# Mode d'emploi de la Calculatrice programmable de Guillaume Nollet (élève de seconde)

## Un grand merci pour le travail accompli

On va aller étape par étape. On va pas tout expliquer en même temps. Ne vous précipitez pas. Ne vous bloquez pas si vous ne comprenez pas : relisez attentivement et intelligemment, demandez à quelqu'un d'autre, appelez-moi au téléphone (sauf qu'en fait j'en ai pas). Merci.

L'ensemble des commandes qui vont suivre ont été vérifiées sur la TI-Nspire. Elles marchent.

# La boucle itérative "for"

On va étudier cette boucle en plusieurs temps, parce que c'est compliqué à comprendre. C'est pas pour autant qu'il faut se décourager ou aller trop vite !

### À quoi elle sert ?

La boucle "for" permet de faire les mêmes instructions plusieurs fois.

Si je veux faire la même commande plusieurs fois, comme "afficher 5 fois le texte [BONJOUR!] ", la boucle "for" est très utile.

### Représentation en pseudo-code

En pseudo-code, la boucle se présente comme ça :

```
DébutFor : répéter 5 fois :
     Afficher "BONJOUR!"
FinFor
```

C'est plutôt simple, non ?

### Le programme sur la calculatrice

```
For a,1,5
    Disp "BONJOUR!"
EndFor
```

Ne vous préoccupez pas du "a" pour l'instant. Si on ne compte pas ça, c'est quand même très ressemblant !

Seul problème, il y a un 1 avant le 5. Pourquoi ? Parce que la calculatrice va prendre en compte le 1 **et** le 5 pour voir combien de fois elle va répéter le même programme. Elle va penser comme ça :

- Instruction "1" : Afficher "BONJOUR!" ← Depuis 1…
- Instruction "2" : Afficher "BONJOUR!"
- Instruction "3" : Afficher "BONJOUR!"
- Instruction "4" : Afficher "BONJOUR!"
- Instruction "5" : Afficher "BONJOUR!" ← …jusqu'à 5 !

**Les deux nombres, de départ comme d'arrivée, sont inclus.**

Il est possible de changer les nombres, tant que si on inclut les deux nombres et tous ceux qui sont entre les deux, ça fait 5. Par exemple :

| **Si on met :** | |
|---|---|
| `For a, 0, 4`<br>`    Disp "BONJOUR!"`<br>`EndFor` | `For a, 12, 16`<br>`    Disp "BONJOUR!"`<br>`EndFor` |
| **La calculatrice pense :** | |
| Instruction "0" : Afficher "BONJOUR!"<br>Instruction "1" : Afficher "BONJOUR!"<br>Instruction "2" : Afficher "BONJOUR!"<br>Instruction "3" : Afficher "BONJOUR!"<br>Instruction "4" : Afficher "BONJOUR!" | Instruction "12" : Afficher "BONJOUR!"<br>Instruction "13" : Afficher "BONJOUR!"<br>Instruction "14" : Afficher "BONJOUR!"<br>Instruction "15" : Afficher "BONJOUR!"<br>Instruction "16" : Afficher "BONJOUR!" |

Au final, le résultat est le même : le message "BONJOUR!" est affiché 5 fois.

*Note : pour ceux qui ont déjà programmé, la commande fonctionne comme la boucle de répétition 'loop'. Pour ceux qui ont jamais programmé, on s'en fiche de cette note.*

**Et pourquoi la calculatrice pense toujours de 1 chiffre par 1 chiffre ?**

Pour ceux qui n'ont pas compris cette question, elle veut dire ça : "Pourquoi la calculatrice elle pense en **1**, **2**, **3**, **4**, **5**, et pas en **1**, 1.5, **2**, 2.5, **3**, 3.5, **4**, 4.5, **5** ?"

Parce que c'est par défaut dans la calculatrice. Le "pas" entre les nombres est de 1, par défaut. Alors comment le changer ? Comme ça :

```
For a, 1, 5, 0.5
    Disp "BONJOUR!"
EndFor
```

Il faut rajouter le nouveau "pas" dont on a besoin. Ici, le pas c'est 0,5. Du coup, la calculatrice va penser comme ça :

- Instruction "1" : Afficher "BONJOUR!" ← De 1…
- Instruction "1.5" : Afficher "BONJOUR!" ← …avec un pas de 0,5…
- …
- Instruction "4.5" : Afficher "BONJOUR!"
- Instruction "5" : Afficher "BONJOUR!" ← …jusqu'à 5 !

Concrètement, dans le cadre de "BONJOUR!", ça va l'afficher **9** fois, du coup. Une fois pour "1", une fois pour "1.5", une fois pour "2", une fois pour "2.5"…

"Mais si ça sert qu'à ça, alors on pourrait pas juste faire `For a, 1, 9` ?", me direz-vous.

Et vous auriez raison. C'est vachement beaucoup de trucs compliqués pour rien. Enfin, pour l'instant…

**Ce à quoi sert le "a" du programme**

Eh oui, ce "a" n'est pas juste là pour décorer ! Et c'est là que ça devient intense, et qu'il va bien falloir vous accrocher. Si vous ne comprenez pas du premier coup, réessayez.

"a", c'est comme ça qu'on va nommer le nombre le temps du décomptage.

Par exemple, qu'est-ce qui se passe si je mets ça ?

```
For a, 1, 5
    Disp a
EndFor
```

Ça ferait afficher la variable "a" 5 fois, pour l'instant, on est d'accord. Sauf que, combien vaut "a" ?
Eh bien, "a" vaut le nombre auquel vous êtes pendant le compte de la boucle. Ça a l'air hyper compliqué, sauf qu'en fait c'est hyper simple. Regardez plutôt :

| **Pendant l'instruction…** | **…On a "a" qui vaut :** | **Résultat affiché sur la calculatrice** |
|---|---|---|
| Instruction "1" : Afficher a | a=1 | 1 |
| Instruction "2" : Afficher a | a=2 | 2 |
| Instruction "3" : Afficher a | a=3 | 3 |
| Instruction "4" : Afficher a | a=4 | 4 |
| Instruction "5" : Afficher a | a=5 | 5 |
| | | *Terminé* |

Vous avez pas tout compris ? Essayez le même programme sur votre propre calculatrice, ce sera plus clair ! (pour de vrai, c'est pas pour vous embêter)

C'est bon ? C'est compris ? Vous comprenez, maintenant, pourquoi je vous disais qu'on pouvait faire pareil avec d'autres nombres que 1 et 5. Lisez colonne par colonne :

| **Si on met :** | | |
|---|---|---|
| `For a, 0, 4`<br>`  Disp a`<br>`EndFor` | `For a, 12, 16`<br>`  Disp a`<br>`EndFor` | `For a, 1, 4, 0.5`<br>`  Disp a`<br>`EndFor` |
| **La calculatrice pense :** | | |
| Instruc° "0" : Afficher a (a=0)<br>Instruc° "1" : Afficher a (a=1)<br>Instruc° "2" : Afficher a (a=2)<br>Instruc° "3" : Afficher a (a=3)<br>Instruc° "4" : Afficher a (a=4) | Instruc° "12" : Afficher a (a=12)<br>Instruc° "13" : Afficher a (a=13)<br>Instruc° "14" : Afficher a (a=14)<br>Instruc° "15" : Afficher a (a=15)<br>Instruc° "16" : Afficher a (a=16) | Instruc° "1" : Afficher a (a=1)<br>Instruc° "1.5" : Afficher a (a=1.5)<br>Instruc° "2" : Afficher a (a=2)<br>Instruc° "2.5" : Afficher a (a=2.5)<br>Instruc° "3" : Afficher a (a=3)<br>Instruc° "3.5" : Afficher a (a=3.5)<br>Instruc° "4" : Afficher a (a=4) |
| **La calculatrice affiche :** | | |
| 0<br>1<br>2<br>3<br>4<br>*Terminé* | 12<br>13<br>14<br>15<br>16<br>*Terminé* | 1<br>1.5<br>2<br>2.5<br>3<br>3.5<br>4<br>*Terminé* |

Vous voyez, je ne fais pas tout ça dans le but de vous fatiguer (enfin pas que)…

Et le "a", comment on l'appelle ? **Une variable**. C'est rien de plus qu'une variable : elle est définie comme nombre et varie au cours du temps.
Bien évidemment, la variable que j'utilise s'appelle "a", mais vous pouvez mettre n'importe quoi. Si j'ai mis "a", c'est parce qu'y avait pas la place pour autre chose dans les tableaux de Word.

## Autre chose que l'affichage

Pour tous les exemples en haut, j'ai utilisé la commande "Disp". Mais bien sûr, vous pouvez utiliser n'importe quelle autre commande.

Pour l'instant, on parle surtout d'affectations de variables, mais regardez quand même (de colonne en colonne) :

| **Si on met :** | |
|---|---|
| `0→b`<br>`For a, 1, 5`<br>`  a+b→b`<br>`EndFor`<br>`Disp b` | `For a, 1, 5`<br>`  a+b→b`<br>`  Disp b`<br>`EndFor` |
| **La calculatrice pense :** | |
| Instruction "1" : b = a+b = 1+0 = 1<br>Instruction "2" : b = a+b = 2+1 = 3<br>Instruction "3" : b = a+b = 3+3 = 6<br>Instruction "4" : b = a+b = 4+6 = 10<br>Instruction "5" : b = a+b = 5+10 = 15 | Instruction "1" : b = a+b = 1+0 = 1 + afficher b<br>Instruction "2" : b = a+b = 2+1 = 3 + afficher b<br>Instruction "3" : b = a+b = 3+3 = 6 + afficher b<br>Instruction "4" : b = a+b = 4+6 = 10 + afficher b<br>Instruction "5" : b = a+b = 5+10 = 15 +afficher b |
| **La calculatrice affiche :** | |
| 15<br>*Terminé* | 1<br>3<br>6<br>10<br>15<br>*Terminé* |
| **Et concrètement :** | |
| Cet algorithme sert à calculer la somme des 5 premiers nombres : 1+2+3+4+5 = 15 | Cet algorithme sert à afficher la somme des 1, 2, 3, 4 puis 5 premiers nombres :<br>1=1  1+2=3  1+2+3=6  1+2+3+4=10<br>1+2+3+4+5=15 |

Vous voyez, ça peut avoir une utilité pour calculer. Là, je n'ai fait que les 5 premiers nombres, mais souvenez-vous de l'exercice 43 du cahier d'algorithmique : il faut faire la somme des 1 000 premiers nombres entiers !

Du coup, il suffit simplement de modifier ceci :

```
0→b
For a, 1, 5
     a+b→b
EndFor
Disp b
```

```
0→b
For a, 1, 1000
→         a+b→b
EndFor
Disp b
```

Et voilà, vous avez le code de l'exercice 43 !

**Et un peu d'interaction !**

Toujours avec l'exemple de l'exercice 43 : plutôt que d'entrer "5" ou "1000" manuellement, on va *demander à l'utilisateur* quel nombre il faut entrer.

Souvenez-vous, pour ça, il faut la commande "request", et donc une (autre) variable.

Eh ben on la rajoute !

```
Request "La somme des X premiers nombres:", var
0→b
For a, 1, var
     a+b→b
EndFor
Disp b
```

Et voilà, vous avez un programme qui calcule la somme de n'importe quel nombre !
Les variables, on peut les mettre à la place de n'importe quel nombre, ça marchera quand même. C'est le but des variables, pouvoir tout modifier sans entrer dans le code.

Rappelez-vous, la communication avec l'utilisateur est **importante**. Vous n'allez certainement pas acheter un téléphone portable qui vous demande d'entrer dans le code source à chaque fois juste pour envoyer un SMS. Eh ben là, c'est pareil.

C'est bon, on en a enfin fini avec la mystérieuse boucle "for" ! Réjouissez-vous !