TP Maple 3 : Fonctions et procédures.
Enonc

ISEN N1

March 11, 2007

Ci - après, on donne une procédure calculant un nombre x à la puissance n :

```
> ma_procedure := proc(x, n)
    local m, resul ;
      resul := 1 ;
      for m to n do resul := resul*x od ;
      resul
    end ;
```

La valeur retournée par une procédure est la dernière expression évaluée.

Ici, une fois ce code entré, ma_procedure(2, 3) retournera 8, ma_procedure(3, 4) retournera 81, etc.

On peut définir des variables locales précédées par le mot réservé local comme ci - dessus. L'emploi de variables locales est vivement conseillé. Dès que la procédure est terminée, ces variables redeviennent indéfinies. Il faut utiliser au maximum ce procédé qui évite des interactions toujours dangereuses avec les variables de l'environnement global. De plus, seules les variables locales peuvent être modifiées dans une procédure.

La procédure donnée en exemple ci - dessus est très lente pour élever un nombre à une grande puissance : la fonction time donnant le temps de travail en seconde depuis le début de la feuille de travail, essayer :

```
> depart := time() :
> ma_procedure(12345.0, 123456) ;
> arrivee := time(): temps:=arrivee - depart ;
> depart := time() :
> 12345.0 ^ 123456 ;
> arrivee := time(): temps:=arrivee - depart ;
```

On voit que Maple utilise un processus d'exponentiation rapide.

On va développer maintenant une procédure d'exponentiation rapide (est - ce la même que celle de Maple ?) :

On utilisera les développements en puissances de 2 :

par exemple x^{37} se décompose en $x\, x^4\, x^{32} = x\, (x^2)^2\, ((((x^2)^2)^2)^2)^2$ qui réduit le nombre de multiplications de 36 à 7.

En fait $37 = 2^0 + 2^2 + 2^5$. Il faut donc commencer par développer 37 en puissances de 2 par convert(37, base, 2).

Écrire une procédure qui utilise ce qui précède ; tester son exactitude et sa rapidité.

Une procédure possible :

```
> expo := proc(xx, nn)
    local liste, element, resul, puissance ;
      liste := convert(nn, base, 2) ;
      puissance := xx ;
      if liste[1] = 1 then resul := xx else resul := 1 fi ;
      for element in subsop(1 = NULL, liste) do
          puissance := puissance ^ 2 ;
          if element = 1 then resul := resul * puissance fi
      od
    end ;
```