# Utilisation d'une boucle dans un algorithme

publié le 14/09/2012

*Descriptif :*

Des exemples d'utilisation des 2 types de boucles dans un algorithme ;

*Sommaire :*


- 1) La boucle " Pour ...Allant de ... A...."
- 2) La boucle « TANT QUE....FAIRE »
- 3) Un exemple de boucles imbriquées

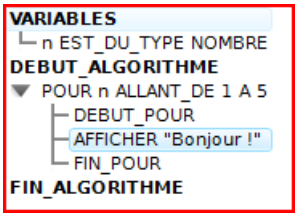

Dans un algorithme, utiliser une boucle permet de recommencer plusieurs fois un bloc d'instructions.
Il y a deux sortes de boucles :
Si l'on sait à l'avance le nombre de fois que le bloc d'instruction doit être exécuté, on utilise la boucle **"Pour ....Allant de ....A..."**
Dans le cas contraire on utilise plutôt une boucle du type **" Tant Que....Faire".**

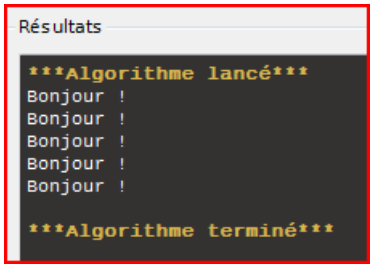
Les exemples présentés utilisent le logiciel Algobox ; les algorithmes sont adaptables sur les calculatrices scientifiques Casio et Ti.

## 1) La boucle " Pour ...Allant de ... A...."

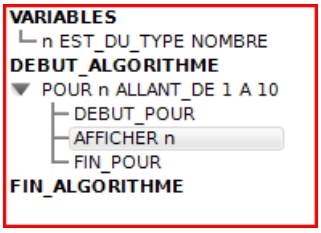
Cette boucle permet de recommencer un nombre de fois décidé à l'avance les instructions écrites entre les lignes **Pour...allant de...à ...** et **Fin_Pour**

**Exemple 1** : *Répéter 5 fois une instruction identique*
L'algorithme suivant affiche 5 fois de suite le message "Bonjour !"




**Exemple 2** : *Répéter une instruction , avec des variantes.*
L'instruction à répéter, peut dépendre de la valeur de n, comme dans l'exemple suivant :
La première fois que la ligne « DEBUT POUR » est lu, n prend la valeur 1, et chaque fois que cette ligne est à nouveau lue, la valeur de n augmente automatiquement de 1.



```
Résultats
1
2
3
4
5
6
7
8
9
10

***Algorithme terminé***
```

Si, comme le font certains débutants par erreur, on place une instruction dans la boucle pour augmenter la valeur de n , comme ci dessous, alors la valeur de n est augmentée deux fois à chaque lecture de la boucle. On obtient alors le résultat à droite, ou seuls les entiers impairs apparaissent..

```
VARIABLES
  n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  POUR n ALLANT_DE 1 A 10
    DEBUT_POUR
    AFFICHER n
    n PREND_LA_VALEUR n+1
    FIN_POUR
FIN_ALGORITHME
```

```
Résultats
***Algorithme lancé***
1
3
5
7
9

***Algorithme terminé***
```

**Exemple 3** : *Répéter une phrase, avec des variantes*

L'algorithme ci dessous, qui utilise une boucle, permet aussi de bien saisir la différence entre la commande « Ajouter Afficher message » et la commande « Ajouter Afficher variable »

```
VARIABLES
  n EST_DU_TYPE NOMBRE
  a EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  POUR n ALLANT_DE 1 A 5
    DEBUT_POUR
    a PREND_LA_VALEUR n*n
    AFFICHER "l'aire d'un carré de côté "
    AFFICHER n
    AFFICHER " est "
    AFFICHER a
    FIN_POUR
FIN_ALGORITHME
```

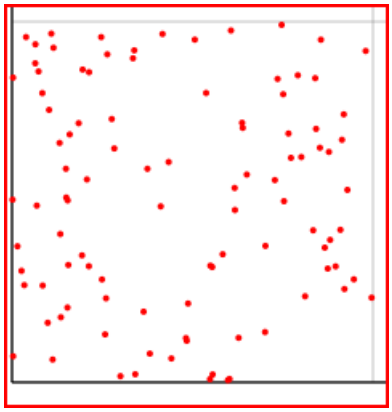
```
Résultats
***Algorithme lancé***
l'aire d'un carré de côté 1 est 1
l'aire d'un carré de côté 2 est 4
l'aire d'un carré de côté 3 est 9
l'aire d'un carré de côté 4 est 16
l'aire d'un carré de côté 5 est 25

***Algorithme terminé***
```

**Exemple 4** : *Des points aléatoires dans un carré de côté 1*

Voilà un exemple simple d'utilisation d'une boucle pour afficher 100 points aléatoires dans un repère. Les coordonnées a et b des points affichés sont des nombres de l'intervalle [0 ;1], obtenues grâce à la fonction random(). Les points changent de place à chaque fois que l'algorithme est lancé.

```
VARIABLES
  n EST_DU_TYPE NOMBRE
  a EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  POUR n ALLANT_DE 1 A 100
    DEBUT_POUR
    a PREND_LA_VALEUR random()
    b PREND_LA_VALEUR random()
    TRACER_POINT (a,b)
    FIN_POUR
FIN_ALGORITHME
```



A noter que les 3 lignes de la boucle auraient pu être condensées en une seule, ce qui permet de n'utiliser qu'une variable. Ainsi, l'algorithme suivant fait la même chose que le précédent.

```
VARIABLES
  n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  POUR n ALLANT_DE 1 A 100
    DEBUT_POUR
    TRACER_POINT (random(),random())
    FIN_POUR
FIN_ALGORITHME
```

**Exemple 5** : *Construction d'un polygone régulier*

Cet algorithme permet de dessiner un polygone régulier comportant un nombre de côtés défini par l'utilisateur. Avec Algobox, c'est l'instruction "Lire p" qui permet de donner à p la valeur souhaitée.

Les p sommets peuvent être les points suivants :

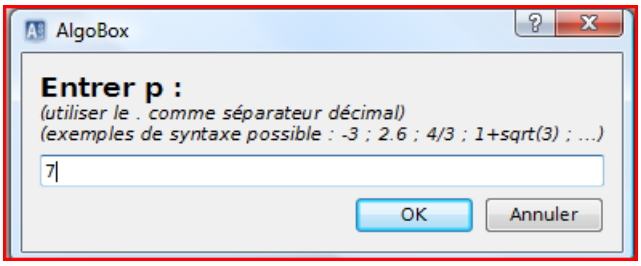
$A_k\left(\cos\left(k\times\frac{2\pi}{p}\right),\sin\left(k\times\frac{2\pi}{p}\right)\right)$ ou $k$ varie de 0 à p-1 ou de 1 à p, ce qui revient au même.

il suffit alors de tracer les p segments qui relient chaque sommet à son sommet voisin.

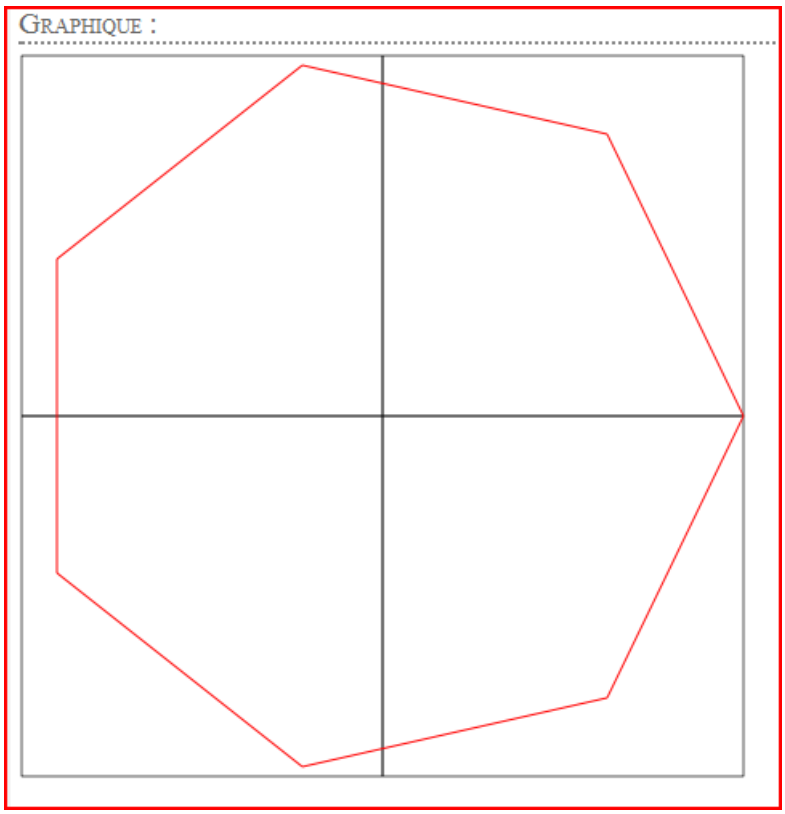
On obtient l'algorithme très court suivant :

```
VARIABLES
  n EST_DU_TYPE NOMBRE
  p EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE p
  POUR n ALLANT_DE 1 A p
    DEBUT_POUR
    TRACER_SEGMENT (cos(2*(n-1)*Math.PI/p),sin(2*(n-1)*Math.PI/p))->(cos(2*n*Math.PI/p),sin(2*n*Math.PI/p))
    FIN_POUR
FIN_ALGORITHME
```

lorsqu'on demande l'exécution de l'algorithme, celui ci demande la valeur de p.




En donnant à p la valeur 7, on obtient le résultat suivant :



## ● 2) La boucle « TANT QUE....FAIRE »

Dans les exemples suivants, on ne sait pas à l'avance le nombre de fois que la boucle va être lue. La boucle étudiée au §1 n'est donc pas adaptée.

**Exemple 1 :** *Deviner un nombre*

Un nombre A est choisi aléatoirement entre 1 et 10 par l'algorithme. L'utilisateur essaye de deviner le nombre. Le processus est arrêté lorsque le nombre est trouvé. Un compteur n comptabilise le nombre d'essais nécessaires.

```
VARIABLES
  A EST_DU_TYPE NOMBRE
  E EST_DU_TYPE NOMBRE
  n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  n PREND_LA_VALEUR 0
  E PREND_LA_VALEUR -1
  A PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,10)
  TANT_QUE (E!=A) FAIRE
    DEBUT_TANT_QUE
    n PREND_LA_VALEUR n+1
    LIRE E
    FIN_TANT_QUE
  AFFICHER "nombre deviné en  "
  AFFICHER n
  AFFICHER " essais"
FIN_ALGORITHME
```

```
Résultats
***Algorithme lancé***
nombre deviné en  9 essais
***Algorithme terminé***
```

Remarques : Noter l'initialisation de E à la valeur -1, pour être sûr de pouvoir entrer dans la boucle une première fois.

la syntaxe "E !=A" signifie E différent de A dans Algobox.

**Exemple 2 :** *Faire travailler son petit frère*

Pour réviser les tables de multiplication, l'algorithme suivant choisit au hasard 2 entiers A et B compris entre 1 et 9, et il demande qu'on lui donne le produit R=AxB.

Tant que la réponse est fausse la question est posée à nouveau.

```
VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  R EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  A PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,9)
  B PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,9)
  R PREND_LA_VALEUR -1
  TANT_QUE (R!=A*B) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "combien font "
    AFFICHER A
    AFFICHER " multiplié par "
    AFFICHER B
    AFFICHER " ?"
    LIRE R
    AFFICHER R
    FIN_TANT_QUE
  AFFICHER "juste, bravo !"
FIN_ALGORITHME
```

```
***Algorithme lancé***
combien font 8 multiplié par 7 ?
48
combien font 8 multiplié par 7 ?
63
combien font 8 multiplié par 7 ?
56
juste, bravo !
***Algorithme terminé***
```

**Exemple 3** *Recherche d'une racine par dichotomie*

L'algorithme suivant détermine un encadrement de largeur e de la racine cubique de 2. On sait que ce nombre est compris entre 1 et 2. La précision e est déterminée par l'utilisateur (0,1 ou 0,01...). La fonction F1 est la fonction définie par $F1(x) = x^3 - 2$

Cet algorithme est plus difficile à construire que les autres mais il doit être étudié au lycée.

Voilà cet algorithme, et le résultat obtenu si on choisit e=0,0001.

```
VARIABLES
  a EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
  e EST_DU_TYPE NOMBRE
  m EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  a PREND_LA_VALEUR 1
  b PREND_LA_VALEUR 2
  LIRE e
  TANT_QUE (b-a>e) FAIRE
    DEBUT_TANT_QUE
    m PREND_LA_VALEUR (a+b)/2
    SI (F1(a)*F1(m)<=0) ALORS
      DEBUT_SI
      b PREND_LA_VALEUR m
      FIN_SI
      SINON
        DEBUT_SINON
        a PREND_LA_VALEUR m
        FIN_SINON
    FIN_TANT_QUE
  AFFICHER a
  AFFICHER b
FIN_ALGORITHME
```

```
Résultats
***Algorithme lancé***
1.2598877
1.2599487
***Algorithme terminé***
```

Remarque : Trouver un intervalle de largeur $10^{-n}$ qui contient un nombre p ne signifie pas que l'on connaisse n décimales de p.

**Exemple 4** *Recherche de seuil pour une suite.*
On va considérer la suite U suivante, définie par récurrence :
$U_0 = 0.5$ et pour tout entier n, $U_{n+1} = 3U_n(1-U_n)$
Supposons démontré que cette suite a pour limite $2/3$, et que les termes de cette suite se rapprochent de la limite au fur et à mesure que le rang augmente.
Ecrivons un algorithme qui permet de savoir pour quelles valeurs de n , $|U_n - \frac{2}{3}| < 0{,}01$

```
VARIABLES
  n EST_DU_TYPE NOMBRE
  U EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  U PREND_LA_VALEUR 0.5
  n PREND_LA_VALEUR 0
  TANT_QUE (abs(U-2/3)>0.01) FAIRE
    DEBUT_TANT_QUE
    U PREND_LA_VALEUR 3*U*(1-U)
    n PREND_LA_VALEUR n+1
    FIN_TANT_QUE
  AFFICHER n
  AFFICHER U
FIN_ALGORITHME
```

```
Résultats
***Algorithme lancé***
529
0.67665492
***Algorithme terminé***
```

On voit donc qu'il faut attendre le 530 ème terme de la suite pour s'approcher de la limite à moins de 1 centième. Il est vrai que la suite a été choisie pour sa convergence lente.
On peut donc écrire : Pour tout entier $n >= 529$, $|U_n - \frac{2}{3}| < 0{,}01$

## 3) Un exemple de boucles imbriquées

- Avant de donner un tel exemple, remarquons qu'une boucle du type "Pour ...Allant_de.....A...." peut toujours être remplacée par une boucle du type " Tant Que.....Faire...."
  Ainsi l'algorithme de gauche vu plus haut, à l'exemple 1, peut être remplacé par celui de droite, le résultat est le même :

```
VARIABLES
  n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  POUR n ALLANT_DE 1 A 5
    DEBUT_POUR
    AFFICHER "Bonjour !"
    FIN_POUR
FIN_ALGORITHME
```

```
VARIABLES
  n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  n PREND_LA_VALEUR 0
  TANT_QUE (n<5) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "bonjour"
    n PREND_LA_VALEUR n+1
    FIN_TANT_QUE
FIN_ALGORITHME
```

Par contre, il sera le plus souvent impossible de transformer une boucle du type "Tant Que ...Faire...." en une boucle "Pour ...Allant_de.....A...."

- On va reprendre l'exemple de la table de multiplication, et imaginer que l'on pose 10 multiplications qui rapporte chacune 2 points si la réponse est juste au premier essai et 1 point si elle juste au deuxième essai. l'exemple 2 du §2 va être modifié, et imbriqué dans une nouvelle boucle "Pour.....Allant de....A...". Il va falloir aussi inclure 2 compteurs C1 et C2, le premier calculant le score sur 20, le second comptant à chacune des 10 questions, le nombre d'essais nécessaires pour obtenir une réponse juste.

```
VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  R EST_DU_TYPE NOMBRE
  C1 EST_DU_TYPE NOMBRE
  C2 EST_DU_TYPE NOMBRE
  N EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  C1 PREND_LA_VALEUR 0
  POUR N ALLANT_DE 1 A 10
    DEBUT_POUR
    A PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,9)
    B PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,9)
    R PREND_LA_VALEUR -1
    C2 PREND_LA_VALEUR 0
    TANT_QUE (R!=A*B) FAIRE
      DEBUT_TANT_QUE
      AFFICHER "combien font "
      AFFICHER A
      AFFICHER " multiplié par "
      AFFICHER B
      AFFICHER " ?"
      LIRE R
      C2 PREND_LA_VALEUR C2+1
      SI (C2==1 ET R==A*B) ALORS
        DEBUT_SI
        C1 PREND_LA_VALEUR C1+2
        FIN_SI
      SI (C2==2 ET R==A*B) ALORS
        DEBUT_SI
        C1 PREND_LA_VALEUR C1+1
        FIN_SI
      FIN_TANT_QUE
    FIN_POUR
  AFFICHER "Ton score est : "
  AFFICHER C1
FIN_ALGORITHME
```

Remarques :
On peut compliquer l'exercice en faisant varier A et B entre 2 et 12
Dans le **SI...Alors**, on peut associer 2 conditions séparées par ET ou par OU.

**Académie de Poitiers**

Avertissement : ce document est la reprise au format pdf d'un article proposé sur l'espace pédagogique de l'académie de Poitiers.

Il ne peut en aucun cas être proposé au téléchargement ou à la consultation depuis un autre site.