Cours de Programmation en MAPLE

- ## Remarques utiles :

1. Chaque instruction doit se terminer par **;** si vous voulez avoir un écho (ie :réponse) ou par **:** dans le cas contraire

   Exemple :

   > **a:=3^2;**

   a := 9

   > **a:=3^2:**

2. L'instruction **print** permet de retourner un message

   Exemple :

   > **print(`MAPLE c'est genial`);**

   MAPLE c'est genial

3. L'instruction **ERROR** a le même rôle que **print** on l'utilise surtout pour retourner un message d'erreur

   Exemple :

> **ERROR(`MAPLE c'est pas génial`);**

Error, MAPLE c'est pas génial

- ## Les boucles en MAPLE :

I. La boucle **for**

On l'utilise pour exécuter des instructions dépendant d'un indice entier ou pour exécuter une même instruction n fois

Syntaxe :

for k from valeur initiale de k **to** valeur finale de k

do instructions ;

od ;

Exemple :

> **a:=0;**

a := 0

> **for k from 1 to 4**

> **do a:=sqrt(a+k);**

> **od;**

a := 1

a := √ 3

a := √ (√ 3 + 3)

a := √ (√ (√ 3 + 3) + 4)

Remarques :

a. Ne jamais oublier de terminer la boucle par **od ;** sinon aucune instruction ne sera exécutée
b. Si les instructions de la boucle ne dependent pas de l'indice k on peut se permettre dans la syntaxe de ne pas écrire **for k**

   Exemple : donner les 5 restes de la division euclidienne de 15487 par 1254

   > a:=15487 :b:= 1254:liste:=NULL:

   > from 1 to 5 do

   > r:=irem(a,b):

   > liste:=liste,r:

   > a:=b:

   > b:=r:

   > od:

   > liste;

   439, 376, 63, 61 , 2

   (*) l'instruction **NULL** permet de déclarer un ensemble vide

c. Parfois on veut exécuter une suite d'instructions mais dans un ordre

   décroissant en sautant chaque fois 2 indices par exemple ,pour cela on utilise la syntaxe suivante :

   for k from valeur finale de k **to** valeur initiale de k **by –2**

   do instructions ;

od ;

Exemple: Afficher les carrés ,dans l'ordre décroissant , des nombres pairs compris

entre 1 et 10

> **for k from 10 to 2 by -2**

> **do k^2;**

> **od;**

100

64

36

16

4

d. Il est parfois possible de remplacer toute cette boucle par l'instruction **seq**

Exemple : le même objectif que précédemment

>**seq((2*(5-k))^2,k=0..4);**

100, 64, 36, 16, 4

I. La boucle **while**

On l'utilise pour exécuter des instructions jusqu'à ce qu'une condition bien donnée soit remplie

Syntaxe :

While la négation de la condition **do**

instructions ;

od ;

Remarque :

La condition inférieur ou égal se traduit en MAPLE par : **<=**

La condition inférieur strictement se traduit en MAPLE par : **<**

La condition supérieur ou égal se traduit en MAPLE par : **>=**

La condition supérieur strictement se traduit en MAPLE par : **>**

La condition inférieur ou égal se traduit en MAPLE par : **<=**

La condition diffèrent se traduit en MAPLE par : **<>**

La condition égal se traduit en MAPLE par : **=**

Exemple : trouver le dernier reste supérieur strictement a 7 parmi les restes de la division euclidienne de 15487 par 1254

```
> a:=15487 :b:= 1254: r:=irem(a,b):
> while r>7 do
> a:=b:
> b:=r:
> r:=irem(a,b):
> od:
>b ;
```


III La boucle **if**

On l'utilise pour exécuter des instructions1 si une condition est vérifiée et d'autres instructions2 dans le cas contraire.

Syntaxe :

if la condition est vérifiée **then** instructions1

else instructions2

fi ;

Exemple : écrire un programme qui donne tous les diviseurs d'un nombre premier

```
> n:=6:
> Div:=NULL:
> d:=1:
> while d<=n do
> if irem(n,d)=0 then Div:=Div,d
```

```
> else fi;
> d:=d+1;
> od:
> Div;
```

1, 2, 3, 6

Remarque :

On peut aussi utiliser la même boucle dans le cas ou la discussion porte sur plusieurs conditions

Syntaxe :

if la condition1 est verifiee **then** instructions1

elif la condition2 est verifiee **then** instructions2

else instructions3

fi ;