Fiche 3 : Boucles for, while et structure conditionnelle

I Boucle

Une boucle est une structure qui permet de répéter plusieurs fois une liste d'instruction.

Une boucle commence par une commande qui indique le type de boucle qui sera effectuée et la façon de réaliser la boucle. Cette année nous en verrons deux : for et while.

Une fois la boucle commencée, l'ordinateur exécute les instructions de haut en bas. Pour plus de lisibilité, ces instructions sont décalées par un alinéa sur la droite.

L'ordinateur poursuit jusqu'à ce qu'on lui signale que la boucle s'arrête par un end.

Une fois arrivée à end l'ordinateur reprend la première instruction et refait la même chose.

L'ordinateur continue jusqu'à ce qu'on arrive à ce qu'on appelle une condition d'arrêt qui dépend de la commande utilisée pour définir la boucle.

II Boucle for (Rappel)

```
for i=p : n
      instruction 1
      instruction 2
      instruction 3
            ⋮
end
```

Lors du premier passage dans la boucle for, l'ordinateur exécute les instructions de haut en bas en prenant p pour valeur de i.
Au second passage, il fait de même en prenant $p+1$ pour valeur de i.
Et ainsi de suite jusqu'au dernier passage où il prend n à la place de i.
Une fois ce passage effectué, la boucle s'arrête et l'ordinateur commence à exécuter les instructions suivantes.

Exemple.

```
P=1
for i=1 : 5
      P=P*i
end
```

```
P=1
for i=1 : 5
      P=P*2
end
```

III Condition

Définition 1. Une condition est une phrase mathématiques qui peut être vraie ou fausse.

Pour écrire une condition, on utilise les opérateurs de comparaison suivants :

`==` : 'est égal à'

`<>` : 'est différent de'

`<=` : 'est inférieur ou égal à'

`>=` : 'est supérieur ou égal à'

`<` : 'est strictement inférieur à'

`>` : 'est strictement supérieur à'

ATTENTION. Pour écrire la **condition** 'a égale b' on écrit `a==b` et non `a=b`.

On peut également associer les conditions entre elles par les opérateurs logiques suivants :

`&` : 'et'

`|` : 'ou'

Exemple. `1<>2`

`2<=1`

`n==3`

`(x<=1)|(x>=4)`

`(x<=2)&(x>=1)`

IV if ... then ...

```
if condition then
    instruction 1
    instruction 2
    instruction 3
        ⋮
end
```

SI la condition est réalisée, l'ordinateur exécute les instructions du haut vers le bas jusqu'à arriver à `end`, il exécute alors l'instruction qui suit `end`.
SINON l'ordinateur ignore les instructions entre `if` et `end` et exécute alors l'instruction qui suit `end`.
Exemple.

```
n=input("Entrer un entier n")
if n>=3 then
    disp("Votre entier est plus grand que 3")
end
```

V if ... then ... else ...

```
if condition then
    instructions 1
    else
    instructions 2
        ⋮
end
```

SI la condition est réalisée, l'ordinateur exécute les instructions du haut vers le bas jusqu'à arriver à `else` (les instructions 1), il exécute alors l'instruction qui suit `end`.
SINON l'ordinateur ignore les instructions entre `if` et `else` et exécute les instructions du haut vers le bas jusqu'à arriver à `end` (les instructions 2) et exécute alors l'instruction qui suit `end`.
Exemple.

```
n=input("Entrer un entier n")
if n>=3 then
    disp("Votre entier est plus grand que 3")
    else
    disp("Votre entier est strictement plus petit que 3")
end
```

VI if ... then ... elseif ... else ...

```
if condition 1 then
    instructions 1
    elseif condition 2
    instructions 2
    else
    instructions 3
        ⋮
end
```

SI la condition 1 est réalisée, l'ordinateur exécute les instructions du haut vers le bas jusqu'à arriver à `elseif` (les instructions 1), il exécute alors l'instruction qui suit `end`.
SI la condition 2 est réalisée, l'ordinateur exécute les instructions du haut vers le bas de `elseif` jusqu'à arriver à `else` (les instructions 2), il exécute alors l'instruction qui suit `end`.
SI aucune des deux conditions n'est vérifiée l'ordinateur ignore les instructions entre `if` et `else` et exécute les instructions du haut vers le bas jusqu'à arriver à `end` (les instructions 3) et exécute alors l'instruction qui suit `end`.
Exemple.

```
n=input("Entrer un entier n")
if n>3 then
    disp("Votre entier est strictement plus grand que 3")
    elseif n==3
    disp("Votre entier est égal à 3")
    else
    disp("Votre entier est strictement plus petit que 3")
end
```

VII Boucle while

```
while condition
    instruction 1
    instruction 2
    instruction 3
        ⋮
end
```

Lors du premier passage dans la boucle while, l'ordinateur vérifie que la condition est vérifiée.
Si tel est le cas, alors l'ordinateur va exécuter les instructions de haut en bas.
Au début du deuxième passage, l'ordinateur vérifie que la condition est vérifiée.
L'ordinateur poursuit tant que, lors de la vérification en début de boucle, la condition est vérifiée.

VIII Utilisation d'un compteur

Définition 2. Un compteur est un entier qui permet de compter le nombre d'itérations effectués par la boucle while.

Exemple.

```
n=0
U=1
while U>10^(-6)
    n=n+1
    U=log(n)/n
end
disp(n)
```