Dans cette seconde leçon de l'Unité 4 vous allez étudier la boucle **While…End**. Nous allons la comparer à la boucle **For**… et également montrer qu'elle est plus puissante et plus polyvalente que la boucle **For…**.

Objectifs :

- Étudier la structure de la boucle **While…End**.
- La comparer à la boucle **For…End**.
- Voir comment elle est utilisée pour assurer la validité des valeurs entrées.

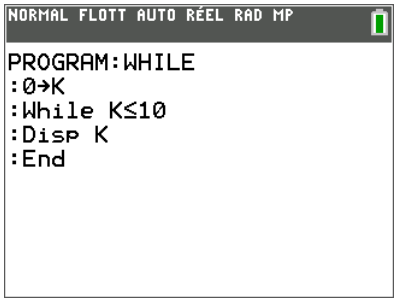
La boucle While… End

La boucle **While…End** continue tant que la <condition> est Vraie. Cela ressemble à ça :

While <condition>
  <corps de la boucle>
End

Note :
La <condition> est une expression logique telle que **X>0.**
Le <corps de la boucle> est un ensemble d'instructions, incluant d'autres boucles et structures **If***. Elle est traitée tant que la <condition> est* Vraie*.*
Le mot-clé **End** *est utilisé pour indiquer la fin du <corps de la boucle>. Lorsqu'il rencontre l'instruction* **End** *le programme retourne à l'instruction* **While** *et teste de nouveau la <condition>.*

'Initialiser' la condition de la boucle **While** : établir une valeur de telle sorte que la condition peut être évaluée à Vrai ou Faux. Si la condition au départ est Fausse la boucle est totalement ignorée. Si la condition est Vraie alors le corps de la boucle est traité. L'affectation **0→K** au début du programme initialise la condition à Vrai. Sans cela, il n'y a aucun moyen de savoir ce qui va arriver parce que n'importe quelle valeur a pu être stockée dans la variable K avant le lancement du programme.
Quelque part dans le <corps de la boucle> il doit y avoir une instruction qui doit affecter la <condition> afin que la boucle finisse par se terminer et que les instructions suivantes soient traitées. Généralement cette instruction est près de la fin du <corps de la boucle>. **K+1→K** assure que **K** finira par devenir plus grand que 10.



Une boucle 'infinie' ! Pourquoi ?

Que va faire ce programme ?

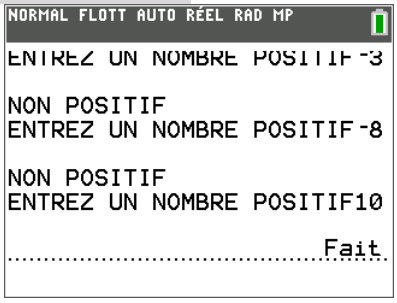
Indication : Il est important d'insister sur le fait que la boucle **While** peut ne pas être traitée du tout. Dans la leçon suivante nous allons examiner la boucle **Repeat** qui est toujours traitée au moins une fois. C'est une subtile mais importante distinction.
Il y a trois éléments pour construire avec succès une boucle **While** : ***Initialisation, test, et changement***.

- ***Initialiser*** la variable.
- ***Tester*** la condition basée sur cette variable.
- ***Changer*** la valeur de la variable pour que la condition finisse par devenir fausse pour que la boucle se termine.

Contrôle de la validité d'un ''Input'' avec While…End

Nous allons écrire une partie de programme (un 'segment de code') qui permet de s'assurer que l'utilisateur entre un nombre positif, en lui disant que le nombre entré n'est pas valide et d'entrer une autre valeur à sa place.

La sortie de ce segment de code est montrée à droite avec l'entrée de nombres négatifs pour voir l'effet.
Voyez si vous pouvez l'écrire cela sans regarder la suite !



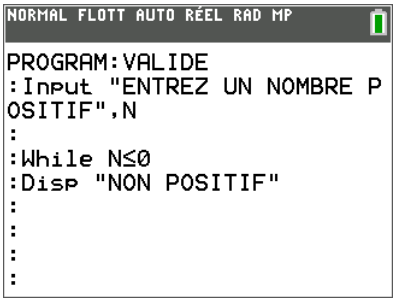
Note : Les utilisateurs de calculatrices TI-8x peuvent avoir à utiliser un message plus court comme l'écran n'est pas aussi large.

Nous commençons par une instruction **Input** avec un message afin que l'utilisateur entre une valeur…

Indication : Ceci ***initialise*** N.

Ensuite commencez la boucle **While**, testez si la valeur entrée est négative. Si c'est le cas, affichez un message d'erreur.

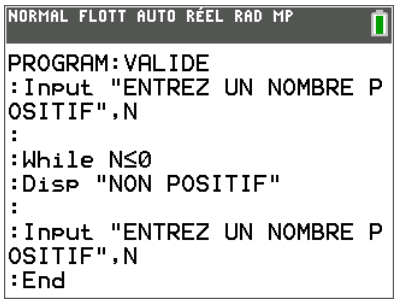
On entre dans le corps de la boucle seulement si N est négatif, sinon la boucle entière est ignorée.



Indication : Ceci définit la condition impliquant N, à ***tester***.

Enfin, utilisez l'instruction **Input** de nouveau à la fin du <corps de la boucle> pour demander à l'utilisateur une autre valeur et terminez le <corps de la boucle>.

Ce segment de code va demander à l'utilisateur une valeur et s'assurer que le nombre entré est positif. Après ce segment de code, il y aura d'autres instructions permettant de traiter le nombre entré.



Indication : Ceci donne la possibilité de **changer** la valeur de N.