Maple – Fiche de Révisions

1- Variables

Une variable a un **nom** et un **type**, elle peut contenir des **données** qui correspondent à son type.
Types usuels en maple : integer, list, listlist, string, float, boolean
En pseudo-code, n'oubliez pas qu'il faut déclarer les variables au début !

2- Boucles

Plusieurs types, les plus usuelles sont for et while.
On utilise une boucle while quand on ne peut pas savoir à l'avance combien d'itération la boucle va avoir, on qu'on veut supprimer des éléments dans une liste. Pour les autres cas, une boucle pour suffit.
Pour une boucle While : ne pas oublier 1- d'initialiser le compteur 2- d'incrémenter le compteur dans la boucle.

3- Fonctions et procédures

Une fonction retourne un **résultat** via l'opérateur RETURN, elle n'a que des paramètres d'entrée.
Une procédure ne retourne **aucun** résultat, mais peut modifier la valeur de certaines variables, on parle alors de paramètres d'entrée/sortie.

Quand on conçoit une fonction, il faut se demander de quels **ingrédients** la fonction a besoin pour retourner son résultat (par exemple, pour qu'une fonction retourne le maximum dans un liste, elle doit connaître la liste en question)
Même principe pour un procédure, sauf qu'il faut aussi se demander quelles variables la procédure va être amenée à modifier, ces paramètres doivent avoir le type name.

Pièges à éviter :

- l'oubli des eval dans une procédure (cf cours)
- ne pas être cohérent entre la **déclaration** des fonctions/procédure et leur **appel**.
  Ex : calories:=proc(LL::listlist, a:: string)
  …
  end :
  a:=calories(« Hamburger »)
  ici, la fonction déclarée prend en entrée une liste de liste et une chaine de caractères, mais on ne l'appelle qu'avec une chaine de caractères.

4- Codes usuels

max dans une liste

```
max:=L[1]:
for k from 2 to nops(L) do
if max<L[k] then
max:=L[k]:
fi:
od:
```

sommer tous les éléments d'une liste

```
c:=0:
for k from 1 to nops(L) do
c:=c+L[k]:
od:
```

booléen qui indique s'il existe un élément vérifiant une certaine propriété dans une liste

```
b:=false
for k from 1 to nops(L) do
if < L[k] vérifie la propriété > then
b:=true:
fi:
od:
end:
```

supprimer d'une liste les éléments qui vérifie une certaine propriété

```
k:=1
while k <=nops(L) do
if < L[k] vérifie la propriété > then
L:=subsop(k=NULL,L):
else
k:=k+1:
fi:
od:
end:
```