Stage algorithmique 1 TI graphiques (82, 83, 84)	Algorithmes mystérieux

Premier algorithme mystère

Que fait ce programme ?

```
Saisir(x)
Saisir(y)
z := x
x := y
y := z
Afficher(x)
Afficher(y)
```

Remarque : noter que l'instruction **z := x** *est équivalente à* **x → z**.

Deuxième algorithme mystère

Que fait ce programme ? Le fait-il dans toutes circonstances ?

```
Saisir(x)
Saisir(y)
x := x + y
y := x – y
x := x – y
Afficher(x)
Afficher(y)
```

Troisième algorithme mystère

Cet algorithme est proposé sous forme de programme en langage machine graphique TI.
Combien de nombres vont-ils être affichés ? Quelle est la suite de nombres ainsi obtenue ?

```
1 → A
1 → B
For(I, 1, 6)
A + B → C
B → A
C → B
Disp C
End
```

Stage algorithmique 1 TI graphiques (82, 83, 84)	Algorithmes mystérieux

Après s'être familiarisé avec les algorithmes et la (ou les) syntaxe(s) correspondant aux logiciels ou calculatrices dont on dispose, l'élève devra savoir lire un petit programme. Au début, on peut proposer des petits programmes simples et courts.

Premier algorithme mystère

Que fait ce programme ?

Saisir(x) Saisir(y) z := x x := y y := z Afficher(x) Afficher(y)	Facile : permutation des valeurs contenues dans x et dans y.

Deuxième algorithme mystère

Que fait ce programme ? Le fait-il dans toutes circonstances ?

Saisir(x) Saisir(y) x := x + y y := x − y x := x − y Afficher(x) Afficher(y)	Ce programme, plus compliqué, se contente aussi de permuter les valeurs contenues dans x et dans y… sauf dans le cas où y, par exemple, est très petit devant x de telle sorte que $x + y \approx x$ …

Il est intéressant de montrer que l'on eut obtenir le même résultat avec des algorithmes différents. Pour les différencier, on peut effectivement examiner le cas où x = 5 et y = 10^{-13}. Dans le premier algorithme, il y a bien échange ; dans le second, on obtient 0 et 5.

Troisième algorithme mystère

Cet algorithme est proposé sous forme de programme en langage machine graphique TI.
Combien de nombres vont-ils être affichés ? Quelle est la suite de nombres ainsi obtenue ?

1 → A 1 → B For(I, 1, 6) A + B → C B → A C → B Disp C End	Il y a six passages (For de 1 à 6), donc six nombres affichés. Ce sont 2, 3, 5, 8, 13, 21. On obtient effectivement les termes successifs de la suite de Fibonacci. On pourra prolonger en compliquant le programme (affichage de la suite des quotients successifs, par exemple, pour obtenir des valeurs de plus en plus proches du nombre d'or).

Le professeur pourra, s'il le souhaite et si sa classe s'y prête, proposer ce dernier programme en langage TI-Nspire :

```
Prgm
Local a,b,c,i
a:=1
b:=1
For i,1,6
c:=a+b
a:=b
b:=c
Disp c
EndFor
EndPrgm
```

Stage algorithmique 1 TI graphiques (82, 83, 84)	Pour débuter (fiche 3/3) Le bloc Si … Alors … Sinon …

Le problème : *On se donne un nombre réel strictement positif N. On se propose de construire un programme qui affiche l'arrondi du nombre N à p chiffres après la virgule* (*p un nombre entier naturel compris entre* 0 *et* 10).

1. Un premier essai à la main

On se donne le nombre $N = 3{,}14159265$.

Déterminer la valeur approchée à 0,01 près de N : ……..

Quelle est celle à 0,001 près de N ? : ……..

Déterminer la règle que l'on doit utiliser pour effectuer le calcul de la valeur approchée de N à 10^{-p} près.

……

……

2. La fonction partie entière

La calculatrice dispose, dans le menu **MATH**, sous-menu **NUM**, de l'instruction **partEnt(** ou **int(**.

Calculer avec votre calculatrice : partEnt(56.251) = ……

On se propose d'utiliser cette instruction pour isoler le troisième chiffre après la virgule du nombre N de la question 1. Ce troisième chiffre est un 1.
Calculer : partEnt(1000*3.14159265) = ……
Quel nombre faut-il ôter du résultat précédent pour afficher le 1 ? ……

Comment faire apparaître ce nombre à partir de N et de la fonction **partEnt** de la calculatrice ? :

……..

Écrire maintenant, sur une seule ligne, la commande qui permettra d'isoler la quatrième chiffre après la virgule du nombre N : ……

3. La partie programmation

Compléter le programme ci-dessous en utilisant l'algorithme et les résultats établis question **2**.

Algorithme	Programme
Demander le nombre réel à arrondir Net le nombre de chiffres P après la virgule souhaité	Prompt N, P
Stocker dans la variable U la partie entière du nombre $10^{P+1}.N$	……
Stocker dans la variable V le produit du nombre 10 par la partie entière du nombre $10^{P}.N$	……
Stocker dans X le nombre U – V	……
Si X est supérieur ou égal à 5	If X≥5
Alors	Then
ajouter 1 au $P^{ième}$ chiffre après la virgule de N et calculer et stocker l'arrondi de N dans A	……
Sinon	Else
Calculer et stocker l'arrondi de N dans A	……
Fin du Si	End
Afficher le nombre arrondi A	A

Tester votre programme avec le nombre N précédent et d'autres nombres réels positifs.

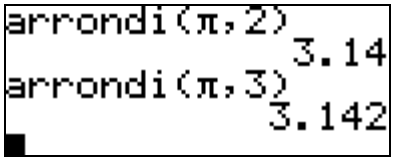
Stage algorithmique 1 TI graphiques (82, 83, 84)	Pour débuter (fiche 3/3) Le bloc Si … Alors … Sinon …

Le problème : *On se donne un nombre réel strictement positif N. On se propose de construire un programme qui affiche l'arrondi du nombre N à p chiffres après la virgule* (*p un nombre entier naturel compris entre* 0 *et* 10).

1. Un premier essai à la main

On se donne le nombre $N = 3{,}14159$. La valeur approchée à 0,01 près de N est : 3,14, celle à 0,001 près de N est : 3,142.

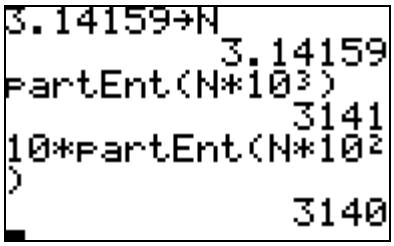
Afin de faire retrouver la règle aux élèves, il peut être intéressant de les faire manipuler la fonction **arrondi** ou **round** dont dispose la calculatrice (Menu **MATH**, sous-menu **NUM**).



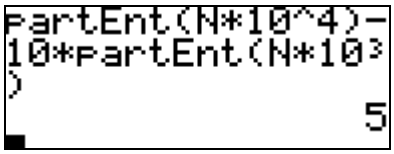
La règle :
- **Si l**e $(p+1)^{\text{ième}}$ chiffre après la virgule de N est supérieur ou égal à 5,
- **Alors** on augmente de 1 le $p^{\text{ième}}$ chiffre après la virgule de N et on supprime tous ceux qui suivent,
- **Sinon** on supprime tous les chiffres qui suivent le $p^{\text{ième}}$ chiffre après la virgule de N.

2. La fonction partie entière et l'arrondi

Pour afficher la troisième décimale de N…



En une seule ligne, pour la quatrième décimale de N…



3. La partie programmation

Algorithme	Programme

Demander le nombre réel à arrondir Net le nombre de chiffres P après la virgule souhaité
Stocker dans la variable U la partie entière du nombre 10^{P+1}.N
Stocker dans la variable V le produit du nombre 10 par la partie entière du nombre 10^{P}.N
Stocker dans X le nombre U – V
Si X est supérieur ou égal à 5
Alors
ajouter 1 au $P^{\text{ième}}$ chiffre après la virgule de N, calculer et stocker l'arrondi de N dans A
Sinon
Calculer et stocker l'arrondi de N dans A
Fin du Si
Afficher le nombre arrondi A
	`Prompt N,P`
	`partEnt(N*10^(P+1))→U`
	`10*partEnt(N*10^P)→V`
	`U-V→X`
	`If X≥5`
	`Then`
	`(partEnt(N*10^P)+1)/10^P→A`
	`Else`
	`partEnt(N*10^P)/10^P→A`
	`End`
	`A`

Stage algorithmique 1 TI graphiques (82, 83, 84)	**Pour débuter (fiche 1/3)** **La boucle Pour ….**

Le problème : *le but de cet exercice est de conjecturer une formule donnant la somme des entiers de* 1 *à n en fonction de n.*

1. Un premier essai à la main

$S_3 = 1 + 2 + 3 =$ …. $S_4 = 1 + 2 + 3 + 4 =$ …. $S_5 = 1 + 2 + 3 + 4 + 5 =$ ….

Comment calculer S_6 à partir de S_5 ? : .....……..…

Plus généralement, comment calculer S_n si on connaît S_{n-1} ? : ……..……

On peut donc de proche en proche, en partant de $S = 0$, passer à $S_1 = S + 1$, puis à $S_2 = S_1 +$ …,

puis à $S_3 = S_2 +$ …, etc. jusqu'à $S_n =$ …….

2. La programmation

Algorithme	Programme

Demander la valeur de l'entier N
Initialiser une variable S à zéro
Pour I allant de 1 à N de 1 en 1
  Remplacer S par I + S
Fin de la boucle Pour
Afficher S
	`Prompt N`
	`Ø→S`
	`For(I,1,N,1)`
	`..... →S`
	`End`
	`S`

3. Comprendre la boucle : Pour … End

Lorsque la calculatrice rencontre l'instruction : `For(I,1,N,1)` (Pour I allant de 1 à N de 1 en 1), elle crée une variable I qu'elle initialise à 1, elle teste ensuite le contenu de I afin de savoir s'il dépasse la valeur de fin de boucle (ici N).

• **Si la valeur de I est inférieure ou égale** à la valeur N de fin de boucle, on entre dans la boucle et on exécute toutes les instructions qui s'y trouvent, puis dès que le programme rencontre l'instruction de fin de boucle, on repart au début de la boucle, on ajoute 1 à la valeur de I et on teste à nouveau I et ainsi de suite..

• **Si la valeur de I est strictement supérieure** à la valeur de fin de boucle, on n'exécute pas le contenu de la boucle, mais le programme saute directement à l'instruction qui suit la fin de la boucle.

Combien de fois l'instruction Afficher « BONJOUR » est-elle exécutée dans chacune des boucles suivantes ?

Pour I allant de 1 à 1 Afficher « BONJOUR » Fin de la boucle	Pour I allant de 1 à 3 Afficher « BONJOUR » Fin de la boucle	Pour I allant de 1 à 2.5 Afficher « BONJOUR » Fin de la boucle
……	……	……..

4. L'exemple de la question 2 détaillé pas à pas

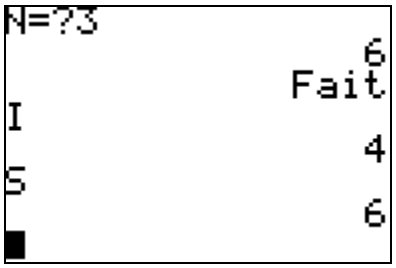
Dans le programme de la question 2, trois variables sont utilisées.
On choisit, pour illustrer notre exemple, de fixer N à 3.
Lire alors le tableau suivant afin de bien comprendre le fonctionnement de la boucle Pour.

	N	I	S	Commendataires
Début du programme	3		0	On saisit la valeur de N. On initialise S à zéro.
On rencontre la boucle **Pour** (une première fois)	3	1	1	I est initialisé à 1, I est comparé à N. On entre dans la boucle, on ajoute 1 à S qui prend la valeur 1. Le programme rencontre l'instruction de fin de boucle et reprend au début de la boucle.
On rencontre la boucle **Pour** (une deuxième fois)	3	2	3	I augmente de 1 (I = 2), I est comparé à N. On entre dans la boucle, on ajoute 2 à S qui prend la valeur 3. Le programme rencontre l'instruction de fin de boucle et reprend au début de la boucle.
On rencontre la boucle **Pour** (une troisième fois)	3	3	6	I augmente de 1 (I = 3), I est comparé à N. On entre dans la boucle, on ajoute 3 à S qui prend la valeur 6. Le programme rencontre l'instruction de fin de boucle et reprend au début de la boucle.
On rencontre la boucle **Pour** (une quatrième fois)	3	4	6	I augmente de 1 (I = 4), I est comparé à N. La valeur de I étant strictement supérieure à N, on sort de la boucle Pour. On affiche alors le contenu de S.

On peut observer l'écran ci-contre.

Le programme de la question 2 a été exécuté avec la valeur N = 3.

Une fois le programme exécuté, on a demandé d'afficher la valeur des variables I et S en tapant leur nom au clavier et en appuyant sur la touche (entrer).



5. La conjecture

Saisir le programme de la question 2 dans votre calculatrice et l'exécuter pour les valeurs 4, 7, 12 et 37 de N.

Compléter alors le tableau ci-contre.

N	S	2S/N
4		
7		
12		
37		

Quelle formule peut-on avancer pour l'expression de S en fonction de N ?

S = ……

Vérifier cette conjecture pour différentes valeurs de N à l'aide du programme.
Attention, ceci ne constitue pas une preuve et cette formule doit maintenant être démontrée….

Stage algorithmique 1 TI graphiques (82, 83, 84)	Pour débuter (fiche 1/3) La boucle Pour ….

Le problème : *Le but de cet exercice est de conjecturer une formule donnant la somme des entiers de* 1 *à n en fonction de n.*

1. Un premier essai à la main

L'idée est de faire découvrir aux élèves la règle de récurrence qui permet de définir la boucle qui sera répétée n fois : $S_n = S_{n-1} + n$.

2. La programmation

Algorithme	Programme

Demander la valeur de l'entier N
Initialiser une variable S à zéro
Pour I allant de 1 à N de 1 en 1
  Remplacer S par **I + S**
Fin de la boucle Pour
Afficher S
	`Prompt N`
	`Ø→S`
	`For(I,1,N,1)`
	`I+S→S`
	`End`
	`S`

3. Comment saisir le programme dans la calculatrice ?

Instructions saisies au clavier	Écran	Commentaires
PRGM ◄ ENTER	PROGRAMME Nom=▮	On ouvre l'éditeur de programme pour enregistrer celui-ci (sous-menu **NOUV**).
B O U C L E ALPHA 1 ENTER	PROGRAM:BOUCLE1 :▮	On donne un nom au programme (8 caractères maximum, on débute par une lettre).
PRGM ► 2 ALPHA LOG ENTER	PROGRAM:BOUCLE1 :Prompt N :▮	L'instruction **Prompt** affiche le message N = ? et stocke la valeur saisie dans la variable N de la calculatrice.
0 STO▶ ALPHA LN ENTER	PROGRAM:BOUCLE1 :Prompt N :0→S :▮	On initialise la variable S à 0.
PRGM 4 ALPHA x^2 , 1 , ALPHA LOG , 1) ENTER ALPHA x^2 + ALPHA LN STO▶ ALPHA LN ENTER	PROGRAM:BOUCLE1 :Prompt N :0→S :For(I,1,N,1) :I+S→S :▮	On sélectionne l'instruction **For(**, début de la boucle. On ajoute la valeur de I au contenu de S que l'on stocke dans S.

Suite du tableau page suivante

Instructions saisies au clavier	Écran	Commentaires
PRGM 7 ENTER ALPHA LN	PROGRAM:BOUCLE1 :Prompt N :0→S :For(I,1,N,1) :I+S→S :End :S	On saisit l'instruction de fin de boucle (**End**). On demande l'affichage du contenu de la variable S.

On quitte le mode d'enregistrement du programme en appuyant sur les touches 2nd MODE ce qui nous renvoie dans l'écran de calcul habituel.

4. Comment exécuter le programme précédent ?

PRGM ▼ ▼ ....ENTER	prgmBOUCLE1	On ouvre l'éditeur de programme. Dans le sous-menu **EXEC**, on descend à l'aide des flèches jusqu'à positionner le curseur sur le nom de notre programme. On appuie sur la touche ENTER, le nom du programme est alors affiché à l'écran.
ENTER	prgmBOUCLE1 N=?	On lance l'exécution du programme qui invite à saisir la première valeur de N.
4 ENTER	prgmBOUCLE1 N=?4 10	On saisit 4 au clavier, le programme s'exécute et affiche la valeur de la somme S = 1 + 2 +3 + 4 = 10.
ENTER 7 ENTER	prgmBOUCLE1 N=?4 10 N=?7 28	On relance le programme pour une autre valeur de N.

5. Comment corriger un programme qui comporte une erreur de syntaxe ?

PRGM ▼ ▼ ....ENTER ENTER	ERR:SYNTAXE 1:Quitter 2:Voir	On a lancé un programme qui comporte une erreur, la calculatrice détecte l'erreur et propose, soit de quitter le programme en cours, soit d'afficher le contenu du programme défectueux.
2	PROGRAM:BOUCLE1 :Prompt N :0→S :For(I,,N,1) :I+S→S :End	On choisit **Voir**. Le curseur s'est positionné sur l'erreur commise. Ici, il manque le 1 du début de la boucle Pour.

Suite du tableau page suivante

2nd DEL 1	PROGRAM:BOUCLE1 :Prompt N :0→S :For(I,1_N,1) :I+S→S :End :S	On appuie sur les touches 2nd DEL afin d'insérer le chiffre 1 manquant. On saisit 1 au clavier.
2nd MODE ENTER 4 ENTER	N=?4 N=?4 10	On quitte le mode d'enregistrement de programme par 2nd MODE. On relance le programme pour s'assurer que l'erreur a bien été corrigée.

6. Comment modifier un programme existant ?

On souhaite par exemple ici afficher, après la valeur de S, la valeur de l'expression $\frac{N(N+1)}{2}$ afin de la comparer à S.

PRGM ▶ ▼ ▼... ▼ ENTER	PROGRAM:BOUCLE1 :Prompt N :0→S :For(I,1,N,1) :I+S→S :End :S	On ouvre l'éditeur de programme. Dans le sous-menu **EDIT**, à l'aide des flèches on place le curseur dans la liste qui s'affiche sur le programme BOUCLE1, on appuie sur ENTER.
▼ ▼ ▼ ▼ ▼ PRGM ▶ 3 ALPHA + ALPHA LN 2nd MATH 1 ALPHA + , ALPHA LN ENTER PRGM ▶ 3 ALPHA + ALPHA LOG (ALPHA LOG + 1) ÷ 2 2nd MATH 1 ALPHA + , ALPHA LOG (ALPHA LOG + 1) ÷ 2	PROGRAM:BOUCLE1 :For(I,1,N,1) :I+S→S :End :Disp "S=",S :Disp "N(N+1)/2= ",N(N+1)/2	On descend jusqu'à la fin du programme, à l'aide des flèches (on se place sur le S de la dernière ligne du programme). L'instruction **Disp** permet d'afficher du texte et le contenu de variables.
2nd MODE PRGM ▼...▼ ENTER ENTER 4 ENTER	prgmBOUCLE1 N=?4 S= 10 N(N+1)/2= 10 Fait	On relance le programme ainsi modifié.

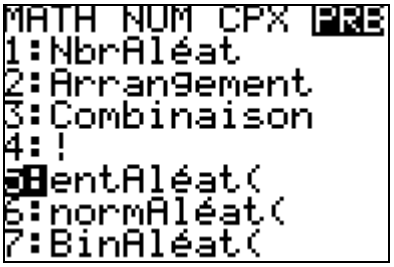
N.B. Il existe des versions de ce fichier BouclePour_prof et du fichier BouclePour_eleve correspondant aux touches francisées des calculatrices *.fr* : par exemple (entrer) au lieu de ENTER.
Voir fichiers BouclePour_prof_fr et BouclePour_eleve_fr.

Stage algorithmique 1 TI graphiques (82, 83, 84)	Pour débuter (fiche 2/3) La boucle Tant que…

Le problème : *On lance un dé cubique parfait autant de fois qu'il le faut pour obtenir un trois. Combien de fois faut-il lancer le dé en moyenne pour atteindre le premier* 3 ?

1. Un premier essai à la main

On trouve dans la menu **MATH**, sous menu **PRB** (probabilités) en cinquième position l'instruction **entAléat(** qui permet de simuler notre dé à six faces.

MATH ◄ ▼ ▼ ▼ ▼	
ENTER 1 , 6) ENTER ENTER ENTER ENTER ENTER ENTER	

Dans l'exemple ci-dessus, il a fallu 6 lancers avant d'obtenir le premier trois.
Faire maintenant soi-même l'expérience avec sa propre calculatrice et compléter le tableau suivant.

Numéro de l'essai	1	2	3
Nombre de fois			

2. La programmation.

Algorithme	Programme
Initialiser une variable compteur C à ……	.....
Tant que le dé sort un nombre différent de 3	While entAléat(1,6)≠3
Ajouter 1 au compteur C	.....
Fin de la boucle Tant que	End
Afficher C	….

Compléter l'algorithme et le programme précédent. On donnera le nom COMBIEN au programme.

N.B. L'instruction ≠ se trouve dans le menu **TEST**, accessible par 2nd MATH.

En particulier, que se passe t-il si on fait un 3 lors du premier lancer ?

……

Que faut-il placer dans la variable C au tout début du programme ? …….

Une fois le programme enregistré dans la calculatrice, l'exécuter 10 fois de suite et faire la moyenne des résultats obtenus.

Moyenne des résultats : ……..

Comparer avec les autres résultats des camarades de la classe. Le nombre d'essais paraît-il suffisant pour donner une réponse à la question posée ? ……..

3. Un nouveau programme pour estimer la moyenne

N désigne ici le nombre d'essais que l'on souhaite réaliser.
S sera la variable qui va totaliser les résultats pour tous les essais.

En s'aidant de l'algorithme, compléter le programme suivant que l'on appellera NFOIS.

Algorithme	Programme
Demander la valeur de l'entier N	.....
Initialiser une variable S à zéro	.....
Pour J allant de 1 à N	.....
Exécuter le programme COMBIEN	`prgmCOMBIEN`
Augmenter S du contenu de C	.....
Fin de la boucle Pour	.....
Afficher la moyenne des résultats	.....

N.B. Pour saisir l'instruction **prgmCOMBIEN** lors de l'enregistrement du programme, appuyer sur la touche [PRGM], choisir **EXEC**, descendre avec les flèches jusqu'à rencontrer le programme COMBIEN, puis appuyer sur [ENTER].

Compléter alors le tableau suivant pour les valeurs de N proposées.

Valeur de N	20	50	100	200
Moyenne				

Comment utiliser les résultats des autres élèves de la classe pour améliorer notre estimation de la moyenne du nombre d'essais ?

……..
Quelle valeur obtient-on alors pour la moyenne ? ……..

Stage algorithmique 1 TI graphiques (82, 83, 84)	Pour débuter (fiche 2/3) La boucle Tant que ….

Le problème : *On lance un dé cubique parfait autant de fois qu'il le faut pour obtenir un trois. Combien de fois faut-il lancer le dé en moyenne pour atteindre le premier* 3 ?

1. Un premier essai à la main

Le fait de manipuler sans l'aide de la programmation permet aux élèves de prendre connaissance de l'instruction **entAléat (1,6)** qui simule le lancer du dé et de percevoir, sur quelques essais, la variabilité du nombre de lancers nécessaires pour retourner un premier 3.

2. La programmation

a) Le fonctionnement de la boucle Tant que

While *condition*
Action(s) à effectuer si la condition est vérifiée
End
Action(s) à effectuer lorsque la condition n'est pas vérifiée.

Lorsque le programme rencontre l'instruction While, il teste la condition,
- **si elle est vraie**, il entre dans la boucle en exécutant son contenu et recommence tant que la condition reste vraie.
- **si la condition n'est pas vérifiée**, il ne rentre pas dans la boucle et saute à l'instruction qui suit le mot End.

b) Retour à notre problème

Lorsque l'on obtient un trois au premier lancer, on ne rentre pas dans la boucle Tant que, le programme saute alors immédiatement à l'instruction qui suit le « End » et affiche la valeur de C.
On doit donc stocker 1 dans la variable C au début du programme.

Algorithme	Programme

Initialiser une variable compteur C à **1**
Tant que le dé sort un nombre différent de 3
Ajouter 1 au compteur C
Fin de la boucle Tant que
Afficher C
	`1→C`
	`While entAléat(1,6)≠3`
	`1+C→C`
	`End`
	`C`

3. Le programme NFOIS

Ce nouveau programme va appeler le programme précédent n fois.
Le programme COMBIEN peut donc être considéré comme un sous programme du programme NFOIS.

Algorithme	Programme

Demander la valeur de l'entier N
Initialiser une variable S à zéro
Pour J allant de 1 à N
Exécuter le programme COMBIEN
Augmenter S du contenu de C
Fin de la boucle Pour
Afficher la moyenne des résultats
	`Prompt N`
	`Ø→S`
	`For(J,1,N)`
	`prgmCOMBIEN`
	`C+S→S`
	`End`
	`S/N`

Voici une liste de réponses obtenues en lançant le programme NFOIS :

Valeur de N	20	50	100	200
Moyenne	5,2	5,92	5,43	5,825

Un peu de théorie (entre nous)

La variable aléatoire X qui prend pour valeur le nombre de lancers nécessaires à l'obtention du premier trois suit une loi géométrique de paramètre 1/6.
Son espérance mathématique est 6.
La loi des grands nombres nous permet de penser que, plus le nombre d'essais devient grand, plus on a de chance de s'approcher de 6.

4. Comment supprimer un programme dans la calculatrice ?

On souhaite par exemple supprimer le programme NFOIS réalisé précédemment[1].

Instructions	Écran	Commentaires
2nd + ▼	MEMOIRE 1:A Propos… 2:Gest Mem/Sup… 3:Efface entrées 4:EffToutListes 5:Archive 6:DéSarchive 7↓Réinitialise	On a accès à tout ce qui est géré par la mémoire de la calculatrice
2 7		On sélectionne l'instruction 2, gestion /suppression de la mémoire, puis l'instruction 7 qui permet d'accéder à la liste de tous les programmes stockés dans la calculatrice.
▼ ▼ .... ▼	RAM LIBRE 13793 ARC LIBRE 242255 *MOYEN 53 *MU 44 *NCUBE 49 *NDES 56 ▸ NFOIS 50 NFOISV1 63	Ces programmes sont rangés par ordre alphabétique, il suffit donc de descendre à l'aide des flèches jusqu'à rencontrer le programme NFOIS.
DEL	Confirmation ? 1:Non 2:Oui	On appuie alors sur la touche DEL, la calculatrice demande confirmation. Attention, si l'on choisit la seconde option, le programme sera effacé et il n'est alors plus possible de revenir en arrière.
2	RAM LIBRE 13843 ARC LIBRE 242255 *MU 44 *NCUBE 49 *NDES 56 ▸ NFOISV1 63 NFOISV2 52 *NORM 71	On peut constater que le programme ne figure plus dans la liste.

[1] Sur la TI-82 Stats.fr, les écrans et la procédure sont légèrement différents. Voir page suivante.

Comment supprimer un programme dans la calculatrice TI-82 Stats.fr ?

Même séquence de touches

L'écran MÉMOIRE comporte en ligne **2** l'option : **Efface…**, que l'on choisit.
Sélectionner ensuite l'instruction **7 : Prgm…**
Descendre jusqu'à ce que le curseur ▸ soit en face de NFOIS.

Appuyer sur [ENTER] pour supprimer le programme.
Aucune confirmation n'est demandée. Il n'est plus possible de revenir en arrière.

Attention !

_____

N.B. Il existe des versions de ce fichier BoucleTantQue_prof et du fichier BoucleTantQue_eleve correspondant aux touches francisées des calculatrices *.fr* : par exemple [entrer] au lieu de [ENTER].
Voir fichiers BoucleTantQue_prof_fr et BoucleTantQue_eleve_fr.

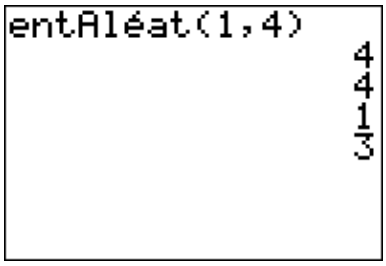
Stage algorithmique 1 TI graphiques (82, 83, 84)	Tir aux canards dans la classe

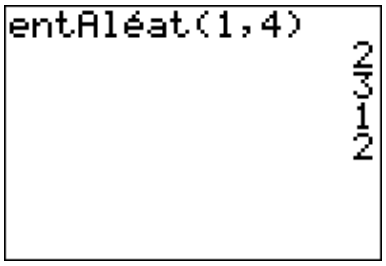
L'énoncé : *Dix chasseurs, tous excellents (ils ne ratent jamais leur cible), tirent simultanément sur dix canards sauvages, chaque chasseur choisissant son canard au hasard.*
On s'intéresse au nombre de survivants,…chez les canards bien sûr !

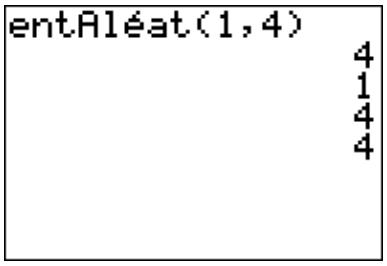
Partie 1 : Une simulation dans la classe avec quatre chasseurs et quatre canards

On peut envisager de faire jouer à 4 élèves le rôle des chasseurs et à 4 autres le rôle moins prestigieux des quatre canards.

Voici par exemple, quelques affichages obtenus pour trois tirs de nos quatre chasseurs.







Le tableau rempli avec 5 tirs et le calcul de la moyenne :

Numéro du canard	Tir n° 1	Tir n° 2	Tir n° 3	Tir n° 4	Tir n° 5
1	0	0	0	0	0
2	1	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
Nombre de survivants	1	1	2	1	0

La moyenne du nombre de survivants sur ces 5 tirs est ici de : (1+1+2+1)/5 = 1.

Partie 2 : L'algorithme et sa programmation

Compléter le programme ci-dessous après avoir analysé l'algorithme.

Algorithme	Programmation TI 82, 83, 84

Construire une liste L1 de dix chiffres 1
Pour I allant de 1 à 10
  Choisir un nombre entier X au hasard entre 1 et 10
  Remplacer le $X^{ème}$ chiffre 1 par un zéro dans la liste
Fin de la boucle pour
Afficher la somme des termes de la liste
	`suite(1,A,1,1Ø)→L1`
	`For(I,1,1Ø)`
	`entAleat(1,1Ø)→X`
	`Ø→L1(X)`
	`End`
	`Somme(L1)`

Voici la liste des instructions utilisées pour saisir le programme dans la calculatrice.

Les instructions saisies au clavier	Commentaires
PRGM ◄ ENTER	On passe en mode enregistrement de programme.
PRGM MATH LOG MATH × x^{-1} LN ENTER	On saisit le titre du programme (la calculatrice est alors automatiquement en mode alphanumérique)
2nd STAT ► 5 1 , ALPHA MATH , 1 , 1 0) STO► 2nd 1 ENTER	On fabrique une liste de dix chiffres 1 qui sera stockée dans la liste L_1.
PRGM 4 ALPHA x^2 , 1 , 1 0) ENTER	On débute la boucle Pour
MATH ◄ 5 1 , 1 0) STO► X,T,Θ,n ENTER	On choisit un nombre entier au hasard entre 1 et 10
0 STO► 2nd 1 (X,T,Θ,n) ENTER	On remplace par 0 le terme de rang X de la liste
PRGM 7 ENTER	Fin de la boucle Pour
2nd STAT ► ► 5 2nd 1)	On calcule la somme des termes de la liste L_1
2nd MODE	On quitte le mode d'enregistrement du programme.
ENTER ▼....▼ ENTER	On lance le programme après l'avoir sélectionné dans la liste des programmes déjà enregistrés.
ENTER ENTER ENTER ENTER	On appuie sur la touche ENTER autant de fois que l'on souhaite exécuter le programme.

Partie 3 : On répète N fois l'expérience précédente

Algorithme	Programme

Demander N
Créer une liste vierge de N termes
Pour J allant de 1 à N
Exécuter le programme CANARDS
Stocker le résultat trouvé par le programme au rang J de la liste précédemment créée.
Fin de la boucle Pour
Afficher la moyenne des termes de la liste.
	`Prompt N`
	`Suite(Ø,A,1,N)→L2`
	`For(J,1,N)`
	`prgmCANARDS`
	`Rep→L2(J)`
	
	`End`
	`Disp moyenne(L2)`

Pour saisir le programme CANARDS, appuyer sur la touche PRGM puis descendre dans la liste des programmes qui s'affiche jusqu'à placer le curseur sur la ligne comportant le nom CANARDS et appuyer sur ENTER.
La fonction moyenne peut être récupérée par 2nd STAT ► ► 3 ENTER.

Le second programme :

Algorithme	Programme

Demander N
Initialiser une variable S à 0
Pour J allant de 1 à N
Exécuter le programme CANARDS
Ajouter le résultat trouvé par le programme au contenu de S.
Fin de la boucle Pour
Afficher le nombre S/N.
	`Prompt N`
	`Ø→S`
	`For(J,1,N)`
	`prgmCANARDS`
	`Rep+S→S`
	`End`
	`Disp S/N`

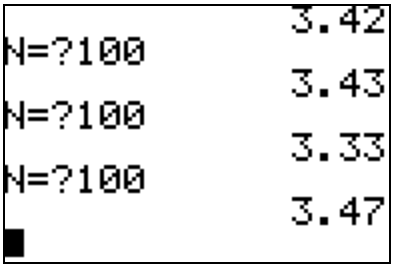
Quelles sont les différences qui existent entre ces deux programmes ?

Le premier programme permet de mémoriser chacune des réponses données par le programme CANARDS et de les visualiser si nécessaire après exécution du programme : [2nd] [2] [ENTER] ou [STAT] [1].

Le second programme utilise une variable S, initialisée tout d'abord à zéro, qui va comptabiliser au fur et à mesure des passages dans la boucle pour chacun des N tirs le nombre de survivants.
Il suffit alors, une fois sorti de la boucle, de récupérer son contenu pour calculer la moyenne du nombre de survivants pour N tirs.

Voici quelques résultats obtenus en utilisant le programme 2 de la partie 3.

Valeur de N	10	20	50	100
Moyenne du nombre de survivants.	3,3	3,4	3,6	3,42



La théorie

Un résultat théorique (entre nous) qui vient confirmer les simulations précédentes.

La probabilité qu'un canard donné survive au massacre peut se calculer aisément : en effet, il survit si chacun des 10 chasseurs choisit un autre canard que lui.
La probabilité que le chasseur numéro 1 choisisse un autre canard que lui est 0,9.
Les différents choix de nos 10 chasseurs sont indépendants les uns des autres, la probabilité qu'un canard donné survive est donc : $(0,9)^{10}$.
Or il y a 10 canards, le nombre moyen de survivants est donc de $10.(0,9)^{10} \approx 3,4867\ldots$.

N.B. 1 L'activité présentée est un peu longue pour être traitée par l'ensemble des élèves en 55 minutes. Si les meilleurs la terminent dans le temps imparti, tous les élèves seront invités à la finir à la maison.

N.B. 2 Il existe des versions de ce fichier Canards_prof et du fichier Canards_eleve correspondant aux touches francisées des calculatrices *.fr* : par exemple (entrer) au lieu de [ENTER].
Voir fichiers Canards_prof_fr et Canards_eleve_fr.

Stage algorithmique 1 TI graphiques (82, 83, 84)	Méthode de Héron d'Alexandrie

- **Héron d'Alexandrie** (I^er siècle apr. J.-C.) est un des premiers mathématiciens à mettre l'accent sur l'idée fructueuse d'approximations successives. Le texte ci-dessous est le premier écrit connu sur un « algorithme » sans doute très ancien ; il est extrait d'*Histoire d'algorithmes* (éditions Belin), page 231.

> *Puisque 720 n'a pas de côté rationnel, nous extrairons le côté avec une très petite différence de la façon suivante. Comme le premier nombre carré plus grand que 720 est 729 qui a pour côté 27, divise 720 par 27 ; cela fait 26 et $\frac{2}{3}$; ajoute 27, cela fait 53 $\frac{2}{3}$; prends-en la moitié, cela fait 26 $\frac{1}{2}\frac{1}{3}$ multiplié par lui-même donne 720 $\frac{1}{36}$, de sorte que la différence (sur les carrés) est $\frac{1}{36}$. Si nous voulons rendre cette différence inférieure encore à $\frac{1}{36}$, nous mettrons 720 $\frac{1}{36}$ trouvé tout à l'heure à la place de 729 et en procédant de la même façon, nous trouverons que la différence sur les carrés est beaucoup plus petite que $\frac{1}{36}$.*
>
> *Les métriques*

Sans qu'aucune justification ne soit donnée, Héron décrit dans ce texte un procédé de calcul, que l'on peut réitérer autant de fois que l'on veut – un algorithme donc –, permettant d'obtenir des valeurs approchées de $\sqrt{720}$ de plus en plus précises. On remarque aussi dans l'écriture des nombres l'utilisation des *quantièmes* (c'est-à-dire les inverses de l'unité et la fraction 2/3), à la manière égyptienne : par exemple, $26\frac{1}{2}\frac{1}{3} = 26 + \frac{1}{2} + \frac{1}{3}$.

- **De valeurs approchées en valeurs approchées…**

Comment peut-on justifier simplement un tel procédé ?

On part de 27, **valeur approchée par excès** de $\sqrt{720}$, assez facile à déterminer puisque $27^2 = 729 > 720$.

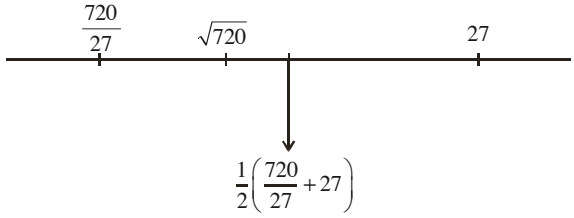
À partir de cette valeur approchée par excès, on fabrique une autre valeur approchée de $\sqrt{720}$, **par défaut cette fois** : $\frac{720}{27}$.

La raison en est simple : le produit $27\times\frac{720}{27}$ étant égal à 720, comme le premier facteur est strictement supérieur à $\sqrt{720}$, l'autre est nécessairement strictement inférieur à cette même racine.

L'idée est alors de prendre la **moyenne arithmétique** de ces deux valeurs soit : $\frac{1}{2}\left(27+\frac{720}{27}\right)$.

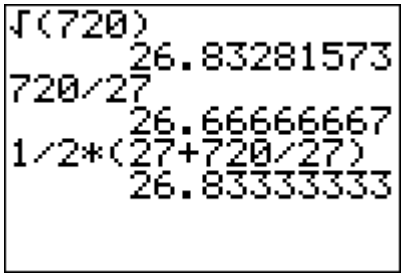
Apparaît alors le calcul proposé par Héron. Il se trouve que cette moyenne est de nouveau une valeur approchée **par excès** de $\sqrt{720}$. En d'autres termes, on a : $\sqrt{720} < \frac{1}{2}\left(27+\frac{720}{27}\right) < 27$.

En effet, la moyenne *arithmétique* de 27 et $\frac{720}{27}$ est plus grande[1] que leur moyenne *géométrique* $\sqrt{27\times\frac{720}{27}}$ ... qui n'est autre que $\sqrt{720}$… et plus petite que 27 (qui est le plus grand des deux nombres).



[1] Comparaison classique et générale entre ces deux moyennes... et fort bien connue à l'époque de Héron !

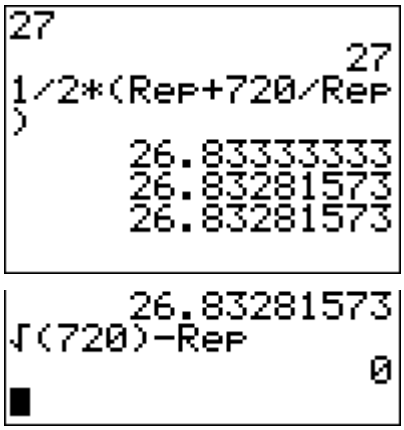
En conclusion, à l'issue de ce calcul, on a obtenu une nouvelle valeur approchée de $\sqrt{720}$ plus précise que la première dont on est parti.
La calculatrice confirme bien sûr les remarques précédentes, en masquant toutefois la compréhension du phénomène :



Rien n'empêche, comme le suggère Héron, de recommencer avec cette valeur ce que l'on a fait précédemment avec 27, et de poursuivre aussi loin que nécessaire.

- Et c'est dans cette répétition d'un *même* type de calcul, répétition déjà suggérée par Héron, qu'apparaît pleinement la notion d'algorithme. Ce que l'on a fait avec 27, on le refait avec 26,833333, et ainsi de suite… en obtenant à chaque fois des valeurs approchées de plus en plus proche de $\sqrt{720}$.

Une première approche, la plus simple pour visualiser le processus, consiste à utiliser la touche Rép de la calculatrice :



Cela conduit, comme prévu, à une excellente valeur de $\sqrt{720}$, aussi bonne en tout cas que celle renvoyée par la calculatrice.

- Automatisons la démarche, ce qui revient à écrire un *programme* à la calculatrice.

Tout d'abord, ce qui a été dit précédemment s'applique au calcul non seulement de $\sqrt{720}$ mais aussi à celui de $\sqrt{a}$ où a est un réel strictement positif.

En effet, si on part d'une valeur par excès b de $\sqrt{a}$, on peut montrer comme précédemment que $\frac{1}{2}\left(b+\frac{a}{b}\right)$ est aussi une valeur approchée par excès de ce nombre, plus petite que le b du départ. Nous admettrons que les nombres obtenus « s'approchent de plus en plus près » de $\sqrt{a}$, dans un sens que l'on ne peut pas préciser en classe de seconde.

Le programme que l'on cherche à écrire doit d'abord demander avec quelle valeur de a on souhaite travailler. Reste aussi le problème de savoir avec quelle valeur b on initialise le processus : cette valeur peut être dans un premier temps fournie par l'utilisateur.

Comme le principe de cette démarche est de réitérer un calcul, l'algorithme se réduit essentiellement à une boucle.
Boucle for ou boucle while ? Tout dépend de la façon dont on veut en sortir... Puisqu'il s'agit de répéter un calcul :

doit-on le répéter un certain nombre de fois fixé à l'avance ? La boucle for s'impose.
doit-on le répéter jusqu'à ce qu'une certaine condition soit réalisée ? C'est une boucle while.

C'est ce dernier choix que nous ferons, en demandant à ce que le calcul s'arrête quand que la différence de deux termes consécutifs est inférieure à 10^{-12}.

Le programme qui suit tient compte de ces remarques :

```
PROGRAM:HERON
:Input "RACINE D
E ",A
:Input "VALEUR I
NITIALE ",B
:1/2*(B+A/B)→C
:While abs(C-B)>
10^-12
:C→B
:1/2*(B+A/B)→C
:End
:Disp C
```

Les résultats sont conformes à ce que l'on attend, avec des performances identiques à la touche racine carrée de la calculatrice :

```
RACINE DE 720
VALEUR INITIALE
27
     26.83281573
            Fait
√(720)
     26.83281573
```

```
RACINE DE 2
VALEUR INITIALE
2
     1.414213562
            Fait
√(2)
     1.414213562
```

- Évitons maintenant la saisie de la valeur de *b* qui amorce le calcul. Il est clair qu'une valeur approchée par excès de $\sqrt{a}$, lorsque $a > 1$, est tout simplement… *a* lui-même. D'où le programme suivant :

```
PROGRAM:HERON2
:Input "RACINE D
E ",A
:A→B
:1/2*(B+A/B)→C
:While abs(B-C)>
10^-12
:C→B
:1/2*(B+A/B)→C
:End
:Disp C
:
```

Ce programme fonctionne en fait aussi pour une valeur de *a* comprise entre 0 et 1 (pourquoi ?).

ANNEXE

La méthode vue dans le cas de 720 par Héron d'Alexandrie se généralise sans peine à n'importe quel nombre réel a supérieur à 1. Dans un langage moderne, on raisonne avec des suites. En posant $g(x)=\frac{1}{2}\left(x+\frac{a}{x}\right)$, pour $x \neq 0$, étudions la suite[2] (u_n) définie par :

> sa valeur initiale $u_0 > \sqrt{a}$;
> et la relation de récurrence $u_{n+1} = g(u_n)$ pour $n \geq 0$.

La suite (u_n) est minorée par $\sqrt{a}$

Montrons que pour tout x réel strictement positif différent de $\sqrt{a}$, $\frac{1}{2}\left(x+\frac{a}{x}\right) \geq \sqrt{a}$ [3].

En effet, l'inégalité précédente équivaut à :…

$$x+\frac{a}{x} \geq 2\sqrt{a} \qquad x+\frac{a}{x}-2\sqrt{a} \geq 0 \qquad \left(\sqrt{x}-\sqrt{\frac{a}{x}}\right)^2 \geq 0$$

Un carré étant toujours positif, cette dernière inégalité est évidente.
Elle est d'ailleurs stricte, si on suppose $x \neq \sqrt{a}$.
Le premier terme de la suite est bien supérieur à $\sqrt{a}$; d'après le calcul précédent, il en est de même de tous les autres.

Remarquons que si le premier terme est choisi quelconque[4], éventuellement inférieur strictement à $\sqrt{a}$, les termes qui suivent seront tous strictement supérieur à $\sqrt{a}$. Ceci garantit que u_n ne s'annule jamais, et donc que la suite est bien définie pour toute valeur de n.

La suite (u_n) est strictement décroissante

Ce qui équivaut à démontrer que pour tout entier naturel n, $u_{n+1} < u_n$. Plusieurs méthodes sont possibles.

On sait que, pour tout n, $u_n > \sqrt{a}$. Donc, $\frac{1}{u_n} < \frac{1}{\sqrt{a}}$ et $\frac{a}{u_n} < \frac{a}{\sqrt{a}} = \sqrt{a}$. Donc $u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right)$ est la moyenne arithmétique de deux nombres distincts u_n et $\frac{a}{u_n}$: u_{n+1} est donc à ce titre inférieur strictement au plus grand de ces deux nombres, à savoir u_n.

Une autre méthode consiste à remarquer que, pour tout n entier naturel, $u_n > \sqrt{a}$ soit $u_n^2 > a$ et en divisant par $u_n > 0$, $u_n > \frac{a}{u_n}$. Par suite, $u_{n+1} = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right) < \frac{1}{2}(u_n + u_n) = u_n$.

Enfin, une troisième méthode plus classique, qui consiste à à étudier le signe de $u_{n+1} - u_n$.

$$u_{n+1} - u_n = \frac{1}{2}\left(u_n + \frac{a}{u_n}\right) - u_n = \frac{u_n^2 + a - 2u_n^2}{2u_n} = \frac{a - u_n^2}{2u_n} = \frac{\left(\sqrt{a} - u_n\right)\left(\sqrt{a} + u_n\right)}{2u_n}.$$

On peut alors conclure immédiatement.
Bref la suite (u_n) est décroissante.

---

[2] Sous réserve qu'elle soit définie...

[3] Là encore, l'inégalité est vraie car la moyenne arithmétique de deux nombres positifs est toujours plus grande que leur moyenne géométrique...

[4] Mais positif...

La suite est donc convergente et converge vers une limite λ.

Il est clair que cette limite vérifie $\lambda = g(\lambda)$, dont les solutions sont $\sqrt{a}$ et $-\sqrt{a}$: comme la suite est positive, elle ne peut converger que vers $\sqrt{a}$.

Précisons la convergence...

On a pour tout n :

$$0 \le u_n - \sqrt{a} = \frac{1}{2}\left(u_{n-1} + \frac{a}{u_{n-1}}\right) - \sqrt{a} = \frac{{u_{n-1}}^2 + a - 2u_{n-1}\sqrt{a}}{2u_{n-1}} = \frac{(u_{n-1} - \sqrt{a})^2}{2u_{n-1}} \le \frac{(u_{n-1} - \sqrt{a})^2}{2\sqrt{a}}.$$

En terme de vitesse de convergence, on peut en déduire que $0 \le \lim\limits_{n\to+\infty} \frac{u_{n+1} - \sqrt{a}}{u_n - \sqrt{a}} \le \lim\limits_{n\to+\infty} \frac{u_n - \sqrt{a}}{2} = 0$, donc que $\lim\limits_{n\to+\infty} \frac{u_{n+1} - \sqrt{a}}{u_n - \sqrt{a}} = 0$, ce qui caractérise les suites à convergence rapide.

Une telle majoration caractérise une convergence quadratique : concrètement, le nombre de chiffres significatifs double à chaque itération.

En effet si l'on connaît u_k à avec une erreur inférieure à 10^{-p}, alors :

$$0 < u_{k+1} - \sqrt{a} < \frac{\left(u_k - \sqrt{a}\right)^2}{2\sqrt{a}} < \frac{10^{-2p}}{2\sqrt{a}} < 10^{-2p}$$

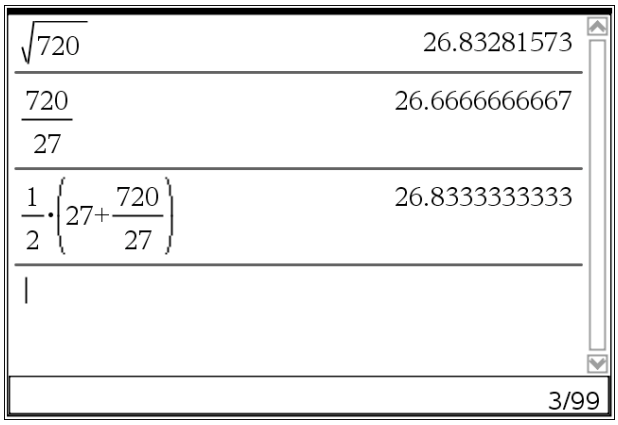
On peut dire que u_{k+1} est connu avec une erreur inférieure à 10^{-2p}.

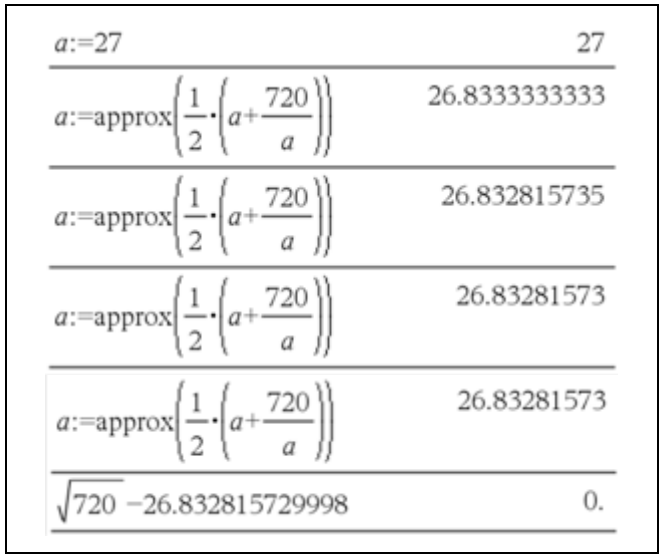
Stage algorithmique 1 TI-Nspire	Méthode de Héron d'Alexandrie

Voici les procédures que l'on peut employer sur TI-Nspire pour parvenir aux mêmes résultats.

Attention ! Les valeurs approchées, si on n'utilise pas la fonction **approx(** sont obtenues avec ctrl enter.

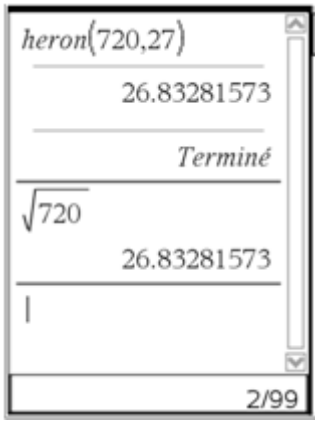
Pour obtenir des calculs successifs, on peut définir la variable *a* à partir de la valeur donnée précédemment à la variable *a*. Des appuis successifs sur enter donnent alors les résultats ci-dessous (à droite).





Dans les deux programmes **heron( )** et **heron2( )**, les nombres ***a*** (dont on cherche la racine) et ***b*** (valeur initiale) sont définis comme des arguments[i] du programme. Ainsi, **heron(**720,27**)** applique le programme pour *a* = 720 et *b* = 27 (voir les résultats obtenus avec le programme **heron**, ci-dessous).

```
Define heron(a,b)=
Prgm
Local c
c:=1/2·(b−a/b)
While |c−b|>10^-12
b:=c
c:=1/2·(b+a/b)
EndWhile
Disp c
EndPrgm
```




```
Define heron2(a)=
Prgm
Local b,c
b:=a
c:=1/2·(b+a/b)
While |b−c|>10^-12
b:=c
c:=1/2·(b+a/b)
EndWhile
Disp approx(c)
EndPrgm
```

Remarque : *Pour obtenir la valeur absolue de c – b, taper* abs(c – b).

[i] Ainsi, des instructions comme Input ou Prompt ne sont pas nécessaires dans ce type de programme.

Stage algorithmique 1 TI graphiques (82, 83, 84)	Treize

Le problème : on se propose de simuler, sur une calculatrice, le jeu suivant :
On lance un dé jusqu'à ce que le total des points obtenus lors des différents lancers soit supérieur ou égal à 13.
Combien faut-il de lancers de dé ?
Évidemment, il faut au moins trois lancers et au plus treize.

1. Description du programme

On utilise :

- Trois variables : une nommée S pour le total des points, initialisée à 0, une autre nommée N pour le nombre de lancers, initialisée également à 0 et une nommée A pour stocker provisoirement le résultat de chaque lancer ;
- Une structure répétitive : placer dans A un nombre aléatoire compris entre 1 et 6, afficher A, ajouter A à S, augmenter N de 1, recommencer tant que S est inférieur strictement à 13 ;
- Un affichage final.

2. Le programme

Algorithme	Programme : DE13

Mettre 0 dans S
Mettre 0 dans N
Tant que S est strictement inférieur à 13
  Mettre dans A un nombre aléatoire entre 1 et 6
  Afficher A
  Ajouter A à S
  Augmenter N de 1
Fin du tant que
Afficher N
	`Ø sto S`
	`Ø sto N`
	`While S<13`
	`entAléat(1,6) sto A`
	`Disp A`
	`S + A sto S`
	`N + 1 sto N`
	`End`
	`Disp "N: ",N`

3. Prolongement

Écrire un programme qui donne les résultats de 100 répétitions du programme précédent DE13.

Pour cela, commencer par supprimer les lignes d'affichage du programme DE13 pour en accélérer l'exécution ou le réécrire sans les deux lignes contenant l'instruction DISP.

Algorithme	Programme : DE100

Préparer une liste L1 de 13 éléments
Mettre 0 dans chaque élément de L1
Mettre 0 dans I
Tant que I est strictement inférieur à 100
  Effectuer le programme DE13 (sans affichage)
  Ajouter 1 au $N^{ième}$ élément de L1
  Augmenter I de 1
Fin du tant que
Afficher L1
	`13 sto dim(L1)`
	`Remplir (Ø,L1)`
	`Ø sto I`
	`While I < 1ØØ`
	`Prgm` `DE13`
	`L1(N) + 1 sto L1(N)`
	`I + 1 sto I`
	`End`
	`L1`

Remarques :

• Écrire **L_1** au lieu de **Disp L_1** suffit pour obtenir l'affichage en fin de programme.
• La liste **L_1** peut être facilement lue dans l'éditeur statistique par **List** 1:Edit.
• Une boucle **For (I, 1, 100) … End** remplace avantageusement **While … End** puisqu'alors il n'est pas nécessaire d'initialiser ni d'incrémenter la variable I.

4. Version sans sous-programme utilisant l'instruction For

Algorithme	Programme : DE100B

Préparer une liste L1 de 13 éléments
Mettre 0 dans chaque élément de L1
Pour I de 1 à 100
  Mettre 0 dans S
  Mettre 0 dans N
  Tant que S est strictement inférieur à 13
  Mettre dans A un nombre aléatoire entre 1 et 6
  Ajouter A à S
  Augmenter N de 1
  Fin du tant que
  Ajouter 1 au $N^{ième}$ élément de L1
Fin du For
Afficher L1
	13 sto dim (L_1)
	Remplir (Ø, L_1)
	For(I, 1, 100)
	Ø sto S
	Ø sto N
	While S<13
	entAléat(1,6) sto A
	S + A sto S
	N + 1 sto N
	End
	L_1(N) + 1 sto L_1(N)
	End
	L_1

Remarque :

On obtient directement l'analyse statistique de l'échantillon en modifiant la fin du programme de la manière suivante :
Éditer le programme, modifier la dernière ligne qui contient L_1 en : **Pause** L_1
Puis ajouter les instructions
suite(J, J, 1, 13) sto L_2
Stats 1-Var L_2, L_1
Après affichage de la liste L_1, on obtiendra les paramètres statistiques en appuyant sur ***Entrer/enter***.