**Unité 3: Luminosité, IF, et WHILE** | **Compétence 1 : Mesure de la luminosité**

Dans cette première leçon de l'unité 3, nous étudions le capteur de luminosité intégré, et introduisons la commande **Output**( en TI-Basic afin d'illustrer une technique efficace permettant d'afficher des nombres de caractéristiques (nombre de décimales) différents **Output**(.

**Objectifs :**

- Lire la valeur de la luminosité reçue par un capteur
- Introduire la boucle **While**
- Utiliser la commande **Output**(
- Introduire **versChaîne**( et la concaténation

Dans les leçons précédentes, nous avons seulement envoyé des instructions au Hub TI-Innovator™ afin d'obtenir une action sur les composants qui équipent ce système (LUMIERE, COULEUR, et SON).


Dans cette unité, nous allons travailler avec le capteur de lumière intégré afin de créer un appareil capable de mesurer la quantité de lumière reçue. Le capteur de lumière produit des valeurs dans l'intervalle (0 à 100) sous forme décimale.

Pour obtenir une mesure de la grandeur proportionnelle à la quantité de lumière reçue par le Hub TI-Innovator la commande exige deux arguments.

- **Send("READ BRIGHTNESS")**
- **Get(B)**

**Mise au point du programme :**

1. Commencer un nouveau programme et le nommer LUMIN1.
2. Ajouter la commande **Effécran** et **Disp** pour afficher le titre comme montré sur l'écran de droite.
3. Appuyer sur la touche [prgm] et utiliser les flèches de direction pour atteindre le menu ***HUB***.
4. Sélectionner **2: Send("READ…** puis choisir **1: BRIGHTNESS**.
5. Appuyer sur la touche [prgm] et utiliser les flèches de direction pour atteindre le menu ***HUB***.
6. Sélectionner **5: Get(**

```
NORMAL FIXE2 AUTO RÉEL RAD MP
PROGRAM:LUMIN1
:Effécran
:Disp "CAPTEUR DE LUMIERE"

:
:Send("READ BRIGHTNESS ")
:Get(B)
:
:
:
```

Comment cela fonctionne-t-il ?

- **READ BRIGHTNESS** demande au Hub TI-Innovator de lire le niveau de luminosité et stocke la valeur dans sa mémoire tampon "buffer"
- **Get(B)** est une commande pour obtenir la valeur depuis le Hub TI-Innovator. Cette instruction transfère la mesure de la mémoire tampon vers une variable B dans la TI-83 Premium CE. B peut être remplacée par n'importe quelle variable numérique, A..Z and Θ (theta).

**Conseil à l'enseignant :** Une mémoire tampon « buffer » est une zone de mémoire sur le Hub TI-Innovator qui conserve temporairement une valeur. Celle-ci est mise à jour lorsqu'une nouvelle demande de lecture est envoyée, ainsi il est fortement recommandé de faire succéder la commande **READ** par une instruction **Get** afin de conserver la valeur lue sur le Hub TI-Innovator dans une variable de la calculatrice. Il est aussi possible de

conserver les valeurs lues depuis le Hub TI-Innovator et de les stocker dans une liste pour une future analyse, mais cela dépasse le cadre de cette introduction.

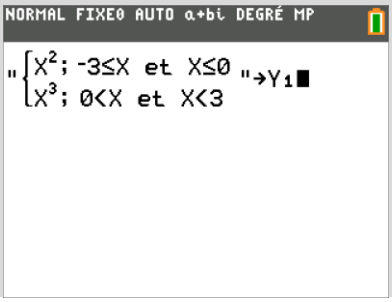
**La boucle While :**

La boucle **While…End** (menu prgm CTL) est utilisée pour réaliser une portion de code tandis qu'une *condition* est vraie. Une *condition* est une instruction logique pouvant être évaluée comme étant vraie ou fausse. Les opérateurs relationnels et logiques se trouvent dans le menu **test** de la calculatrice (**2nd math**).

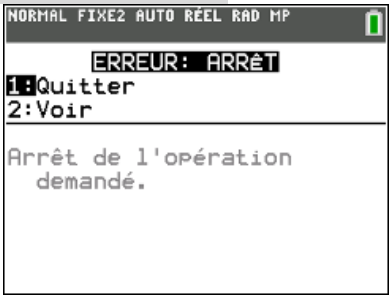
- Les opérateurs de relation sont **=**, **≠** **<**, **>**, **≤**, et **≥**.
- Les opérateurs logiques sont **et**, **ou**, **non(**, et **xor**

Ces opérateurs peuvent être utilisés ensemble afin de composer des instructions composées **x>0 and y>0.**

**Conseil à l'enseignant :** Il existe un autre menu dans celui des **tests**: CONDITIONS. Celles-ci sont utilisées pour construire des fonctions par morceaux. Voir **maths** B : par morceaux(



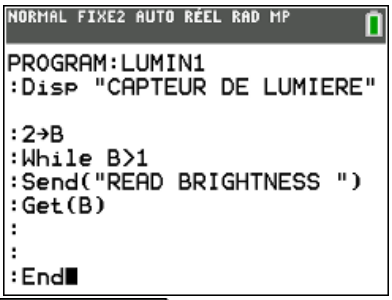

Nous allons utiliser une simple boucle **While** qui s'arrête lorsque la luminosité atteint une valeur inférieure à 1. Pour terminer le programme, couvrir simplement avec votre main le capteur de lumière sur la face latérale du Hub TI-Innovator.




Une autre méthode pour arrêter brutalement un programme consiste à appuyer sur la touche **ON**. Vous verrez un message d'erreur : **ERROR : ARRÊT** en haut de l'écran avec deux options **1: Quitter** pour retourner à l'écran d'accueil et **2: Voir** l'éditeur de programme s'ouvre alors en la ligne où a été appelée l'interruption **ON** stoppe le programme. C'est aussi une bonne méthode pour continuer l'édition d'un programme.

**Ajouter une boucle While :**

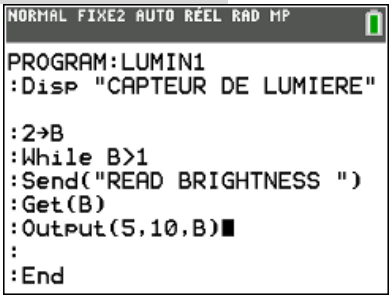
1. Avant la commande **Send(** de votre programme, ajouter l'instruction :
   **2→B**
   **While B>1 [**utiliser le menu [test] pour le symbole **>]**
   - Cette commande initialise la boucle. Tant que la condition B>1 est vraie, la boucle continue la lecture du capteur de lumière. Dès qu'elle devient fausse, c'est à dire lorsque la quantité de lumière reçue par le



capteur est insuffisante ou que celui-ci est recouvert avec la main, la boucle et le programme se terminent.

2. Le **End** de la boucle **While** doit aussi être entré, sous la commande **Get(** Ajouter enfin la commande **End** afin de marquer la fin de la boucle **While**.

**Conseil à l'enseignant :** C'est toujours une bonne idée d'écrire complètement une boucle avant le corps de celle-ci afin d'éviter des erreurs de structure.
Souvenez-vous que ***End*** n'est pas la fin du programme mais plutôt la fin de la structure de contrôle. (***If…Then…End***, ***For…End***, ***While…End***, et ***Repeat…End***). Dans des programmes plus importants, lorsqu'il y a de nombreuses instructions *End*. Le processeur identifie parfaitement à quelle structure de contrôle, la commande End fait référence. Mais c'est au programmeur de concevoir un code construit avec rigueur.




3. Ajouter la commande **Output(** après l'instruction **Get(** et avant le **End** de la boucle comme montré sur l'écran de droite, en tapant [prgm],puis en utilisant les touches de direction vers le menu **E/S** et en choisissant enfin **6: Output(**.
   - La commande **Output(** vous donne la possibilité de contrôler à l'écran l'emplacement où une information doit être affichée. La structure de la commande est : **Output(**<ligne#>, <colonne#>, <chaîne ou variable>)

Exemples :

- **Output(3,7,"HELLO")** placer la lettre "H" à la ligne 3, colonne 7 de l'écran d'accueil et le reste du mot suivra la lettre "H".
- **Output(5,10,B)** placera la valeur de la variable B au début de la ligne 5, colonne 10 de l'écran d'accueil et les autres digits à la suite.


4. Quitter l'éditeur de programme et le faire fonctionner après avoir connecté le Hub TI-Innovator.

- Vous devriez voir le message de titre en haut de l'écran et une valeur dans le centre de l'écran qui change en fonction de l'intensité lumineuse lue par le capteur. Malheureusement, les chiffres que vous voyez peuvent ne pas être corrects !

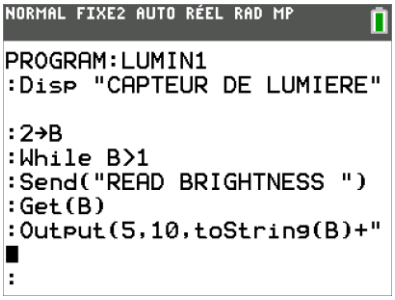
La commande **Output(** n'efface pas les digits en excès si un nombre plus court est affiché après un précédent comportant plus de décimales. Par exemple, si une valeur affichée est 1.23456 et la valeur suivante à afficher est 55, alors vous verrez 5523456. La dernière modification propose une façon élégante afin de corriger ce problème.

Pour corriger le problème de « suivi des chiffres » nous convertissons la variable B dans une chaîne et ajoutons des espaces à la fin de celle-ci pour effacer complètement la valeur précédente affichée.

**toString(** se trouve dans le menu **prgm E/S**.

La commande correcte sera finalement : **Output(5,10,toString(B)+“ ”)**

Il y a 10 espaces entre les guillemets.




Maintenant, lorsque vous exécutez le programme, vous verrez que certaines valeurs sont plus courtes que les autres parce que les espaces que nous avons ajoutés à la représentation de la chaîne B effacent les chiffres précédents.


**Concaténation** :

Le signe “**+**” utilisé dans l’instruction **Output(** ne correspond pas à une addition : il est utilisé pour « concaténer » (associer) deux chaînes de caractères. Les espaces (entre guillemets) sont ajoutées à la fin de la chaîne incluant la variable B.

**Conseil à l’enseignant :** La manipulation des chaînes de caractères n’est pas nécessaire pour ces leçons, mais il doit être clair que la connaissance des instructions permettant de travailler avec des chaînes de caractères constitue pour le programmeur une expérience indispensable.