Fonctions et procédures

Description d'une fonction :
nomFonction :=**proc(***arguments : :type***)**
local *i, ..., k*;
global *l, ..., p*;

instruction 1;
...
instruction n;

RETURN(*var***)**;
end;

Les arguments sont passés sous forme de liste, éventuellement suivis de leur type (par exemple i : :integer pour une variable i entière).
Les variables locales sont celles qui seront utilisées au sein de la fonction, et qui n'auront d'existence qu'à l'intérieur de la fonction.
Les variables globales sont celles qui existent hors de la fonction, mais que l'on souhaite pouvoir utiliser à l'intérieur de la fonction.
Une déclaration de fonction débute avec un **proc** et termine avec **end**. La fonction sera appelée avec des valeurs séparées par des virgules, en utilisant le nom de fonction auquel on l'a affectée, suivi des arguments entre parenthèses (par exemple nomFonction(argument1, argument2)).
Un argument en paramètre ne peut pas être affecté dans le corps de la fonction.
La fonction retourne la valeur dans la variable du **return**, ou s'il n'y en a pas la dernière valeur déclarée.

Exercice 1.

Ecrire une fonction prenant en argument une liste et une variable quelconque, et retournant le nombre d'occurences de l'élément dans la liste.

Modifier cette fonction pour qu'elle renvoie la liste des indices de l'élément dans la liste. (Par exemple, si l'élément se trouve en 2^{eme}, 4^{eme} et 5^{eme} position, la fonction doit renvoyer la liste [2,4,5]).

Une fonction peut s'avérer insuffisante ou inadaptée à l'usage que l'on veut en faire. Par exemple, elle ne permet pas de renvoyer directement plusieurs valeurs.
Il est alors possible d'écrire une procédure. Une procédure aura la même syntaxe qu'une fonction, hormis pour les paramètre, qui devront être déclarés du type *name*, par exemple maProcédure := proc(i : :name, j : :name).
La principale différence avec une fonction sera qu'il est possible de modifier les valeurs des variables passées en paramètre. On appelle cela un passage par adresse, par opposition aux fonctions qui ont un passage par valeur. On l'appellera en passant les arguments de type name entre apostrophes.

Exercice 2.

Ecrire une procédure swap qui prend deux variables quelconques en argument et échange leurs valeurs.

Exercice 3.

Ecrire une procédure mySort(L) qui trie L.

Exercice 4.

Parfois, il est utile de renvoyer une valeur ET de passer par adresse un ou plusieurs arguments, par exemple la fonction member. On appellera alors ces fonctions des fonctions procédurales.

Ecrire une fonction procédurale myMember ayant le même fonctionnement que la fonction member déjà existante.

Problème. *Extrait de l'examen de septembre 2000*

On relève certains jours de l'année le niveau des précipitations. Une observation sera une liste de 3 valeurs $[< numero\ du\ jour >, < numero\ du\ mois >, < niveau >]$. Par exemple, 10mm d'eau observés le 13 février correspond à la liste [13,2,10]. On suppose qu'une variable **Observations** contient une liste de telles observations. C'est donc une liste de listes.

- On définit la variable Mois :=[31, 28, 31, 30, 31, 10, 31, 31, 30, 31, 30, 31] correspondant aux nombres de jours des mois d'une année non bissextile.
  Définir une fonction numjour qui, à une observation donnée, associe le numéro du jour dans l'année. Par exemple, numjour([12,2,10]) vaut 43 (31 jours de janvier + 12 jours de février).
- Définir une fonction à valeur booléenne correct(x) qui retourne true si l'observation x correspond à un mois compris entre 1 et 12, un jour entre 1 et le nombre de jours du mois donné et un niveau de précipitations positif ou nul.
- Donner une ou des instructions permettant de supprimer les observations incorrectes (si elles existent) de la variable observations.
- Donner une fonction à valeur booléenne Avant(x, y) prenant la valeur true si l'observation x a lieu avant y.
- Donner une ou des instructions Maple pour que la variable Observations soit triée chronologiquement.