Les structures conditionnelles

1) if ... then ... end if

a) L'exécution d'une commande peut être soumise à une **condition**, exprimée par une expression booléenne.
> **if** 3>2 **then** "hello" **end if**;
L'instruction est exécutée, la condition s'évaluant à **true.**
> **if** 3<2 **then** "hello" **end if**;
L'instruction n'est pas exécutée car la condition est **false**.

ATTENTION L'expression booléenne doit être *évaluable* :
> **if** Pi>3 **then** Pi **fi**; # erreur : on doit comparer des "*numeric*"
> **if** evalf(Pi)>3 **then** evalf(Pi) **fi**; # écriture correcte...
(NB : pour comparer deux réels, il est naturel de comparer leurs développements décimaux…)

b) On peut conditionner simultanément *plusieurs instructions* :
> **if** 3>2 **then** "hello"; "bye,bye" **fi**;

c) *Instructions en cascade* :
> **if** *condition1* **then** *instruction1* **else** *instruction2* **end if;**
> **if** *condition1* **then** *instruction1* **else**
if *condition2* **then** *instruction2* **else** *instruction3* **end if**
end if;
La commande **elif** est une contraction de **else if**.
> **if** condition1 **then** instruction1 **elif** condition2 **then** instruction2 **else** instruction3 **fi**;
elif ne nécessite pas de balise « **end if** » supplémentaire. Autre avantage : elle n'augmente pas le niveau d'affichage (cf. ci-dessous). Il faut toujours préférer **elif** à **else if**.

d) *Niveau d'affichage*
Lorsque Maple procède à une évaluation, il attribue au résultat une valeur entière appelée « niveau d'affichage ». Le niveau habituel est 0. **Les instructions conditionnelles ou les boucles augmentent le niveau d'affichage de 1.** Si le niveau d'affichage dépasse l'entier contenu dans la variable *printlevel*, Maple n'affiche pas le résultat à l'écran. Par défaut, *printlevel* =1.
Lorsqu'on exécute un programme et que Maple « rend la main » sans rien afficher (en particulier, sans message d'erreur), c'est qu'il y a probablement un problème de niveau d'affichage. Il suffit souvent d'augmenter la valeur de *printlevel*.
> *printlevel* := 2;
NB : Il est inutile de tenir compte de ce problème pratique pour l'épreuve d'algorithmique de l'X.

2) Utilisation dans une procédure

a) *Evaluation de la condition*
```
> f:=x -> if x>2 then 3 else 5 end if:
> f(1),f(2),f(3);
```
5,5,3
L'évaluation d'une procédure est retardée, celle d'une expression est complète. Il faut en tenir compte pour que la condition présente dans l'instruction soit évaluable au moment où Maple en a besoin.

> plot(f(x),x); # erreur car MAPLE cherche à évaluer f(x), x n'étant pas connu.
> plot(f) ou
> plot('f(x)',x); # correct car l'évaluation de f ou de 'f(x)' est retardée.

b) *Terminaison d'une procédure*
roulette :=**proc**(n)
if n=0 **then** return(n) **end if**
if n>0 **then** error(`nombre positif`) **end if**
"hello"
end proc ;

3) Utilisation dans une boucle
a) *Condition de terminaison*

do *instruction1* ;
if *condition* **then** break **end if** ; # condition de terminaison de la boucle
instruction2 ;
od ;

La boucle est interrompue dès que la condition devient vraie (cf. fiche « boucles »)

b) Niveau d'affichage

> **for** k **to** 100 **do if** isprime(k) **then** k **end if end do**; # affichage des nombres premiers seulement
Attention : la boucle "for" a augmenté le niveau d'affichage !

4) Une autre structure conditionnelle : *piecewise*

La procédure *piecewise* permet de définir des **expressions** "par morceaux" :
> F:=piecewise(x<3,20,10);
F est une fonction (au sens de Maple) de x qui vaut 20 lorsque x<3, et 10 « par défaut » (ici, si x≥ 1).
> **whattype(F);**

function

On peut enchaîner les conditions :
> piecewise(condition1, valeur1,condition2,valeur2,valeur_par_défaut)
Dès qu'une condition est true, Maple donne à l'expression la valeur correspondante et arrête l'évaluation. Si elle n'est pas précisée, la valeur par défaut est 0 (zéro).

Évaluation en un point :
> F(4);# erreur : F n'est pas une procédure!!!!
> eval(F,x=4);# correct

NB : *piecewise* permet de retarder l'évaluation de la condition. Comparer :
> **piecewise(x>0,1,-1);**

$$\begin{cases} 1 & 0 < x \\ -1 & otherwise \end{cases}$$

> **if x>0 then 1 else -1 end if;**
Error, cannot evaluate boolean: -x < 0