TP2 : assignation/réassignation de variables et boucles (sous Maple12)

Cette seconde séance de TP (deux séances d'1h20) a essentiellement pour objectif de vous familiariser avec l'assignation, la réassignation (et *a fortiori* la libération) de variables sous le logiciel de calcul symbolique `Maple12`. Ensuite, on s'entrainera à la construction de codes (sous `Maple12`) impliquant les commmandes

```
> for i from d to f by p while [...] do
instruction1 ;
instruction2 ;
...
instructionN ;
end do:

> for s in S while [...] do
instruction1 ;
instruction2 ;
...
instructionN ;
end do:

> if [condition1] then
instruction1 ;
instruction2 ;
...
instructionN ;
elif [condition2] then
instruction1bis ;
instruction2bis ;
...
instructionNbis ;
else
instruction1ter;
instruction2ter ;
...
instructionNter ;
end if:
```

Ces commandes correspondent pour les deux premières à la réalisation de boucles `do`, tandis que la dernière correspond à la mise en œuvre d'un processus de décision (alternative ici à trois volets, le second volet suivant l'instruction `elif` pouvant

ête éliminé). Dans les boucles `do` les éléments de syntaxe `by ...` et `while [...]` peuvent, le cas échéant, être omis si leur présence n'est pas justifiée par l'algorithme. On notera que l'instruction `:` au lieu de `;` lors de l'instruction `end` évite les affichages intermédiaires au fur et à mesure de l'exécution du code.

La réalisation de codes sous `Maple12` sera toujours faite ici en se positionnant derrière le prompt et en suivant sous les onglets du bandeau le chemin

$$Insert \rightarrow Code\ Edit\ Region$$

L'exécution du code, une fois celui ci rédigé dans la fenêtre adéquate, se fait avec le clic droit.

EXERCICE 1 (réalisation d'un processus de décision). Ouvrez une feuille de travail vierge et titrez la `Boucles if`. En utilisant les instructions commandant une alternative :

```
if [...] then
instruction ;
else
instructionbis ;
end if ;
```

réalisez (puis validez) un code qui, étant données deux variables `a` et `b` auxquelles on assigne des valeurs rationnelles (par exemple `a:= 2/5; b:= 22/7;`), renvoie le message `a est plus grand que b` lorsque $a \geq b$ et le message `b est strictement plus grand que a` sinon. L'affichage d'un message (par exemple le mot `message`) sous `Maple12` se fait via l'instruction

```
> "message" ;
```

EXERCICE 2 (réalisation d'un processus de décision). Continuez avec la feuille de travail précédemment ouverte. Comme dans l'exercice 1, réalisez (puis validez) un code qui, étant donnés trois nombres `a`, `b`, `c` auxquels on assigne soit des valeurs rationnelles, soit des valeurs flottantes (faites un code dans les deux cas), pourvu que l'on assigne à `a` une valeur non nulle, renvoie :

- dans le cas où les valeurs assignées aux variables `a,b,c` sont rationnelles, l'expression algébrique des deux racines du trinôme aX^2+bX+c dans $\mathbb{C}$ (ou de l'unique racine double si $b^2-4ac=0$) ;
- dans le cas où les valeurs assignées aux variables `a,b,c` sont des réels flottants, des valeurs approchées pour les deux racines (éventuellement complexes) du trinôme aX^2+bX+c dans $\mathbb{C}$. Est-il raisonnable dans ce cas d'isoler le sous cas où il y a une racine double ($b^2-4ac=0$) ?

Sauvez votre feuille de travail en l'enregistrant sous votre répertoire `TPMaple12` comme `TP2-boucleIF`.

EXERCICE 3 (réalisation de boucles `do`). Ouvrez une feuille de travail vierge sous `Maple12`.

(1) Que se passe t-il lorsque l'on excécute le code

```
for i from 0 to 7 do
[i,i*i];
ifactor (i!);
‘-----‘;
```

```
end do:
```

Comparez avec ce qui se passe lorsque l'on exécute le code :

```
for i from 0 to 7 do
[i,i*i];
ifactor (i!);
‘-----‘;
end do;
```

(2) Réalisez (sous Maple12) un code permettant de calculer la somme

$$\sum_{k=1}^{N} k^2$$

lorsque N désigne un entier auquel on assigne ensuite une valeur. Testez ce code.

(3) Réalisez (toujours sous Maple12), en faisant intervenir dans la syntaxe les instructions `for ... to ... by ... do ... end do`, un code permettant de calculer la somme des entiers impairs compris entre 1 et un entier naturel $N \in \mathbb{N}^*$ auquel on assigne ensuite une valeur. Toujours sur le même principe, $N \in \mathbb{N}^*$ et $p \in \{1, ..., N\}$ désignant deux entiers naturels auxquels on assignera ensuite des valeurs, réalisez (et testez pour des valeurs de N et de p) un code permettant de calculer la somme

$$\sum_{\substack{1 \leq p \leq N \\ p \text{ divise k}}} k^3.$$

(4) Réalisez (toujours sous Maple12) un code permettant de calculer le plus petit entier N tel que la somme

$$\sum_{k=1}^{N} k^3$$

soit au moins égale à un seuil $M \in \mathbb{N}^*$ auquel on assigne une valeur donnée. Testez ce code avec $M = 10^{20}$ en demandant au terme de l'exécution l'affichage à la fois de cet entier $N = N(M)$ et de la valeur de la somme correspondante $\sum_1^{N(M)} k^3$.

Sauvez votre feuille de travail en l'enregistrant sous votre répertoire TPMaple12 comme TP2-boucleDO.

EXERCICE 4 (boucles `do ... end do` et `if ... else ... end if` enchainées et conjecture de Syracuse). Une célèbre conjecture, dite *Conjecture de Syracuse*, soulevée autour de 1930 par le mathématicien allemand Lothar Collatz (voir par exemple, pour une histoire de cette conjecture et l'« état de l'art » aujourd'hui, le site

http://fr.wikipedia.org/wiki/Conjecture_de_Syracuse)

stipule que, pour tout entier $N \in \mathbb{N}^*$, la suite générée par récurrence suivant

$$u(1) = N$$
$$u(k+1) = \begin{cases} \frac{u(k)}{2} & \text{si } u(k) \text{ est pair (soit } \texttt{type(u(k),even) = true}\text{)} \\ 3u(k)+1 & \text{si } u(k) \text{ est impair (soit } \texttt{type(u(k),odd) = true}\text{)} \end{cases}$$
$$\forall\, k \geq 1$$

finit par atteindre en un temps fini la valeur 1.

(1) Expliquez pourquoi, une fois cette valeur 1 atteinte pour $k = k(N)$, la suite va nécessairement répéter indéfiniment le motif

```
1  4  2  1  4  2  1  4  2  1  4  2  1  4  2  1 ...
```

(2) Ouvrez une nouvelle feuille de travail, que vous intitulerez (en mode Texte) `Conjecture de Syracuse`. En utilisant la syntaxe

```
u:= N :
for i from 1 to M while u > 1 do
if ...  then
...
else
...
end if :
...
end do ;
```

réalisez un code affichant les valeurs prises par la suite $(u(k))_{k\geq 1}$ (initiée à la valeur N) tant que $k \leq M$ et que $u(k)$ reste différent de 1, ainsi que la valeur du premier cran $1 \leq k \leq M$ (s'il existe) tel que $u(k) = 1$. Testez ce code avec les valeurs de $N = 127$, $N = 537$, $N = 1235$ en prenant (ici arbitrairement) $M = 500$ comme « marge de sécurité ». La conjecture de Syracuse est-elle bien validée pour ces valeurs de N ? Que vaut dans chacun de ces trois cas le cran $k \geq 1$ tel que $u(k)$ prenne la valeur 1 pour la première fois ? Peut on affirmer ce cran est de plus en plus grand au fur est à mesure que N augmente ?

(3) Modifiez le code que vous venez d'élaborer pour que soient affichées uniquement au terme de son exécution :
- le cran k (s'il existe entre 1 et M) où $u(k)$ prend la valeur 1 pour la première fois ;
- la valeur maximale prise par l'entier $u(k)$ avant d'atteindre cette valeur 1.

Sauvez votre feuille de travail en l'enregistrant sous votre répertoire `TPMaple12` comme `TP2-Syracuse`.

EXERCICE 5 (Algorithmes d'Euclide et d'Euclide étendu). Ouvrez une nouvelle feuille de travail sous `Maple12`.

(1) Apprenez à vous familiariser avec les commandes `iquo` et `irem` (fournissant quotient et reste dans la division euclidienne de deux entiers) et `quo` et `rem` (fournissant quotient et reste dans la division euclidienne de deux polynômes à coefficients entiers ou rationnels).

(2) Dans l'exercice 9 de la feuille 1 (question 3), vous avez observé la syntaxe du code `PGCD.m` (et du code `div.m` que ce code appelle) fournissant le calcul du PGCD de deux entiers sous `MATLAB`. On rappelle ces deux codes ici, tels qu'ils sont donnés dans le polycopié de cours :

```
function [q,r] = div (x,y) ;
q = floor (x./y) ;
r = x- q*y ;
```

```
function PGCD=PGCD(a,b);
x=a ;
y=b ;
while y>0
    [q,r] = div(x,y);
    if r==0
        PGCD = y;
        y = 0 ;
    else
        [q1,r1] = div(y,r);
        x = r;
        PGCD = x ;
        y=r1 ;
    end
end
```

En vous inspirant de ces modèles, élaborez et testez :

- un code sous `Maple12` (utilisant les instructions `iquo` et `irem`) permettant, au terme de son exécution, de calculer le PGCD de deux entiers naturels a et b, avec $b > 0$;
- un code sous `Maple12` (utilisant les instructions `quo` et `rem`) permettant, au terme de son exécution, de calculer le PGCD (dans $\mathbb{Q}[X]$) de deux polynômes A et B à coefficients rationnels, avec $B \neq 0$.

(3) Dans l'exercice 10 du TP1, vous avez observé la syntaxe du code `bezout.m` permettant de manière récursive (ce code s'auto-appelle) de calculer, étant donnés deux entiers relatifs a et b avec $b \neq 0$, une solution (u, v) de l'identité de Bézout :

$$\mathrm{PGCD}(|a|, |b|) = au + bv. \qquad (*)$$

On rappelle ici ce code, tel qu'il figure dans le polycopié de cours :

```
fonction [PGCD,u,v]=bezout(a,b);
x=a ;
y= abs(b) ;
[q,r]=div(x,y);
if r==0
    PGCD = y;
    u=0 ;
    v=1;
```

```
else
 [d,u1,v1]=bezout(y,r);
 PGCD=d;
 u=v1;
 v=sign(b)*(u1- q*v1);
end
```

En vous inspirant de ce modèle, élaborez et testez :

- un code sous `Maple12` (utilisant cette fois les instructions `iquo` et `irem`) permettant, au terme de son exécution, de calculer, étant donné un couple d'entiers relatifs (a, b) avec $b \neq 0$), le PGCD de $|a|$ et $|b|$ ainsi qu'un couple d'entiers (u, v) solution de l'identité de Bézout $(*)$;
- un code sous `Maple12` (utilisant les instructions `quo` et `rem`) permettant, au terme de son exécution, de calculer le PGCD (dans $\mathbb{Q}[X]$) de deux polynômes A et B à coefficients rationnels, avec $B \neq 0$, ainsi qu'un couple de polynômes U et V de $\mathbb{Q}[X]$ tels que
$$A(X)U(X) + B(X)V(X) = \mathrm{PGCD}(A, B)$$
(le PGCD étant ici considéré à un facteur multiplicatif $\lambda \in \mathbb{Q}^*$ près).

Sauvez votre travail dans votre dossier `TPMaple12` en l'enregistrant sous le nom `TP2-Euclide`.

Exercice 6 (familiarisation avec les commandes `eval` et `subs`). Soit l'expression algébrique en trois variables :
$$P(X, Y, Z) = 3XY + 5XYZ^2 - 2XY^3Z.$$

(1) En utilisant la commande `eval` (on en examinera avec `?eval` la syntaxe), déclarez sous `Maple12` la fonction de la variable t :
$$t \longmapsto P(t^\alpha, t^\beta, t^\gamma),$$
où α, β, γ sont trois nombres rationnels strictement positifs auxquels des valeurs seront ensuite assignées.

(2) Affichez le graphe de la fonction :
$$t \in [0, 1] \mapsto \Big[P(x, y, z)\, \cos(xy)\Big]_{x=t^{1/5}, y=t^{1/8}, z=t^{1/4}}.$$

(3) En utilisant la commande `subs` (on en examinera avec `?subs` la syntaxe), déclarez sous `Maple12` la fonction des variables (u, v, w) :
$$(u, v, w) \longmapsto P(u + v + v, uv + vw + wu, uvw).$$