Introduction à la programmation (suite)
TD du 20 octobre 2005

1 Règles de syntaxe

1.1 Rappel: *if* et *for*

`if` *conditionbooleenne* `then` *sequenceinstruction*(s)
(`elif` *conditionbooleenne* `then` *sequenceinstruction*(s))
(`else` *sequenceinstruction*(s))
`end if;`

`for` *nom* `from` *expr* `to` *expr* `by` *expr*
`do`
sequenceinstruction(s)
`end do;`

OU

`for` *nom* `in` *expr* (liste, séquence, ensemble...)
`do`
sequenceinstruction(s)
`end do;`

Remarque: on peut remplacer `end if` et `end do` par `fi` et `od`...

1.2 *while*

C'est l'instruction: *tant que...*

`while` *conditionbooleenne*
`do`
sequenceinstruction(s)
`end do;`

1.3 Indentation

On n'est pas obligé de passer à la ligne lorsqu'on écrit ces lignes de commandes mais on le fera, tout comme on s'efforcera de respecter les règles d'*indentation*, c'est-à-dire les espacements au début des lignes du programme qui permettent une compréhension rapide de celui-ci. Bien indenter permet de voir comment les diffrentes boucles du programme sont imbriquées.

Exemple:

```
L[1]:=[]: L[2]:=[]: L[3]:=[]: L[4]:=[]: #initialisation des variables
for i to 1000
do
    if i<=500 then
        if isprime(i) then L[1]:=[op(L[1]),i]
        end if:
        if type(i/3,integer) then L[3]:=[op(L[3]),i]
        end if:
    else
        if isprime(i) then L[2]:=[op(L[2]),i]
        end if:
        if type(i/2,integer) then L[4]:=[op(L[4]),i]
        end if:
    end if:
end do:
L[1];L[2];L[3];L[4];
# 4 listes: les premiers<=500, ceux>500, les multiples de 3<=500
# et les pairs>500
```

2 Procédures

La procédure est le format classique d'un programme Maple. Sa syntaxe est la suivante:

nom:=`proc(`*arguments*`)`
(`local` *sequence de variables locales* ;)
(`global` *sequence de variables globales* ;)
Programme
`end;`

- Les variables locales ne sont reconnues qu'à l'intérieur de la procédure alors que les variables globales sont accessibles en dehors de la procédure. Exemple:

  ```
  messages:=proc()
    global message1;
    local message2;
      message1:="Hello world!";
      message2:="Bonjour monde!";
  end;
  ```

  Puis essayez: `messages(); message1,message2;`
  Les lignes `local` et `global` sont optionnelles mais bien les utiliser évite beaucoup de problèmes.

- On peut préciser à une procédure le type d'argument qu'elle peut recevoir: *nom_proc*:=proc(x::*type*1,y::*type*2)...
- Attention! Un paramètre formel passé à une procédure ne peut être modifié à l'intérieur de cette procédure.
  Exemple: diviser un nombre entier tant qu'il est pair

```
div:=proc(x::posint)
    while type(x,even) do x:=x/2 end do;
end;
```

  Ce programme n'est pas bon car on change la valeur de l'argument que l'on a donné à Maple. Il faudrait écrire:

```
div:=proc(x::posint)
  local y;
    y:=x;
    while type(y,even) do y:=y/2 end do;
    y;
end;
```

3 Exercices

Exercice 1.

Ecrire une procédure qui prend en entrée les coefficients d'un polynôme de degré 2 et renvoie ses racines.

Exercice 2.

Ecrire deux procédures qui donnent la somme des entiers de 0 à n pour n entier naturel (une itérative et une récursive).

Exercice 3.

Ecrire une procédure qui, étant donné un entier naturel ne comportant pas de 0 dans son écriture décimale, renverse les chiffres de ce nombre. Prévoir un message d'erreur au cas où il y ait un zéro dans le nombre.

Exercice 4.

Ecrire une procédure qui donne les n premiers nombres premiers par la méthode du crible d'Eratosthène.

Exercice 5.

La conjecture de Syracuse: On choisit un nombre entier naturel. Si il est pair, on le divise par 2; si il est impair, on le multiplie par 3 et on ajoute 1. On répète

cette opération. Essayez cet algorithme sur plusieurs nombres. Que constatez-vous? On appelle *temps de vol* d'un nombre le nombre de fois qu'il faut le faire entrer dans la boucle pour arriver à 1 et *altitude* du nombre la valeur maximale atteinte au cours de l'application de l'algorithme. Calculez les temps de vol et altitudes des nombres entiers inférieurs à 1000.

Exercice 6

Formule d'Archimède: calculer une approximation de π en approximant le cercle par des polygones.

Exercice 7

Ecrivez une procédure récursive qui transforme un segment donné en flocon de Von Koch d'ordre n.