurs fois par la suite. Les procédures sont là pour ça. La syntaxe pour définir une procédure en Maple est la suivante :

```
[> nom := procédure (paramètre1::type1, ..., paramètrez::typez)
global variable1 ... variablek ;
local variable1' ... variablen' ;
description "ce que fait la procédure" ;
instructions
end proc ;
```

La première ligne ressemble à une affectation. En fait, c'en est une, on est en train de définir une variable de type procédure, que vous pourrez ensuite appeler (par son nom) quand bon vous semble, comme n'importe quelle fonction Maple prédéfinie. Les paramètres indiqués sur la première ligne sont les données dont la procédure a besoin pour tourner (il peut très bien ne pas y en avoir du tout). Vous pouvez préciser leur type à l'aide de du ::, mais ce n'est pas obligatoire (pour information, les types les plus fréquents sont integer, rational, float, boolean, string, et les types composés que nous étudierons plus tard). Si vous le faites, Maple renverra une erreur quand vous essaierez d'appliquer la procédure à des valeurs n'ayant pas le bon type. Les paramètres ne peuvent pas être modifiés en cours d'éxecutions de la procédure. Si vous avez besoin de modifier un paramètre, affectez d'abord sa valeur à une variable (locale ou globale) de la procédure.

Les deux lignes suivantes définissent les variables utilisées par la procédure. Il est nécessaire de définir ces variables (hormis les paramètres) avant de les utiliser dans la procédure. Quelle est la différence entre variables globales et locales ? Une variable locale n'est définie qu'à l'intérieur de la procédure, et perdra les valeurs que vous aurez pu lui affecter pendant son exécution dès que la procédure est terminée. Vous pouvez même donner à une variable locale le même nom qu'à une variable externe à la procédure, Maple saura faire la différence entre les deux (mais ce n'est pas conseillé !). Au contraire, une variable globale a une existence en-dehors de la procédure. Si vous l'utilisez dans votre procédure, Maple ira chercher la valeur qui lui était affectée avant le début de la procédure, et les éventuelles nouvelles affectations de cette variable auront toujours effet après l'exécution de la procédure.

La ligne de description est totalement facultative mais pas forcément inutile quand on commence à programmer des choses complexes, et surtout quand on doit lire des programmes écrits par quelqu'un d'autre. Enfin les instructions constituent le corps du programme. N'oubliez pas le end proc, indispensable. Une procédure renvoie par défaut la valeur du dernier calcul effectué mais vous pouvez toujours imposer une sortie via l'instruction RETURN(), appliquée à une chaine de caractères ou à une variable (dans ce cas-là c'est sa valeur qui sera affichée).

Dernière remarque, Maple sait gérer les programmes récursifs, c'est-à-dire que vous pouvez très bien faire appel à une procédure à l'intérieur de cette procédure elle-même. très pratique pour programmer tout ce qui est récursif, justement.

Exemples

Une petite procédure qui calcule le maximum de trois nombres a, b et c :

```
[> maxi3 := proc (a,b,c)
local d;
if a > b then d := a else d :=b :
if d > c then d else c
end;
```

Si on demande ensuite [> maxi3(2, 7, 5);, Maple va nous renvoyer 7.
Un deuxième exemple où on a une procédure récursive, pour calculer la factorielle :

```
[> facto := proc (n : :integer)
if n=0 then 1 else n*(facto(n-1))
end;
```

Attention aux manipulations de procédures récursives, il faut bien sûr s'assurer que la programme va s'arrêter un jour (ici, ne pas oublier de préciser ce qu'il se passe si $n = 0$). Maple dispose toutefois d'une procédure d'arrêt en cas de trop grand nombre d'appels à une même procédure.

Exercices

1. Reprendre le dernier exercice du TD2 en utilisant une procédure.
2. Écrire une procédure qui prend en argument une fonction et qui trace sa dérivée.
3. Écrire une procédure prenant pour paramètres trois réels a, b et c et calculant les solutions éventuelles de l'équation $ax^2 + bx + c = 0$ (essayez d'être le plus complets possibles).
4. On définit la suite de Fibonacci par $u_0 = u_1 = 1$ et $\forall n \in \mathbb{N}$, $u_{n+2} = u_{n+1} + u_n$. Écrire une procédure qui calcule le n-ème terme de la suite de Fibonacci.