Maple TD3 **Novembre 2004**

La plupart des langages de programmation ont des structures communes comme les procédures, les tests ou les boucles. Une fois que vous aurez compris comment ces structures fonctionnent, il vous suffira d'apprendre la syntaxe spécifique à chaque langage pour pouvoir écrire un programme.
Ceux qui savent déjà programmer vont donc trouver ce TD plutôt facile. Pour eux, nous avons prévu des exercices plus difficiles, à faire en seconde lecture, dont la difficulté est signalée par le symbole (*).

1 Procédure

Une procédure est un petit programme, qui a un nom, des entrées, une sortie et des variables qui lui sont propres (variables locales). Tapez l'exemple ci-dessous, modifiez-le pour mieux comprendre comment il fonctionne.

```
[>double := proc(x)
   RETURN(2*x);
   end;
[>double(1.3);double(z);
```

```
[>double := proc(x)
   local z;
   z :=x; RETURN(x+z);
   z :=3; end :
[>double(4);
[>x,z;
[>x :=4 : z :=12 :
[>x,z;
[>double(17);
[>x,z;
```

On peut utiliser des variables à l'intérieur d'une procédure, mais ces variables sont **locales**.

```
[>je_m_appelle :=proc(nom) printf('bonjour %s ! ', nom) : end :
[>je_m_appelle(Sophie) :
```

' se fait en tapant AltGr + 7

Exercice 1 *Écrire une procédure qui prend en argument une fonction et qui trace sa dérivée.*

2 Tests

```
[>minimum :=proc(x,y) if x<y then RETURN(x) else RETURN(y) fi end;
```

Exercice 2 *Écrire la procédure* `Premier(x)` *qui dit si* x *est premier ou non (*`isprime`*).*

Exercice 3 (facultatif) *Écrire une procédure qui prend en argument un polynôme du second degré et renvoie ses racines.*

Exercice 4 (facultatif) *Écrire une procédure qui renvoie pile ou face avec une probabilité* $1/2$ *(*`rand`*).*

Problème I : Une fermière possède un panier d'oeufs, et sait que l'un d'entre eux est pourri (son poids n'est pas le même).
Écrire une procédure disant de quel oeuf il s'agit, en pesant tour à tour les différents oeufs. Cette procédure prendra en entrée une fonction, oeuf, définie par exemple comme suit :

```
[>oeuf := n->piecewise(n<>4,120,n=4,110);
```

et donnera en sortie le numéro de l'oeuf et le nombre de pesées réalisées.

* Il y a 8 oeufs, l'oeuf pourri pèse plus lourd, à faire en au plus 3 pesées.

** Il y a 8 oeufs, on ne sait pas si l'oeuf pourri est plus lourd ou plus léger, à faire en 3 pesées.

*** Il y a 12 oeufs, on ne sait pas si l'oeuf pourri est plus lourd ou plus léger, à faire en 3 pesées.

3 Boucles

```
[>N :=10 : f :=1 : for i from 1 to N do f :=i*f od;
[>N!;f;
[>N :=10 : i :=1; f :=1 : while i<N do i :=i+1 : f :=f*i : od;
```

Exercice 5 *Afficher les* 20 *premiers nombres premiers (rappel :* 1 *n'est pas un nombre premier ...)*

Exercice 6 *Réécrire la procédure* `Premier(x)` *sans utiliser* `isprime`.

Exercice 7 (facultatif) *Calculer de deux façons différentes la somme des* 20 *premiers nombres pairs. (utiliser* `sum` *et* `for`*)*

Exercice 8 (facultatif) *Pour* n *variant de* 0 *à* 39, *vérifier que* n^2+n+41 *est un nombre premier.*

Problème II : Un palindrome est un mot qui se lit de la même façon à l'endroit et à l'envers (kayak, "Esope reste et se repose", abracaddacarba sont des palindromes).
* Écrire un programme qui détecte si un mot est un palindrome (`length`, `substring`).

4 Récursivité

Les procédures de MAPLE peuvent être **récursives**, c'est à dire s'appeler elles-même... Une fois compris l'exemple ci-dessous, vous pourrez utiliser la récursivité, mais méfiez-vous, c'est une source d'erreurs importante.

```
[>factorielle :=proc(n) if n=0 then RETURN(1) else RETURN(n*factorielle(n-1))
fi; end;
[>factorielle(0);factorielle(10);
[>factorielle(-1);
```

Exercice 9 *Définir la* suite de Fibonacci : $\begin{cases} u_0 = u_1 = 1 \\ u_{n+2} = u_{n+1} + u_n \quad \forall n \geq 0 \end{cases}$.

Exercice 10 *Écrire une procédure qui définit récursivement le binôme de Pascal à partir de la formule :* $\begin{pmatrix} n \\ p \end{pmatrix} = \begin{pmatrix} n-1 \\ p-1 \end{pmatrix} + \begin{pmatrix} n-1 \\ p \end{pmatrix}$. *(faire attention à l'initialisation).*

Problème III : Petit jeu de devinette : comment trouver un nombre tiré au hasard entre 1 et n à partir des questions "Plus grand ou plus petit ?".

* Écrire un programme qui devine un nombre x compris entre 1 et n.

** Ajouter un compteur pour savoir en combien de coups la solution a été trouvée.