▶ **MP1 – Maple**
Premier TD de maple
Lundi 30 septembre 2013

Rappels, DL, Courbes

1 Rappels

1.1 L'aide

Votre meilleur amie pour Maple. Placez vous sur un nom de fonction Maple et appuyez sur F2 ; ou lancez l'aide *via* Ctrl+F1 ou Help > Maple Help.

1.2 Principes de base

Instruction terminée par **;** (avec affichage) ou **:** (sans affichage), rappel de la dernière expression calculée avec %, commentaires avec #

```
2 + 2; # blabla qui ne sera pas évalué
3 * 4:
% ** 5;
```

Affectation *via* **:=**

```
x := 3;
x * 18;
```

Libération d'une variable *via* **' '**

```
x := 3;
x;
x := 'x';
x;
```

Comprendre les problèmes de variables est vraiment crucial quand on travaille avec Maple. Utiliser une variable non libre (*i.e.* qui a été assignée et pas libérée) dans une expression provoquera son évaluation immédiate.
Tester par exemple :

```
x := 3;
y := x + 2;
x := 'x';
y;
y := x + 2;
z := y ** 2;
```

Évaluation approchée *via* **evalf**

```
x := 99**98**97;
evalf(%);
```

Fonction *via* **->**

```
f := x -> x**3;
f(4);
g := y -> f(y+4);
g(0);
```

Fonction *via* **unapply**

```
p := 3*x**3 + 5*x**2 + 2*x + 8;
f := unapply(p, x);
```

Changement de variable *via* **subs**

```
subs(x = 3, x ** 2 + 3 * x + 4);
subs(x = ln(y ** 3), exp(x**(1/3)));
subs(cos(x) = u, cos(x)/(1 - cos(x)));
```

1.3 Structures de contrôle

Conditions avec **if** ... **then** ... **else** ... **fi**
Éventuellement avec des **elif**

```
if 42**12 < 17**28 then 17 else 12 fi;
if 42**12 < 17**28
then 17
elif 5**3 > 4**4
then 15
else 12 fi;
f := (x,y) -> if (x + y > 0) then 1
              else -1 fi;
f(1,-2);
```

Boucles avec **while** ... **do** ... **od**

```
x := 2;
while x < 10**100
do
  x := x ** x;
od;
```

Boucles avec
for ... **from** ... **to** ... **by** ... **while** ...
do ... **od**

```
for i from 0 to 10 do i od;
x := 1;
for i from 1 to 42 by 2 while i < 12
do
  x := x * i:
od:
x;
```

Procédures avec **proc**

```
f := proc(a,b)
local t;
t := a; # Ça sert à rien mais bon
t ** 3 + b ** 2
end;
f(2, 3);
```

Procédures récursives (*i.e.* s'appelant elles-mêmes avec de nouveaux arguments)

```
ackermann := proc(m,n)
if (m < 0) or (n < 0) then ERROR(``m or n < 0'')
elif m = 0 then n + 1
```

```
elif n = 0 then ackermann(m - 1, 1)
else ackermann(m - 1, ackermann(m, n - 1))
fi:
end:
ackerman(3, 3);
```

1.4 Fonctions internes à connaître

À vous de consulter le manuel pour savoir ce qu'elles font et comment les utiliser !

- `sqrt`, `exp`, `ln`
- `iquo`, `irem`
- `seq`, `zip`
- `solve`
- `simplify`, `expand`, `combine`
- `int`, `diff`, `D`, `series`, `limit`, `sum`, `product`
- `plot`

2 Au boulot

▶ **Question 1** *Écrire une procédure* ***pgcd*** *qui calcule le pgcd de deux entiers positifs à l'aide de l'algorithme d'Euclide (pgcd(a,b) = pgcd(b, reste(a/b)) si b > 0, a sinon) avec une boucle* ***while***

▶ **Question 2** *Réécrire cette procédure sans boucle*

▶ **Question 3** *Écrire une procédure calculant* $u_n = 1 + \sum_{i=0}^{n-1} (n-i)u_i$ *et calculer* u_{100}

▶ **Question 4** *Calculer* $\int_0^{1/2} \frac{\ln^2(x)}{1-x} dx$

▶ **Question 5** *Tracer* $f(x) = \cos(3x) + 2\sin(x)$ *entre 0 et 10*

2.1 Développements limités

▶ **Question 6** *Calculer les développements limités usuels de* exp, ln, sin, cos... *Noter que Maple fait ses développements en O et pas en o, il faut donc monter à l'ordre suivant pour avoir nos développements en o.*

▶ **Question 7** *Donner un équivalent simple au voisinage de* $+\infty$ *de* $\sqrt{x+\sqrt{x^2+1}} - \sqrt{x+\sqrt{x^2-1}}$

2.2 Courbes planes

▶ **Question 8** *Tracer la courbe paramétrée* $x(t) = t(t+1)$, $y(t) = 2t^2(t+1)$ *sur* $-1..1 \times -1..1$

▶ **Question 9** *Déterminer les points multiples de cette courbe paramétrée*

▶ **Question 10** *Tracer la courbe en polaires* $\rho(\theta) = 1 - \sin(\theta)$

2.3 Qu'est ce que ça fait ?

Déterminer ce que font les programmes suivants en les exécutant/traçant leur exécution.

▶ **Question 11**

```
F:=proc(n)
  local i,j,k,l,x := 3;
  for i from 0 to n-1 do
    j := i*n; l :=i*j*n-j*x*k;
    x := j-l*i;
    if (x < 0) then x :=-x fi
  od; x; end proc;
```

▶ **Question 12**

```
kmp:=proc(n)
  local ret := 1;
  for i from 1 to n do
  ret := ret*i;
  end proc;
```

▶ **Question 13**

```
pgcd:=proc(r0, r2)
if r0 > r2 then r0 := r2; r2 := r0; fi;
if r0 > 0 then pgcd(irem(r2/r0),r0)
else r2;
end proc;
```

▶ **Question 14**

```
div:=proc(l, k)
local m;
if k mod 2 then l*div(l,k-1)
else m := div(l,k/2); m*m
fi;
end proc;
```

▶ **Question 15 *** *Donner un développement à l'infini de l'inverse de la fonction* $f : x \to x \times ln(x)$.