Cette application fait usage de boucles pour entrer autant de valeurs que nécessaire et, comme prolongement, contrôle la validité des entrées et utilise des instructions **If** pour afficher un message approprié.

Objectifs :

- Étudier des instructions servant de compteurs et d'accumulateurs.
- Utiliser une boucle dans un programme pour obtenir une quantité de données indéterminée.
- Utiliser une valeur 'repère' (flag) pour terminer une boucle.

Indication : Ce projet illustre la tendance vers la complexité que l'on constate lors de l'élaboration de parties de logiciel. *L'imbrication* consiste à mettre une structure de contrôle à l'intérieur d'une autre. Comme expliqué ci-dessous, l'OS sait quel End 'marche' avec quelle instruction.

Structures Imbriquées

- *L'imbrication* (*nesting*) est la technique de programmation consistant à placer une instruction de contrôle à l'intérieur d'une autre. Le terme est dérivé de l'idée de placer une boîte en carton dans une autre afin de gagner de la place.
- Il est important de mettre la structure *toute entière complètement* dans un bloc d'une autre afin d'éviter des erreurs.
- Le listing du programme à droite montre une structure **If** à l'intérieur du bloc **Else** d'une autre structure **If**. Notez les utilisations multiples de l'instruction **End** ; la calculatrice sait quel **End** fait partie de la structure **If**.
- L'indentation est là seulement pour un souci de lisibilité. Vous n'avez pas à indenter les lignes en TI-Basic.
- Le programme tout d'abord teste si **A<0**. Si c'est le cas, le programme affiche "A est négatif !" et rien de plus. Mais quand **A** n'est pas négatif alors la partie **soulignée**, comprenant le calcul de la racine carrée, une autre instruction **If** et l'instruction **Disp**, est exécutée.
- Le programmeur place des structures **If** à l'intérieur des boucles et des boucles à l'intérieur des structures **If** pour accomplir les tâches complexes que nécessite le programme.

```
prgmSQUARE
Prompt A
If A<0
Then
   Disp "A est négatif !"
Else
   √(A)→S
   If S=partEnt(S)
   Then
     Disp " A est un carré   parfait."
   Else
     Disp "A n'est pas un carré parfait."
   End
   Disp "Sa racine carrée est",S
End
```

Résumé des trois types de boucles :

For(*var*, *début*, *fin***)**	**While** *condition vraie*	**Repeat** *until condition vraie*
End	**End**	**End**

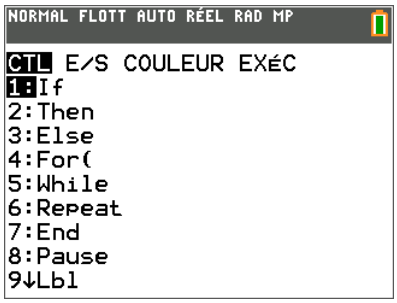
For(est utilisée pour ‘compter’ ou pour le traitement d’une suite arithmétique de valeurs (itération).
While est utilisée quand vous pourriez être en mesure de sauter complètement le corps de la boucle.
Repeat est utilisée quand vous voulez que le corps de la boucle soit exécuté au moins une fois.

À propos de End

Le mot-clé **End** est utilisé dans toutes les structures de contrôle multi-lignes :

If …	**If …**	**For(…)**	**While …**	**Repeat …**
Then	**Then**			
	Else			
End	**End**	**End**	**End**	**End**

C'est pour cela que le menu CTL de [prgm] est arrangé ainsi :



7: End vient après les six premiers éléments du menu CTL car il termine chacune de ces structures de contrôle de programmation.

L'instruction **End** peut apparaître plusieurs fois dans un programme, et l'ordinateur sait comment les **End** sont connectés à leurs structures de contrôle.

Unité 4 Application : Programme Examen

Lors d'un examen on fait la moyenne des notes obtenues par le candidat. Si la moyenne est supérieure à 10, l'élève est admis, si elle inférieure à 8, il est refusé et sinon il doit passer un oral de rattrapage.

Écrivons un programme qui permet à l'utilisateur de rentrer des notes d'examen, de déterminer la moyenne, puis d'afficher le nombre de notes entrées, la moyenne des notes et un message approprié. Les messages peuvent être: "ADMIS", "RATTRAPAGE", et "REFUSÉ".

Nous pouvons utiliser deux méthodes pour entrer un nombre indéterminé de notes :

- Méthode 1 : demander en premier la totalité des notes et utiliser une boucle **For** pour entrer les notes.
- Méthode 2 : demander les notes mais utiliser une valeur 'repère' (flag) comme par exemple -99 pour indiquer qu'il n'y a plus de note. Cette méthode utilisera une boucle **While** ou une boucle **Repeat**.

Dans les deux méthodes nous devrons garder un total cumulé des notes. Dans la Méthode 2 nous devons aussi compter les notes afin de diviser le total par le nombre de notes.

Votre programme doit afficher le nombre de notes entrées, la moyenne des notes et le message approprié :

Si la moyenne est en dessous de 8 : "REFUSÉ"

… de 8 à 10 : "RATTRAPAGE"

… au dessus de 10 : "ADMIS"

Indication : La section ci-dessous examine deux idées de programmation apparentées : une variable compteur et une variable 'accumulateur'. Insistez sur le fait que la variable à gauche de l'opérateur → est la même que celle à droite mais qu'elles représentent des valeurs différentes en raison du traitement concerné.

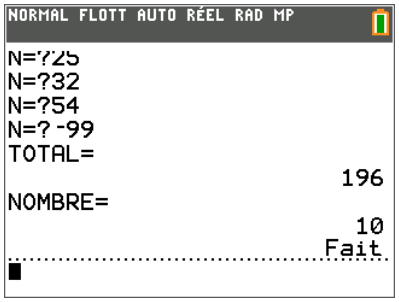
Compteurs et Accumulateurs

Une instruction comme **C+1→C** est appelée un *compteur* car elle ajoute 1 à la variable **C** chaque fois qu'elle est exécutée.

Une instruction comme **T+N→T** est appelée un *accumulateur* car il garde un total cumulé des valeurs de la variable **N**. La valeur de **N** est ajoutée à la variable **T**, puis cette somme est stockée dans la variable **T**. À la fin de la boucle, **T** va contenir le total des valeurs de **N**.

Voici un exemple qui utilise un **C**ompteur, un **A**ccumulateur et une valeur 'repère' pour garder trace des notes dans le programme :

	Notes :
	Initialisation des variables
0→C	**C** pour **C**ompteur
0→N	**N** pour **N**ote
0→T	**T** pour **T**otal
Prompt N	récupérer la première **N**ote
While N≠-99	tant que ce n'est pas -99
C+1→C	ajouter 1 au **C**ompteur
T+N→T	ajouter la **N**ote au **T**otal
Prompt N	demander une autre **N**ote
End	
Disp "Total = ",T	
Disp "Nombre = ",C	



La boucle **While**, ci-dessus, continue à compter et à accumuler les notes tant que -99 n'est pas entré. Quand -99 est entré, la boucle s'arrête et les résultats sont affichés.

> **Indication :** Remarquez les deux instructions **Prompt** dans le code ci-dessus. Le premier récupère la première note et le dernier récupère le reste des notes. Le dernier est à la *fin* de la boucle pour préparer le retour à l'instruction **While** pour tester de nouveau la valeur de N.

Prolongement

À l'intérieur de votre routine de saisie des notes, il faut une vérification pour être sûr que la valeur entrée est une note valable (entre 0 et 20) et appliquer l'action appropriée quand la note entrée n'est pas valide.

> **Indication :** Le prolongement nécessite une autre boucle auprès des instructions de saisie pour être sûr que la valeur entrée est une note valable. L'exemple de code ci-dessous stoppe simplement le programme quand une valeur non valide est entrée. On peut faire mieux…

Exemple de Réponse (rappel : l'indentation n'est là que pour une question de lisibilité) **:**

```
prgmEXAMEN
EffÉcran
0 → N
0 → C
0 → T
Input "ENTREZ UNE NOTE :",N

While N≠-99
 If N<0 or N>20
 Then
  Disp "EN DEHORS DES LIMITES !"
  Stop
 Else
  C+1→ C
  T+N → T
 End
 Input "UNE AUTRE ? (OU -99):",N
End
T/C → M

EffÉcran
Output(1,1,"NOMBRE DE NOTES : ")
Output(2,1,"TOTAL DES NOTES : ")
Output(4,1,"MOYENNE : ")
Output(1,19,C)
Output(2,19,T)
Output(4,11,M)
If M<8
Then
 Output(7,9,"REFUSÉ")
Else
 If M<10
 Then
  Output(7,9,"RATTRAPAGE")
 Else
  Output(7,9,"ADMIS")
 End
End
Pause
EffÉcran
```

Lorsque vous testez un programme essayez des cas extrêmes comme entrer -99 comme première valeur, ou entrer une valeur négative n'importe où. Si le programme génère un message d'erreur c'est que le programmeur n'a pas pris tous les scénarios en compte. Le programme ci-contre va échouer quand -99 est entré comme première valeur car il va essayer de diviser 0 par 0 ce qui est indéfini. Le calcul de la moyenne, M, devrait être inséré dans un test pour vous assurer qu'au moins une note a été saisie. Ci-dessous une solution possible :

```
If C>0
Then
 T/C → M
Else
 Disp "PAS DE NOTE ENTRÉE !"
 Stop
End
```

L'instruction **Stop** est utilisée dans ce programme, elle sert à terminer un programme *immédiatement*. Ceci peut être ajouté n'importe où dans le programme afin que l'exécution soit interrompue et que le message 'Fait' apparaisse dans l'écran de calcul.