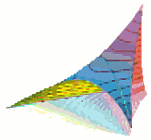

MI 101 Maple
Programmation 2
Feuille de TD numéro 5 v. 2- jeudi 08 novembre

Table des matières

1 Infos diverses **1**

2 Procédures **1**

3 Exercices **2**

4 Corrections **4**

1 Infos diverses

Ce TD clôt les fonctionnalités Maple en terme de structures de programmation. On verra donc les procédures, après avoir repris la partie 'boucles' de la séance précédente.

2 Procédures

Les procédures permettent de regrouper plusieurs instructions, en donnant un nom à ces instructions regroupées, puis de faire effectuer celles-ci à plusieurs reprises, en opérant sur des variables qui peuvent prendre des valeurs différentes à chaque appel. On dispose ainsi d'une commande polyvalente, construite selon les besoins spécifiques de ce qu'on souhaite faire avec Maple, et qu'on peut appliquer à répétition... Une procédure est un nouveau type d'objet Maple, qui est stocké dans une variable, et qui a donc un nom. Celle-ci peut être effacée par `unassign()`, et toutes sont effacées lors d'un `restart`.

La syntaxe-type d'une déclaration de procédure est:

```
nom:=proc(variables)
local varlocales;
global varglobales;
options options;
instructions
end;
```

- *nom* est le nom de la procédure qu'on définit, et qui servira à la rappeler ensuite.
- *variables* est la séquence des variables (indiquées par leurs noms séparés par des virgules) utilisées dans la procédure qui seront précisées à chaque appel; le nom donné à celles-ci n'a aucun lien avec les variables éventuellement définies à l'extérieur de la procédure; le champ *variables* peut être vide, et la procédure ne nécessite alors pas de préciser de variables lors de son appel.
- les *instructions* incluses dans la procédure peuvent être quelconques; on peut y inclure des boucles, des tests, appeler d'autres procédures déjà définies,...

Une procédure exécute des *instructions* pour certaines valeurs de ses *variables*, ou paramètres, et fournit en retour un résultat. Ce dernier est affiché lorsqu'on exécute la procédure (suivie d'un ;), et peut également être stocké dans une variable, par exemple par `X:=maprocedure(1,2,3)`. Maple considère comme résultat de la procédure le dernier objet qui

est manipulé lors de l'exécution de celle-ci. Il peut être utile, pour s'assurer que le résultat est bien celui que l'on souhaite, de terminer les *instructions* par une ligne d'évaluation du résultat souhaité. Si on souhaite afficher d'autres données lors de l'éxécution de la procédure, il faut utiliser `print()`.

Les *instructions* de la procédure peuvent agir sur des variables de trois types : les variables que l'utilisateur précise à chaque appel de la procédure (indiquées dans le champ *variables*), les variables qui sont définies dans la procédure et qui ne sont utilisées qu'à l'intérieur de celle-ci (des sortes d'outils internes, comme pour un compteur, ou pour stocker des résultats transitoires), et les variables qui sont définies à l'extérieur de la procédure et qu'on ne précise pas à chaque appel (en général des paramètres qui changent rarement, mais qu'on veut pouvoir modifier sans modifier la définition de la procédure). Ces deux derniers types de variables sont indiquées ainsi :

- *varlocales* est la séquence des variables qui seront définies et utilisées dans la procédure et qui n'ont n'ont de portée qu'à l'intérieur de celle-ci.
- *varglobales* indique (toujours sous forme de séquence, donc séparées par des virgules) les noms des variables extérieures à la procédure, donc qu'on ne précise pas à chaque appel.

Enfin, on peut préciser quelques détails lors de la déclaration de la procédure:

- *options* indique éventuellement à Maple certains modes de fonctionnement particuliers, comme `remember` qui garde en mémoire tous les résultats partiels obtenus, afin de gagner du temps.

Chacune de ces parties de la déclaration d'un procédure est facultative, car on n'a pas toujours besoin de variables locales, globales, ou d'options. On peut donc construire une procédure simpliste, qui a pour effet d'afficher 'Bonjour!!', par les instructions

```
bon:=proc()
print('Bonjour!!');
end;
```

, qu'on valide comme toute instruction Maple (le logiciel la reformule parfois, et la garde ensuite en mémoire), puis qu'on éxécute par `bon();`. Pour une procédure destinée à afficher autant de fois qu'on veut ce 'Bonjour!!', il sufit de construire la procédure suivante, semblable :

```
bonbon:=proc(k)
local i; for i from 1 to k do print('Bonjour!!'); od;
end;
```

qui, lorqu'on exécute `bonbon(25);`, affiche 25 fois 'Bonjour!!'. Enfin, une procédure prenant pour argument une liste ou un ensemble et dont le résultat est la somme de ses éléments se définit

```
somme:=proc(L)
local i,s;
s:=0;
for i from 1 to nops(L) do
s:=s+op(i,L);
od;
s;
end;
```

et est appelée, par exemple, par `somme([1,1,2,3,5,8,13,21,34]);`.

3 Exercices

Exo 1
Définissez une procédure qui prend en argument un ensemble de nombres réels et affiche son nombre d'éléments, le plus grand et le plus petit d'entre eux.

Exo 2
Définissez une procédure qui prend en argument une liste et renvoie une liste contenant la même séquence, en ordre inverse.

Exo 3
PROCÉDURE RÉCURSIVE: *En remarquant que* $n! = n \times (n-1)!$ *si* $n > 0$ *et* $n! = 1$ *sinon, définissez une procédure qui calcule la factorielle de* n *en faisant appel à la factorielle de* $n - 1$.

Exo 4
Définissez une procédure qui prend en argument une liste et renvoie la moyenne des valeurs contenues dans la liste.

Exo 5
Définissez une procédure qui renvoie la liste des valeurs de $\sin(2i\Pi/k)$, *pour* i *entre* 0 *et* k, k *étant un paramètre de la procédure.*

Exo 6
En utilisant la procédure précédente, définissez une procédure, prenant les arguments k, $n0$ *et* n *qui renvoie la liste des valeurs de* $\sin(2i\Pi/k)$, *pour* i *entre* $n0$ *et* $n0 + n$. *Il suffit d'extraire une liste de celle construite par la procédure précédente (voir* `op()` *ou* `seq()`*).*

Exo 7
MOYENNE MOBILE : *En utilisant les procédures précédentes, définissez une procédure qui renvoie chacune des moyennes des* $\sin(2i\Pi/k)$, *pour* i *entre* $n0$ *et* $n0 + n$, *en les stockant dans une liste;* k *et* n *sont des paramètres de la procédure, et* $n0$ *va de* 1 *à* $k - n$.

4 Corrections

Exo 1

```
info:=proc(S)
nops(S),max(op(S)),min(op(S));
end;
```

Exo 2

```
revlist:=proc(L)
local l,i; l:=[];
for i from 1 to nops(L) do l:=[op(i,L),op(l)]; od;
l;
end;
```

Exo 3

```
fact:=proc(b)
option remember;
if b=0 then 1 else b*fact(b-1); fi;
end;
```

Exo 4

```
moy:=proc(L)
local S,i; S:=0;
for i from 1 to nops(L) do S:=S+op(i,L); od;
S/nops(L);
end;
```

Exo 5

```
val:=proc(k)
local L;
L:=[seq(sin(2*i*Pi/k),i=0..k)];
L;
end;
```

ou plus condensé...

```
val:=proc(k) [seq(sin(2*i*Pi/k),i=0..k)]; end;
```

Exo 6

```
extval:=proc(k,n0,n)
local L1,L2; L1:=val(k);
L2:=[seq(op(i,L1),i=n0..n0+n)];
L2;
end;
```

ou plus condensé...

```
extval:=proc(k,n0,n) [seq(op(i+n0,val(k)),i=0..n)]; end;
```

Exo 7

```
moymob:=proc(k,n)
local L1,L,i; L:=[];
for i from 1 to k-n do L:=[op(L),moy(extval(k,i,n))]; od;
L;
end;
```

ou plus condensé...

```
moymob:=proc(k,n) [seq(moy(extval(k,i,n)),i=1..k-n)]; end;
```