Dans cette troisième leçon de l'Unité 4 vous allez étudier la boucle **Repeat…End**. Nous allons la comparer à la boucle **For**… et montrer pourquoi elle est plus puissante.

Objectifs :

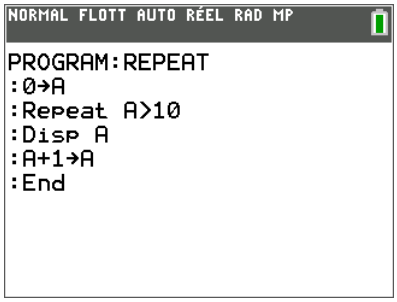
- Étudier la structure et la logique de la boucle **Repeat…End**.
- La comparer à la boucle **While…End**.
- Voir comment **Repeat…End** est utilisée dans la programmation de la suite de Fibonacci.

Indication : Bien que la condition de la boucle **Repeat** soit écrite au début de la boucle, elle n'est en fait testée qu'à la fin de la boucle ! Une boucle **Repeat** par conséquent est toujours exécutée au moins une fois.
Il y a dans le langage, à la fois les boucles **While** et **Repeat**, car elles ont un comportement légèrement différent.

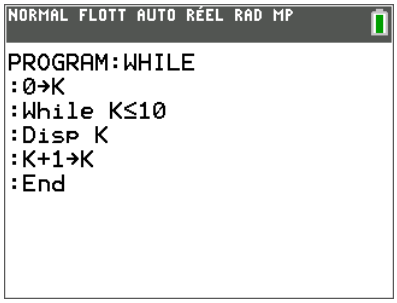
La boucle Repeat… End

La boucle **Repeat…End** continue tant que la <condition> est Fausse. C'est exactement le *contraire* du comportement de la boucle **While** et ce n'est pas la seule différence ! Cela ressemble à ça (c'est assez proche de la structure **While**) :

Repeat <condition>
<corps de la boucle>
End



Les programmes à droite donnent la même chose. Quelles sont les différences?



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**. Il est **exécuté une fois**, puis continue **jusqu'à** ce que la <condition> soit vraie, ainsi le comportement ressemble plutôt à ceci :

Repeat
<corps de la boucle>
until <condition> vraie **End**

Mais il n'y a pas de mot-clé 'until' en TI-Basic. Il est implicite. Même si la <condition> est vraie au départ, le corps de la boucle est toujours traité au moins une fois car la condition est testée en fin de boucle.

Dans une boucle **While**, il est très important 'd'initialiser' la ou les variable(s) dont dépend la boucle. Dans une boucle **Repeat** ceci se passe à l'intérieur du corps de la boucle, ainsi l'initialisation n'est pas nécessaire.

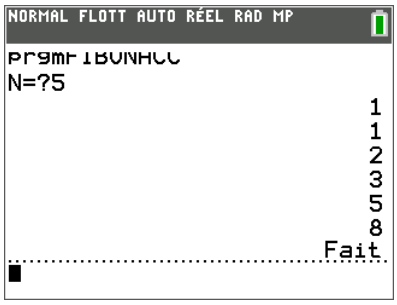
Comme avec **While**, quelque part dans le <corps de la boucle> on doit trouver une instruction qui doit avoir un effet sur la <condition> afin que la boucle finisse

par se terminer et que les instructions suivantes soient exécutées. Généralement cette instruction est près de la fin du <corps de la boucle>.

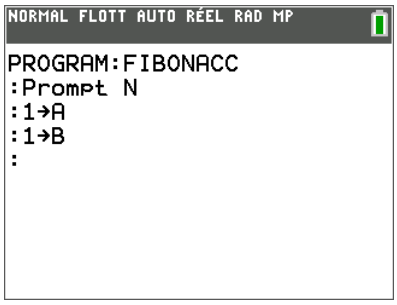
Programmation de la suite de Fibonacci

Écrivons un programme qui affiche la suite de *Fibonacci* jusqu'à une certaine valeur. Vous pouvez chercher la suite de *Fibonacci* si vous n'en avez jamais entendu parler.

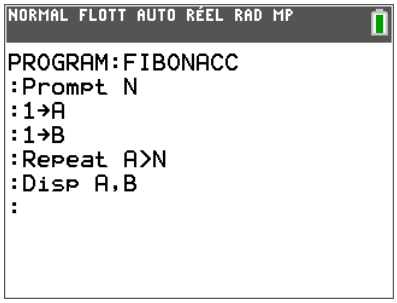
La sortie du programme est montrée à droite. Pouvez-vous écrire ce programme sans regarder plus bas ?



Nous commençons par une instruction **Prompt** pour que l'utilisateur donne **N** la borne supérieure. Les deux premiers nombres de Fibonacci sont égaux à 1 aussi nous stockons cette valeur dans les variables **A** et **B**. Ces variables vont être utilisées pour calculer les autres nombres de Fibonacci (jusqu'à **N**).

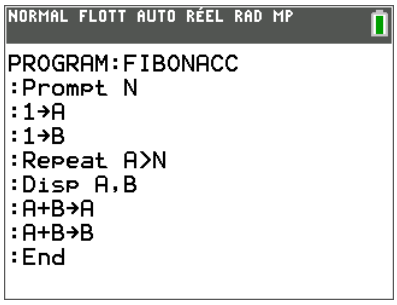


Puis on commence la boucle **Repeat** en utilisant la condition **A>N**, signifiant ''*jusqu'à ce que* **A** soit plus grand que **N**''. Dans la boucle nous affichons en premier les valeurs courantes des deux variables **A** and **B**.



Enfin, nous calculons les deux nombres de Fibonacci suivants et **terminons** la boucle.

Les deux dernière instructions montrent que A+B est stocké à la fois dans A et dans B. Il semble donc que A et B reçoivent la même valeur... **Ce qui n'est pas le cas !** Essayez avec 1 et 1. La première instruction stocke 1+1 dans A, la valeur contenue dans A est donc 2. La seconde instruction stocke 2+1 dans B, la valeur contenue dans B est donc 3. Essayez vous même !



Exécutez le programme avec plusieurs valeurs différentes. Ce comporte-t-il comme prévu ?

Essayez de remplacer la condition de **Repeat** par **B>N**. Quel est l'effet ? Comment peut-on modifier le programme pour afficher l'ensemble 'correct' des nombres, s'arrêtant lorsqu'exactement le premier nombre de Fibonacci plus grand que **N** a été affiché ?