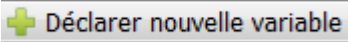
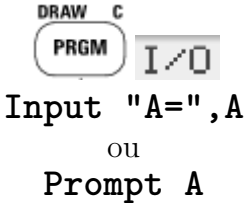
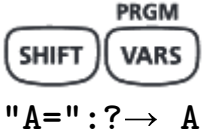
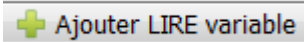
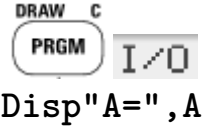
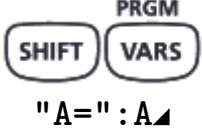
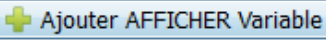
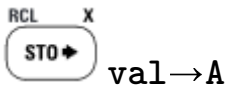

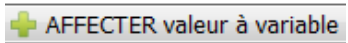

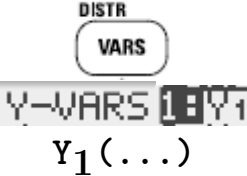


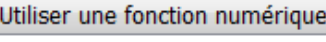
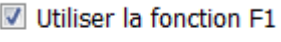



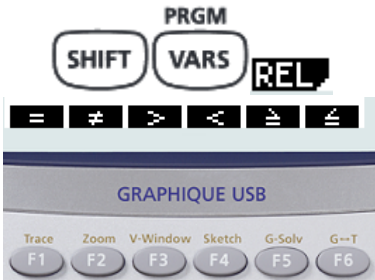

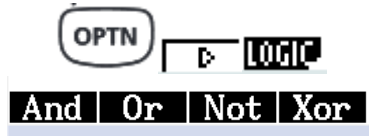


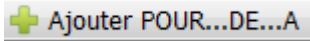

# **ALGORITHMIQUE.**



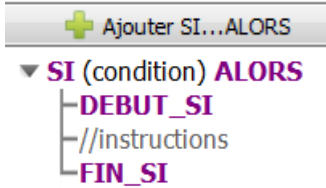


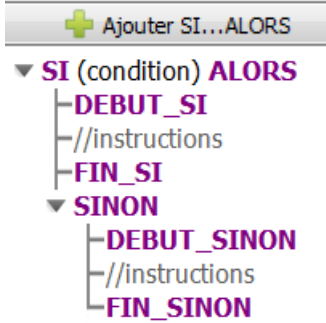


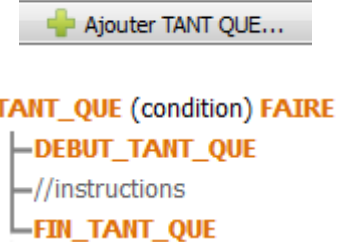
**Avril 2014**

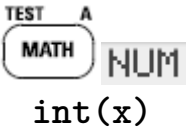
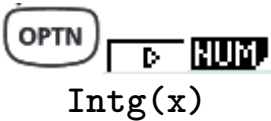
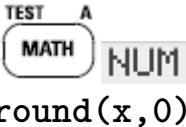
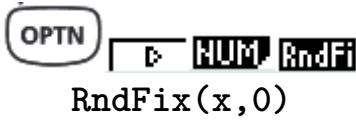
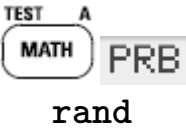

## TABLE DES MATIERES












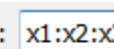







<b>Principales commandes pour programmer dans différents langages</b> <i>Tableau de comparaison pour les calculatrices TI, Casio, les logiciels Algobox et Xcas.</i>	<b>Page 3</b>
<b>Exemple de progression pour aborder l'algorithmique en seconde.</b> <b>Algorithmes au programme.</b> <i>Apprentissage progressif des instructions.</i>	<b>Page 8</b>
<b>Le jeu du « c'est plus, c'est moins ».</b> <i>Utilisation d'une boucle avec arrêt conditionnel et instruction conditionnelle. Calculatrices Casio et TI. A partir de la seconde.</i>	<b>Page 21</b>
<b>Longueur d'une courbe.</b> <i>Utilisation d'une boucle. A partir de la seconde.</i>	<b>Page 22</b>
<b>Tracer une courbe point par point.</b> <i>Boucles et fonctions. Comparaison des 2 types de boucles. A partir de la seconde.</i>	<b>Page 24</b>
<b>Méthode pour trouver les solutions de <math>f(x)=0</math>.</b> <i>Une alternative à la dichotomie. Utilisation de boucles, et d'instructions conditionnelles. A partir de la seconde.</i>	<b>Page 25</b>
<b>Boucles et boucles imbriquées</b> <i>Utilisation d'Algobox. A partir de la seconde</i>	<b>Page 26</b>
<b>Equation du second degré.</b> <i>Utilisation d'une instruction conditionnelle : sur calculatrice Casio et TI. A partir de la 1ère.</i>	<b>Page 27</b>
<b>Le jeu de « Pile-Face ».</b> <i>Utilisation d'une boucle avec arrêt conditionnel, et instruction conditionnelle. Utilisation de la fonction random. Logiciel Algobox. A partir de la 1ère.</i>	<b>Page 28</b>
<b>Un exemple de marche aléatoire.</b> <i>Utilisation de boucles, instructions conditionnelles, et de la fonction random. A partir de la 1èreS.</i>	<b>Page 30</b>
<b>Déplacement d'un robot sur un quadrillage.</b> <i>Utilisation de boucles, boucles imbriquées, d'instructions conditionnelles et de la fonction random. A partir de la 1èreS.</i>	<b>Page 33</b>
<b>Les records dans une suite de nombres.</b> <i>Boucles imbriquées, instructions conditionnelles. Logiciel Algobox, calculatrices TI. A partir de la 1èreS.</i>	<b>Page 37</b>
<b>Extraits d'exercices posés au baccalauréat</b>	<b>Page 41</b>
<b>La courbe du « Dragon », une fractale obtenue par pliages successifs.</b> <i>Boucles et instructions conditionnelles. Logiciel Algobox. Activité post-bac</i>	<b>Page 43</b>

Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Déclarer une variable $A$	Inutile	Inutile		<code>local A ;</code>
Saisir $A$	 ou Prompt $A$	 "A=" : ? → $A$		<code>saisir("Entrer A",A);</code> ou <code>saisir(A);</code> ou si on a une fonction : <code>nom_programme(A):={</code> <code>instruction(s); }::</code>
Afficher $A$	 Disp "A=", $A$	 "A=" : A $\blacktriangleleft$		<code>afficher("A vaut :",A);</code> ou <code>afficher(A);</code> ou si on a une fonction : <code>nom_programme(paramètres):={</code> <code>instruction(s);</code> <code>retourne A; }::</code>
Affecter à $A$ la valeur $val$	 $val \rightarrow A$	 $val \rightarrow A$		<code>A:=val;</code>
Utiliser une fonction externe dans un programme	Saisir la fonction dans l'éditeur graphique  puis la rappeler dans un programme :  $Y_1(\dots)$	Saisir la fonction dans le menu  ou  puis la rappeler dans un programme : $Y_1(\dots)$	cliquer sur l'onglet :  Saisir la fonction :  $F_1(x) = \text{pow}(x,3) - x - 1$ puis la rappeler dans un programme : $F_1(\dots)$	Définir la fonction (3 méthodes) : $f(x) := x^3 - x - 1$ $f := x \rightarrow x^3 - x - 1$ $f := \text{unapply}(x^3 - x - 1, x)$ On peut aussi utiliser une fonction comme variable d'un programme : <code>nom_programme() := {</code> <code>local f, ...;</code> <code>saisir(f); ... }::</code> Dans ce cas il faudra saisir dans l'invite : $x \rightarrow \dots$

Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Opérateurs de test et de logique				
Opérateurs de tests $=, \neq, >, <, \geq, \leq$			<ul style="list-style-type: none"> <li>• "<math>x = 2</math>" s'écrit <code>x==2</code></li> <li>• "<math>x \neq 2</math>" s'écrit <code>x!=2</code></li> <li>• "<math>x &lt; 2</math>" s'écrit <code>x&lt;2</code></li> <li>• "<math>x &gt; 2</math>" s'écrit <code>x&gt;2</code></li> <li>• "<math>x \leq 2</math>" s'écrit <code>x&lt;=2</code></li> <li>• "<math>x \geq 2</math>" s'écrit <code>x&gt;=2</code></li> </ul>	<ul style="list-style-type: none"> <li>• "<math>x = 2</math>" s'écrit <code>x==2</code></li> <li>• "<math>x \neq 2</math>" s'écrit <code>x!=2</code></li> <li>• "<math>x &lt; 2</math>" s'écrit <code>x&lt;2</code></li> <li>• "<math>x &gt; 2</math>" s'écrit <code>x&gt;2</code></li> <li>• "<math>x \leq 2</math>" s'écrit <code>x&lt;=2</code></li> <li>• "<math>x \geq 2</math>" s'écrit <code>x&gt;=2</code></li> </ul>
Opérateurs logiques et, ou, ou exclusif, non			<ul style="list-style-type: none"> <li>• le "et" s'écrit <code>ET</code></li> <li>• le "ou" s'écrit <code>OU</code></li> </ul>	<ul style="list-style-type: none"> <li>• le "et" s'écrit <code>et</code></li> <li>• le "ou" s'écrit <code>ou</code></li> <li>• le "ou exclusif" s'écrit <code>xor</code></li> <li>• le non s'écrit <code>non</code></li> </ul>
Boucle Pour ...de ...jusque ...faire ...Fpour				
Pour $I$ de 1 jusque $N$ faire <i>instructions</i> Fpour	 <b>For(I,1,N)</b> <i>instructions</i> <b>End</b>	 <b>For 1→I To N</b> <i>instructions</i> <b>Next</b>	Il faudra déclarer auparavant la variable <b>I</b>  <b>POUR I ALLANT_DE 1 A N</b> <b>DEBUT_POUR</b> //instructions <b>FIN_POUR</b>	<b>pour j de 1jusque N faire</b> <i>instructions</i> ; <b>fpour</b> ;  Ne pas utiliser la variable <b>i</b> comme compteur car c'est une lettre prédéfinie qui désigne le <b>i</b> des complexes.

Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Instruction conditionnelle <b>Si...alors...[Sinon]...Fsi</b>				
Si <i>conditions</i> alors <i>instructions</i> Fsi	 <b>If</b> <i>conditions</i> <b>Then</b> <i>instructions</i> <b>End</b>	 <b>If</b> <i>conditions</i> <b>Then</b> <i>instructions</i> <b>IfEnd</b>		<b>si</b> <i>conditions</i> <b>alors</b> <i>instructions</i> ; <b>fsi</b> ;
Si <i>conditions</i> alors <i>instructions</i> Sinon <i>instructions</i> Fsi	 <b>If</b> <i>conditions</i> <b>Then</b> <i>instructions</i> <b>Else</b> <i>instructions</i> <b>End</b>	 <b>If</b> <i>conditions</i> <b>Then</b> <i>instructions</i> <b>Else</b> <i>instructions</i> <b>IfEnd</b>		<b>si</b> <i>conditions</i> <b>alors</b> <i>instructions</i> ; <b>sinon</b> <i>instructions</i> ; <b>fsi</b> ;
Boucle avec arrêt conditionnel <b>Tantque ...faire ...Ftantque</b>				
Tant que <i>conditions</i> faire <i>instructions</i> Ftantque	 <b>While</b> <i>condition</i> <i>instructions</i> <b>End</b>	 <b>While</b> <i>condition</i> <i>instructions</i> <b>WhileEnd</b>		<b>tantque</b> <i>condition</i> <b>faire</b> <i>instructions</i> ; <b>ftantque</b> ;

Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Fonctions mathématiques				
Racine carrée $\sqrt{x}$	$\sqrt{x}$	$\sqrt{x}$	<code>sqrt(x)</code>	<code>sqrt(x)</code>
Puissance $x^n$	$x^n$	$x^n$	<code>pow(x,n)</code>	$x^n$
Partie entière de $x$	 int(x)	 Intg(x)	<code>floor(x)</code>	<code>floor(x)</code>
Arrondi à l'unité de $x$	 round(x,0)	 RndFix(x,0)	<code>round(x)</code>	<code>round(x)</code>
Reste de la division euclidienne de $A$ par $B$	<code>A-B*int(A/B)</code>	MOD(A,B) (certaines calculatrices) <code>A-B*Intg(A÷B)</code>	<code>A%B</code>	<code>irem(A,B)</code>
Logarithme népérien de $x$ : $\ln(x)$	<code>ln(x)</code>	<code>ln(x)</code>	<code>log(x)</code>	<code>ln(x)</code>
Exponentielle de $e^x$	<code>e^x</code>	<code>e^x</code>	<code>exp(x)</code>	<code>exp(x)</code>
Nombre réel pseudo-aléatoire dans $[0; 1[$	 rand	 Rand#	<code>random()</code>	<code>rand(0,1)</code>
Entier aléatoire dans $[a; b]$ , avec $a$ et $b$ deux entiers donnés	avec la partie entière : <code>a+int((b-a+1)*rand)</code>	avec la partie entière : <code>a+Intg((b-a+1)*Rand#)</code>	ALGOBOX_ALEA_ENT(a,b) ou <code>a+floor((b-a+1)*random())</code>	<code>a+rand(b-a+1)</code>

Langage algorithmique	Langages de programmation			
	Sur TI	Sur Casio	Logiciel Algobox	Logiciel Xcas
Listes				
Créer et remplir une liste	<p>Les listes <math>L_1, L_2, \dots, L_2</math> existent déjà dans le mode</p> <p>STAT :  </p> <p>On peut donc les remplir directement avec ce menu.</p> <p>Cela peut se faire aussi dans le menu courant avec la commande :</p> <p><math>\{x_1, \dots, x_n\} \rightarrow L_1</math></p> <p>On peut l'afficher dans le menu courant en tapant :</p> <p>  </p>	<p>Les listes <b>List 1</b>, <b>List 2</b>, ..., <b>List 26</b> existent déjà dans le menu <b>STAT</b> :</p> <p></p> <p>On peut donc les remplir directement avec ce menu.</p> <p>Cela peut se faire aussi dans le menu courant avec la commande :</p> <p><math>\{x_1, \dots, x_n\} \rightarrow \text{List 1}</math></p> <p>On peut l'afficher dans le menu courant en tapant :</p> <p> </p>	<p> <b>Déclarer nouvelle variable</b></p> <p>puis préciser le type <b>Liste</b>.</p> <p>Pour la remplir :</p> <p> <b>AFFECTER valeur à variable</b></p> <p>puis</p> <p> prend la valeur : </p> <p>le de la liste : </p> <p>en mettant <b>1</b> au rang de la liste et en séparant chaque valeur par :</p> <p>Pour afficher le contenu d'une liste, on utilise une boucle.</p>	<p>Pour créer une liste <math>L := [x_1, \dots, x_n]</math></p> <p>Pour afficher le contenu d'une liste :</p> <p><b>retourne L</b></p>
Élément de rang $k$ d'une liste	<p>Le premier rang d'une liste <math>L_1</math> est 1 et le dernier rang est <b>Dim(<math>L_1</math>)</b>.</p> <p><math>L_1(k)</math> est le terme de rang <math>k</math> de la liste 1.</p>	<p>Le premier rang d'une liste <b>List 1</b> est 1 et le dernier rang est <b>Dim List 1</b>.</p> <p><b>List 1[k]</b> est le terme de rang <math>k</math> de la liste 1.</p>	<p><math>L[1]</math> est le premier terme de la liste <b>L</b> (on peut débiter à 0 : <math>L[0]</math>).</p> <p><math>L[k]</math> est le terme de rang <math>k</math> de la liste <b>L</b>.</p> <p>La longueur d'une liste commençant à 1 est donnée par <b>L.length-1</b></p>	<p> <math>L[0]</math> ou <math>L(1)</math> désignent le premier terme de la liste <b>L</b>.</p> <p><math>L[k]</math> est le terme de rang <math>k</math> de la liste <b>L</b> donc le <math>(k + 1)</math>-ème terme de cette liste</p> <p>La longueur d'une liste est donnée par <b>dim(L)</b></p>
Remplir une liste avec $p$ entiers aléatoires pris dans $[a; b]$ , avec $a$ et $b$ deux entiers donnés	<p>Avec la commande <b>seq</b></p> <p>  </p> <p><b>seq(a+int((b-a+1)*rand),K,1,p,1) → L<sub>1</sub></b></p>	<p>Avec la commande <b>Seq</b></p> <p> </p> <p><b>seq(a+Intg((b-a+1)*Rand#),K,1,p,1) → List 1</b></p>	<p>Il faut créer une boucle pour remplir la liste terme après terme :</p> <p><b>POUR k ALLANT_DE 1 A p</b></p> <p>  <b>DEBUT_POUR</b></p> <p>    <b>L[k] PREND_LA_VALEUR ALGOBOX_ALEA_ENT(a,b)</b></p> <p>  <b>FIN_POUR</b></p>	<p><b>L :=</b></p> <p><b>[(a+rand(b-a+1))\$(k=1..p)]</b></p>

## EXEMPLE DE PROGRESSION POUR ABORDER L'ALGORITHMIQUE EN SECONDE.

### Introduction :

Le but des séances présentées est de familiariser les élèves à la lecture d'algorithmes simples, à leur création en langage naturel, puis à les réaliser soit avec le logiciel ALGOBOX soit en les programmant sur leur calculatrice.

Les notions du programme ont été abordées en trois temps durant les séances de module à 18 élèves. Chaque partie débute par la découverte des notions, puis quelques définitions, suivis des syntaxes : algorithme papier – logiciel ALGOBOX – calculatrice TI – calculatrice CASIO. Enfin des applications sont proposées pour mettre en pratique ces notions.

Toutes ces activités ont été menées lors des deux premiers trimestres. Au dernier trimestre, les élèves par petits groupes ont du traiter l'un des devoirs maison proposés.

### ALGORITHMIQUE (1ere partie)

#### Les instructions d'entrée-sortie, l'affectation.

Voici un programme de calcul :

- \*choisir un nombre
- \*le multiplier par 5
- \*ajouter 3 au produit obtenu
- \*Multiplier le nombre obtenu par celui choisi au départ
- \*Ecrire le résultat

On appelle  $x$  le nombre choisi au départ. Appliquer ce programme pour  $x = 5$ , puis  $x = 26$  et  $x = 100$

#### Définition d'un algorithme

Un algorithme est une succession d'instructions à enchaîner **dans un ordre bien précis**, permettant de **résoudre un problème de façon systématique**. Il est écrit dans un langage compréhensible par tous.

Voici l'algorithme qui correspond au programme de calcul.

Variables :  $x$ ,  $a$  : réels

Début :

Saisir  $x$   
 $a$  reçoit .....  
afficher  $a$

Fin

Le compléter.

#### Affectation.

Il s'agit d'attribuer une valeur à une variable, valeur qui peut être de plusieurs types : numérique (entier ou réel), alphanumérique (texte), booléen (vrai ou faux).

Syntaxe :

**\*  $a$  prend la valeur 2** ; on affecte la valeur 2 à la variable  $a$  ou  $a$  reçoit la valeur 2

**\* .....** ; on affecte à la variable  $b$  le contenu de la variable  $a$  auquel on ajoute 3, c'est-à-dire .....



### Entrée d'une valeur.

Au moment de l'exécution de l'algorithme, l'utilisateur affecte une valeur à une variable. Lors du fonctionnement de l'algorithme, celui-ci s'arrête à cette instruction et ne se poursuit que lorsque l'utilisateur a entré une valeur.

Syntaxe : « Saisir a » ou « lire a »

### Affichage d'une valeur.

Il s'agit d'afficher la valeur d'une variable.

Syntaxe : « afficher a »

### Syntaxe des instructions

Algorithme papier	albox	Calculatrice TI	Calculatrice Casio
A prend la valeur 2	A prend la valeur 2	$2 \longrightarrow A$	$2 \longrightarrow A$
Saisir A	Lire A	Prompt A ou Input « A= »,A	? $\longrightarrow$ A
Afficher A	Afficher A	Disp A	A $\Delta$

- Ecrire un algorithme papier, puis avec Albox et enfin avec votre calculatrice permettant le calcul des coordonnées du milieu du segment [AB] connaissant les coordonnées des points A et B

Algorithme papier	albox	Calculatrice TI	Calculatrice Casio
<p>Variables : <math>x_A, y_A, x_B, y_B, x_I, y_I</math> : réels</p> <p>Début :</p> <p>Saisir <math>x_A, y_A, x_B, y_B</math></p> <p><math>x_I</math> prend la valeur <math>\frac{x_A + x_B}{2}</math></p> <p><math>y_I</math> prend la valeur <math>\frac{y_A + y_B}{2}</math></p> <p>Afficher « les coordonnées sont »</p> <p>Afficher <math>x_I, y_I</math></p> <p>Fin</p>	<p><b>VARIABLES</b></p> <ul style="list-style-type: none"><li>xA EST_DU_TYPE NOMBRE</li><li>xB EST_DU_TYPE NOMBRE</li><li>yA EST_DU_TYPE NOMBRE</li><li>yB EST_DU_TYPE NOMBRE</li><li>x EST_DU_TYPE NOMBRE</li><li>y EST_DU_TYPE NOMBRE</li></ul> <p><b>DEBUT_ALGORITHME</b></p> <ul style="list-style-type: none"><li>LIRE xA</li><li>LIRE yA</li><li>LIRE xB</li><li>LIRE yB</li><li>x PREND_LA_VALEUR <math>(x_A + x_B)/2</math></li><li>y PREND_LA_VALEUR <math>(y_A + y_B)/2</math></li><li>AFFICHER "les coordonnées du milieu sont "</li><li>AFFICHER x</li><li>AFFICHER y</li></ul> <p><b>FIN_ALGORITHME</b></p>	<p>Input « XA= »,X</p> <p>Input « YA= »,Y</p> <p>Input « XB= »,Z</p> <p>Input « YB= »,T</p> <p><math>(Z+X)/2 \longrightarrow C</math></p> <p><math>(Y+T)/2 \longrightarrow D</math></p> <p>Disp"XI=",C</p> <p>Disp"YI=",D</p>	<p>« XA » ? <math>\longrightarrow</math> X</p> <p>« YA » ? <math>\longrightarrow</math> Y</p> <p>« XB » ? <math>\longrightarrow</math> Z</p> <p>« YB » ? <math>\longrightarrow</math> T</p> <p><math>(Z+X)/2 \longrightarrow C</math></p> <p><math>(Y+T)/2 \longrightarrow D</math></p> <p>« XI= »</p> <p>C<math>\Delta</math></p> <p>« YI= »</p> <p>D<math>\Delta</math></p>

- Ecrire un algorithme papier, puis avec Albox et enfin avec votre calculatrice permettant le calcul de la longueur AB connaissant les coordonnées des points A et B

## ALGORITHMIQUE (2ème partie) : La structure alternative ou test

### Découverte : (d'après le livre Transmath de 2de)

Dans un repère orthonormé (O ; I, J), on considère les points A, B et C de coordonnées respectives  $(x_A, y_A)$ ,  $(x_B, y_B)$ ,  $(x_C, y_C)$ .

1. Exprimer  $CB^2$  et  $AC^2$  en fonction des coordonnées de A, B et C.
2. Justifier «  $CB^2 = AC^2$  » implique « ABC est un triangle isocèle en C »
3. Compléter l'algorithme suivant :

Variables :  $x_A, y_A, x_B, y_B, x_C, y_C, S, H$  : réels

Début

Saisir  $x_A, y_A, x_B, y_B, x_C, y_C$

S reçoit  $(\dots)^2 + (\dots)^2$

H reçoit  $(\dots)^2 + (\dots)^2$

Si ..... Alors

Afficher « ABC est un triangle isocèle en C »

Sinon

Afficher « ABC n'est pas un triangle isocèle en C »

Fin SI

Fin

### Définition

Une condition est un énoncé qui peut être vrai ou faux. Par exemple  $a = b$  ou  $n$  est pair

Dans un programme, selon qu'une condition est vraie ou fausse, on peut effectuer un traitement ou un autre : on parle de traitements conditionnels.

On traduit la structure alternative par les instructions suivantes :

Syntaxe :

```
Si      condition      alors
                        Traitement 1
                        Sinon
                        Traitement 2
```

FinSi

On peut également imaginer des tests imbriqués

### Syntaxe des instructions

Algorithme papier	Algobox	Calculatrice TI	Calculatrice Casio
Si A=2 alors ..... Sinon ..... FinSi	SI (A==2) ALORS DEBUT SI ..... FIN_SI SINON DEBUT_SINON ..... FIN_SINON	If A=2 Then ..... Else ..... End	If A=2 Then ..... Else ..... If End

Application:

1. Tracer une droite graduée et y placer les nombres 3; (-2);  $\frac{2}{3}$
2. Indiquer la distance à 0 de chacun de ces nombres.
3. Cette distance se nomme la valeur absolue du nombre. Compléter la définition :  
  
Si  $x$  est positif, la valeur absolue de  $x$  est .....  
  
Si  $x$  est négatif, la valeur absolue de  $x$  est .....
4. Ecrire un algorithme papier, puis avec Algobox et enfin avec votre calculatrice permettant d'afficher la valeur absolue d'un nombre donné.

Algorithme papier	algobox	Calculatrice TI	Calculatrice Casio
Variable : X, A : réels Début : Saisir X Si $X \geq 0$ alors A prend la valeur X Sinon A prend la valeur (-X) FinSI Afficher « la valeur absolue est » Afficher A Fin	<pre> <b>VARIABLES</b> ├── X EST_DU_TYPE NOMBRE ├── A EST_DU_TYPE NOMBRE └── <b>DEBUT_ALGORITHME</b>     ├── LIRE X     ├── SI (X &gt;= 0) ALORS     │   ├── <b>DEBUT_SI</b>     │   ├── A PREND_LA_VALEUR X     │   └── <b>FIN_SI</b>     └── SINON         ├── <b>DEBUT_SINON</b>         ├── A PREND_LA_VALEUR -X         └── <b>FIN_SINON</b>     AFFICHER "La valeur absolue est "     AFFICHER A   <b>FIN_ALGORITHME</b>           </pre>	Prompt X If $X \geq 0$ Then $X \rightarrow A$ Else $-X \rightarrow A$ End Disp "LA VALEUR ABSOLUE EST",A	? $\rightarrow$ X If $X \geq 0$ Then $X \rightarrow A$ Else $-X \rightarrow A$ IfEnd "LA VALEUR ABSOLUE EST" A $\Delta$

## ALGORITHMIQUE (3ème partie) : Les structures itératives ou boucles

### Découverte :

Partie 1 : d'après le livre Math'x de 2de

Voici un algorithme :

Variables : N, I, S : réels

Début

Afficher « saisir un entier N : »

Saisir N

Pour I de 0 à 12

S reçoit  $N \times I$

Afficher N « × » I « = » S

Fin Pour

Fin

1. Tester cet algorithme pour  $N=5$
2. Quel est le but de cet algorithme ?
3. A quoi sert le « Pour I de 0 à 12 »

Partie 2 : d'après le livre Transmath de 2de

Un jeu de 100 cubes. Un enfant construit des « pyramides » avec ses cubes.

Etape 1 : 1 cube

Etape 2 : 3 cubes empilés (2 cubes à la base et un en 2<sup>ème</sup> ligne)

Etape 3 : 6 cubes empilés (3 cubes à la base, 2 en 2<sup>ème</sup> ligne et un en 3<sup>ème</sup> ligne)

Jusqu'à quelle étape peut-on aller et combien de cubes resteront-ils alors ?

1. Déterminer le nombre total de cubes empilés à l'étape 4, puis à l'étape 5.
2. Combien de cubes sont nécessaires pour passer de l'étape 3 à l'étape 4, puis de l'étape 4 à l'étape 5, puis de l'étape (n-1) à l'étape n avec  $n \geq 2$  ?
3. Compléter l'algorithme suivant dont le but est d'afficher le nombre N d'étapes que l'on peut réaliser avec 100 cubes et le nombre R de cubes restants.

Variables : N, S, R : réels

Début :

N reçoit 0

S reçoit 0

R reçoit 100

Tant que  $R \geq N+1$

N reçoit ....

S reçoit  $S + N$

R reçoit

Fin Tant que

Afficher .....

Afficher .....

Fin

4. Compléter le tableau d'avancement suivant jusqu'à  $N=4$

N	S	R	Variables
0	0	100	Initialisation
1			Déroulement de la boucle

5. Que représente S ?

### Partie 3 :

1. Dans ces deux activités, on a utilisé des boucles. Comment les définir ?
2. Quelle est la différence entre ces deux boucles ?

#### **Définitions**

Une boucle permet de répéter plusieurs fois de suite un même traitement.

- Lorsque le nombre de répétitions (ou itérations) noté « n » est connu à l'avance, on utilise un compteur initialisé à 1 et qui s'incrémente automatiquement de 1 à chaque itération jusqu'à n. On parle de boucle itérative

#### Syntaxe :

Pour I de 1 à n  
    Traitement  
Fin Pour

- Lorsque le nombre de répétitions (ou itérations) noté « n » n'est pas connu à l'avance, il peut dépendre d'une condition, le traitement est répété tant que la condition est vraie. Lorsqu'elle est fausse, on sort de la boucle. On parle de boucle conditionnelle.

#### Syntaxe :

Tant que condition  
    Traitement  
Fin Tant que

#### **Syntaxe des instructions**

Algorithme papier	Algobox	Calculatrice TI	Calculatrice Casio
Pour I de 1 à N ..... FinPour	POUR I ALLANT DE de 1 à N DEBUT_POUR ..... FIN_POUR	For(I,1,N) ..... End	For 1→I to N ..... Next
Tant que X<2 ..... FinTant que	TANT_QUE X<2 FAIRE DEBUT_TANT_QUE ..... FIN_TANT_QUE	While X<2 ..... End	While X<2 ..... WhileEnd

#### **Applications :**

1. (d'après le livre Transmath de 2de)

On souhaite écrire un programme sous Algobox permettant d'afficher la courbe représentant la fonction

$$f(x) = \frac{x^3}{2} - x + 1 \text{ sur l'intervalle } I = [-2 ; 2]. \text{ Pour cela répondre aux questions suivantes :}$$

- A. On partage I en 100 intervalles de même amplitude. Quelle est la distance entre les abscisses de deux points consécutifs ?
- B. Quelles sont les coordonnées du premier point tracé ?
- C. Quelles sont les coordonnées du second point tracé ?
- D. Quelles sont les coordonnées du dixième point tracé ?

E. Quelles sont les coordonnées du dernier point tracé ?

F. Compléter l'algorithme suivant :

Variables :  $x, y, i$  : réels

Début :

Pour  $i$  allant de ..... à .....

$x$  reçoit  $-2 + \dots$

$y$  reçoit .....

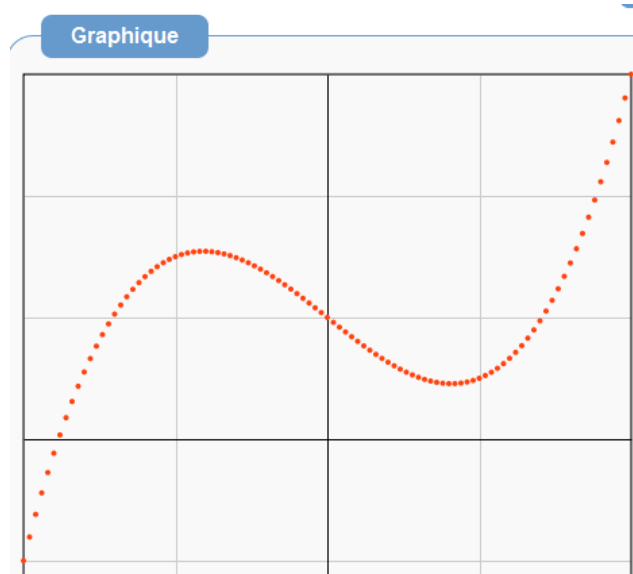
Placer le point de coordonnées  $(x; y)$

Fin Pour

Fin

G. Ecrire le programme sur Algobox

```
▼ VARIABLES
├── x EST_DU_TYPE NOMBRE
├── y EST_DU_TYPE NOMBRE
└── i EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
├── POUR i ALLANT_DE 0 A 100
│   ├── DEBUT_POUR
│   ├── x PREND_LA_VALEUR -2+i/25
│   ├── y PREND_LA_VALEUR F1(x)
│   ├── TRACER_POINT (x,y)
│   └── FIN_POUR
└── FIN_ALGORITHME
```



Remarque:

1. Il faut entrer la fonction  $f$  dans l'onglet « utiliser une fonction numérique »

Opérations standards   Utiliser une fonction numérique   Dessiner dans un repère

☒ Utiliser une fonction

Définir la fonction

$F1(x) = (x*x*x)/2 - x + 1$

2. Il faut définir le repère dans l'onglet « Dessiner dans un repère »

Opérations standards   Utiliser une fonction numérique   Dessiner dans un repère

☒ Utiliser un repère

Ajouter TRACER POINT   Ajouter TRACER SEGMENT

Définir le repère

Xmin :	-2	Xmax :	2	Graduations X :	1
Ymin :	-2	Ymax :	3	Graduations Y :	1

2. Pierre place 5000€ sur un compte épargne à 2% par an. Chaque année, les intérêts s'ajoutent au capital. Il compte aussi placer 200€ de plus par an. Il souhaite savoir au bout de combien d'années son épargne dépassera 10 000€, et combien il aura alors.

Ecrire un algorithme papier, puis avec Algorithbox, puis sur votre calculatrice pour répondre au problème.

Algorithme papier	algorithbox	Calculatrice TI	Calculatrice Casio
Variable : X, N : réels Début : X reçoit 5000 N reçoit 0 Tant que $X \leq 10000$ N reçoit N+1 X reçoit $X \cdot 1.02 + 200$ FinTant que Afficher « le nombre d'années est » Afficher N+1 Fin	<pre> <b>VARIABLES</b>   X EST_DU_TYPE NOMBRE   N EST_DU_TYPE NOMBRE <b>DEBUT_ALGORITHME</b>   X PREND_LA_VALEUR 5000   N PREND_LA_VALEUR 0   <b>TANT_QUE</b> (X &lt;= 10000) <b>FAIRE</b>     <b>DEBUT_TANT_QUE</b>       N PREND_LA_VALEUR N+1       X PREND_LA_VALEUR X*1.02+200     <b>FIN_TANT_QUE</b>     AFFICHER "le nombre d'année est "     N PREND_LA_VALEUR N+1     AFFICHER N   <b>FIN_ALGORITHME</b>           </pre>	<pre> 5000 → X 0 → N While X ≤ 10000   X*1.02+200 → X   N+1 → N End Disp "LE NOMBRE D'ANNEE EST ",N+1           </pre>	<pre> 5000 → X 0 → N While X ≤ 10000   X*1.02+200 → X   N+1 → N WhileEnd "LE NOMBRE D'ANNEE EST » N+1 ↵           </pre>

3. Calculs d'image par une fonction homographique. (d'après Transmath 2de)

On considère l'algorithme suivant :

Variable : A, B, Q : réels

Début :

Saisir A, B

Si  $B \neq 0$

Alors Q reçoit  $\frac{A}{B}$

Afficher Q

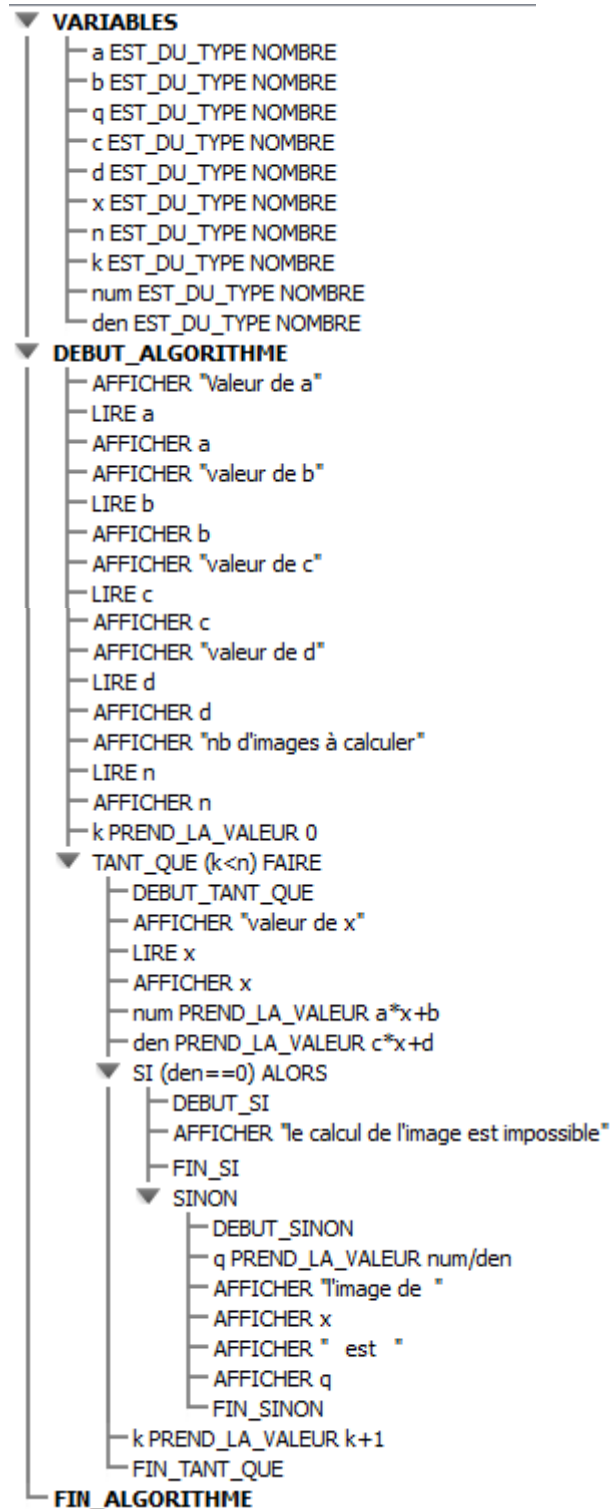
Sinon Afficher « calcul impossible »

Fin Si

Fin

- Tester cet algorithme pour  $a = 2$  et  $b = 4$ , puis  $a = 0$  et  $b = 3$ , enfin  $a = 10$  et  $b = 0$
- Quel est le but de cet algorithme ?
- Ecrire un algorithme papier permettant d'afficher l'image d'un nombre  $x$  par la fonction  $f$  définie par  $f(x) = \frac{3x+2}{x-5}$  si c'est possible et sinon d'afficher un message d'impossibilité
- Le taper sous algorithbox
- Modifier votre algorithme pour que ce dernier demande la valeur de  $x$ .
- Modifier votre algorithme pour éviter de relancer l'algorithme si on désire calculer l'image de plusieurs réels.
- Modifier votre algorithme pour qu'il fasse le travail demandé pour n'importe quelle fonction homographique.

Voici un exemple d'algorithme pour cette dernière question :





#### 4. Méthode de dichotomie :

On considère la fonction polynôme  $f(x) = x^2 - x - 4$ , et on cherche une valeur approchée de la solution  $\alpha$  de l'équation  $f(x) = 0$  sur  $[1 ; 3]$ .

- Calculer les images de 1, 2 puis de 3 par  $f$ . En déduire un encadrement de  $\alpha$
- Calculer l'image de 2.5 par  $f$ . Que peut-on en déduire pour l'encadrement de  $\alpha$ .
- Calculer l'image de 2.75 par  $f$ . Que peut-on en déduire pour l'encadrement de  $\alpha$
- Voilà l'algorithme permettant de trouver un encadrement d'amplitude souhaitée de  $\alpha$

Variable : A, B : réels

N : entier

Début :

Saisir A, B, N

Tant que  $B - A > 10^{-N}$

Si  $f(A) \times f(\frac{A+B}{2}) > 0$

Alors A reçoit  $\frac{A+B}{2}$

Sinon B reçoit  $\frac{A+B}{2}$

Fin Si

FinTant que

Afficher A, B

Fin

\*Pourquoi teste-t-on  $f(A) \times f(\frac{A+B}{2}) > 0$

\*Quelle est l'utilité de Tant que  $B - A > 10^{-N}$

\*Effectuer cet algorithme à la main en complétant le tableau suivant et en prenant  $N=1$ :

Valeur de A	Valeur de B	Valeur de B-A	Valeur de $\frac{A+B}{2}$	Signe de $f(A) \times f(\frac{A+B}{2})$	Nouvelle valeur de A	Nouvelle valeur de B
1	3					

5. Application : Ecrire un algorithme papier, puis avec Algobox permettant d'encadrer à  $10^{-3}$  près  $\sqrt{150}$

Algorithme papier	algobox
<p>Variable : A, B, M, N : réels</p> <p>Début :</p> <p>N reçoit 1</p> <p>Tant que <math>N^2 &lt; 150</math></p> <p>N reçoit <math>N+1</math></p> <p>Fin Tant que</p> <p>A reçoit <math>N-1</math></p> <p>B reçoit N</p> <p>Tant que <math>B-A &gt; 10^{-3}</math></p> <p>M reçoit <math>\frac{A+B}{2}</math></p> <p>Si <math>M^2 &lt; 150</math></p> <p>Alors A reçoit M</p> <p>Sinon B reçoit M</p> <p>Fin Si</p> <p>FinTant que</p> <p>Afficher « La racine de 150 est comprise entre »</p> <p>Afficher A</p> <p>Afficher « et »</p> <p>Afficher B</p> <p>Fin</p>	<pre> VARIABLES   a EST_DU_TYPE NOMBRE   b EST_DU_TYPE NOMBRE   m EST_DU_TYPE NOMBRE   n EST_DU_TYPE NOMBRE  DEBUT_ALGORITHME   n PREND_LA_VALEUR 1   TANT_QUE (pow(n,2) &lt; 150) FAIRE     DEBUT_TANT_QUE       n PREND_LA_VALEUR n+1     FIN_TANT_QUE     a PREND_LA_VALEUR n-1     b PREND_LA_VALEUR a+1     TANT_QUE (b-a &gt; 0.001) FAIRE       DEBUT_TANT_QUE         m PREND_LA_VALEUR (a+b)/2         SI (pow(m,2) &lt; 150) ALORS           DEBUT_SI             a PREND_LA_VALEUR m           FIN_SI         SINON           DEBUT_SINON             b PREND_LA_VALEUR m           FIN_SINON       FIN_TANT_QUE     AFFICHER "la racine demandée est comprise entre "     AFFICHER a     AFFICHER " et "     AFFICHER b   FIN_TANT_QUE FIN_ALGORITHME </pre>

Epilogue : Voici les différents devoirs à la maison proposés

### DEVOIR A LA MAISON N°A

Dans un repère (O,I, J) orthonormé, on considère les points A (4 ; 0), B(4 ;4) et C(0 ;4).

Un point M se déplace sur les côtés du carré en partant de O et en suivant le chemin

$O \rightarrow A \rightarrow B \rightarrow C \rightarrow O$

On note d la distance parcourue par le point M depuis le départ O.

- A quel intervalle appartient d ?
- Déterminer les coordonnées de M et la longueur OM dans les cas suivants en s'aidant d'une figure si nécessaire:
  - $d = 2$
  - $d = 4$
  - $d = 7$
  - $d = 9$
  - $d = 13$
- Déterminer les coordonnées de M et la longueur OM dans les cas suivants en s'aidant d'une figure si nécessaire:
  - $M \in [OA]$
  - $M \in [AB]$
  - $M \in [BC]$
  - $M \in [CO]$
- Créer un algorithme avec ALGOBOX permettant l'affichage des coordonnées de M et la longueur OM .

### **DEVOIR A LA MAISON N°B**

**Partie 1 :** Dans un repère (O ;I, J) orthonormé, on considère les points A (1 ; 2), B(7 ;0) et C(5 ;4).

1. Calculer les longueurs AB, AC et BC
2. Le triangle ABC est-il équilatéral ? Justifier.
3. Le triangle ABC est-il isocèle ? Justifier.
4. Le triangle ABC est-il rectangle ? Justifier.

**Partie 2 :** Créer un algorithme avec ALGOBOX qui permet de connaître la nature d'un triangle ABC connaissant les coordonnées des trois points A(xA,yA), B(xB ,yB), C(xC , yC).

### **DEVOIR A LA MAISON N°C**

**Partie 1 :** Dans un repère (O.I, J) orthonormé, on considère les points A(2 ; 3), B(6 ;4) et C(7 ;0) et D(3 ;-1).

1. Calculer les coordonnées des vecteurs  $\overrightarrow{AB}$  et  $\overrightarrow{DC}$
2. Que pouvez-vous en déduire pour le quadrilatère ABCD ?
3. Calculer les longueurs AB, AC et BC.
4. Le quadrilatère ABCD est-il un losange ? Justifier.
5. Le quadrilatère ABCD est-il un rectangle ? Justifier.
6. Le quadrilatère ABCD est-il un carré ? Justifier.

**Partie 2 :** Créer un algorithme avec ALGOBOX qui permet de connaître la nature d'un quadrilatère ABCD connaissant les coordonnées des quatre points A(xA,yA), B(xB ,yB), C(xC , yC), D(xD,yD).

### **DEVOIR A LA MAISON N°D**

**Partie 1 :** Dans un repère (O.I, J) orthonormé, on considère les points A (2 ; 3), B(6 ;4) et C(7 ;0) et D(3 ;-1).

1. Calculer les coordonnées des milieux de [AC] et [BD]
2. Que pouvez-vous en déduire pour le quadrilatère ABCD ?
3. Calculer les longueurs AB, AD et BC.
4. Le quadrilatère ABCD est-il un losange ? Justifier.
5. Le quadrilatère ABCD est-il un rectangle ? Justifier.
6. Le quadrilatère ABCD est-il un carré ? Justifier.

**Partie 2 :** Créer un algorithme avec ALGOBOX qui permet de connaître la nature d'un quadrilatère ABCD connaissant les coordonnées des quatre points A(xA,yA), B(xB ,yB), C(xC , yC), D(xD,yD).

### **DEVOIR A LA MAISON N°E**

**Partie 1 :** On considère le polynôme du second degré  $f(x) = x^2 - 4x - 6$

1. Que valent a, b et c
2. Calculer les coordonnées du sommet de la parabole.
3. Quelles sont les variations de f ? Justifier.

**Partie 2 :** Créer un algorithme avec ALGOBOX qui calcule et affiche les coordonnées du sommet de la parabole, ainsi que les variations de f connaissant les trois réels a, b et c.

### **DEVOIR A LA MAISON N°F**

**Partie 1 :** Dans un repère (O.I, J) orthonormé, on considère les points A (-2 ; 3.1), B(0 ;-1.7) et C(5 ;-5.2) .

1. Calculer les coordonnées des vecteurs  $\overrightarrow{AB}$  et  $\overrightarrow{AC}$
2. Ces deux vecteurs sont-ils colinéaires ?
3. Que peut-on en déduire pour les points A, B, C ? Justifier.
4. Même travail avec A (1.2 ; 2.4), B(-2 ;-4) et C(3 ;6) .

**Partie 2 :** Créer un algorithme avec ALGOBOX qui permet de savoir si les points A, B et C sont alignés, connaissant les coordonnées des trois points A(xA,yA), B(xB ,yB), C(xC , yC)

## **DEVOIR A LA MAISON N°G**

### **Partie 1 :**

Dans un repère (O.I, J) orthonormé, on considère une fonction  $f$  telle que  $f(-2) = 3.1$ ,  $f(0) = -1.7$  et  $f(5) = -5.2$  .

1. Calculer de deux façons le taux de variations de  $f$
2. Que remarquez-vous ?
3. Cette fonction  $f$  est-elle affine ?
4. Même travail avec  $f(1.2) = 2.4$ ,  $f(-2) = -4$  et  $f(3) = 6$  .

**Partie 2 :** Créer un algorithme avec ALGOBOX qui permet de savoir si une fonction  $f$  est affine ou pas connaissant trois réels et leurs images par cette fonction :  $f(a) = b$ ,  $f(c) = d$  et  $f(e) = k$ .

Principe du jeu : la calculatrice choisit au hasard un entier entre 1 et 100 et l'utilisateur doit le deviner en faisant des propositions. La calculatrice répond alors selon deux types de réponses :

- plus grand (la valeur cherchée est supérieure à la valeur proposée)
- plus petit (la valeur cherchée est inférieure à la valeur proposée)

Un compteur à l'intérieur de la boucle compte le nombre d'essais.

## Programmation TI

### DEVINE

```
:RandInt#(1,100)→N
ou
:Int(100*Rand+1)→N
:0→A
:0→I
:While A=0
:Input "VOTRE CHOIX",P
:If P=N
:Then
:1→A
:Else
:If P>N
:Then
:Disp"PLUS PETIT"
:Else
:Disp"PLUS GRAND"
:End
:End
:I+1→I
:End
:Disp"BRAVO",N
:Disp"NBRE ESSAIS",I
```

## Programmation CASIO

### DEVINE

```
RanInt#(1,100)→N↵
ou
Int(100*Ran#+1)→N↵
0→A↵
0→I↵
While A=0↵
"VOTRE CHOIX":?→P↵
If P=N↵
Then 1→A↵
Else↵
If P>N↵
Then "PLUS PETIT"↵
Else↵
"PLUS GRAND"↵
IfEnd↵
IfEnd↵
I+1→I↵
WhileEnd↵
"BRAVO":N▲
"NBRE ESSAIS":I▲
```

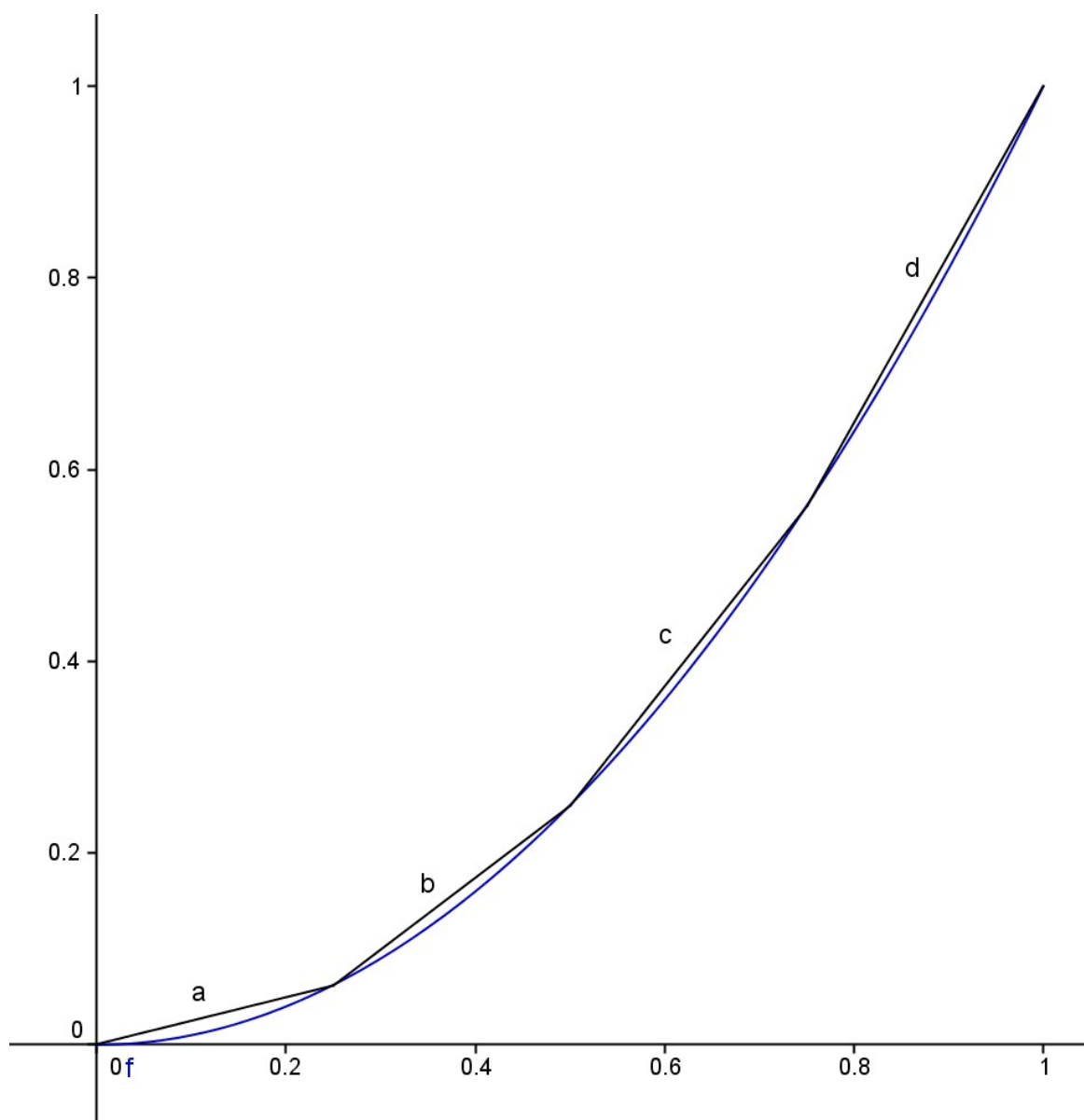
## Longueur d'une courbe.

*Activité proposée en seconde.*

### Exercice :

*On souhaite approcher la longueur d'une courbe représentant une fonction  $f$  donnée en la remplaçant par une succession de cordes.*

Exemple : On prendra la courbe représentative de la fonction carrée dans un repère orthonormé, entre les points d'abscisse 0 et 1.

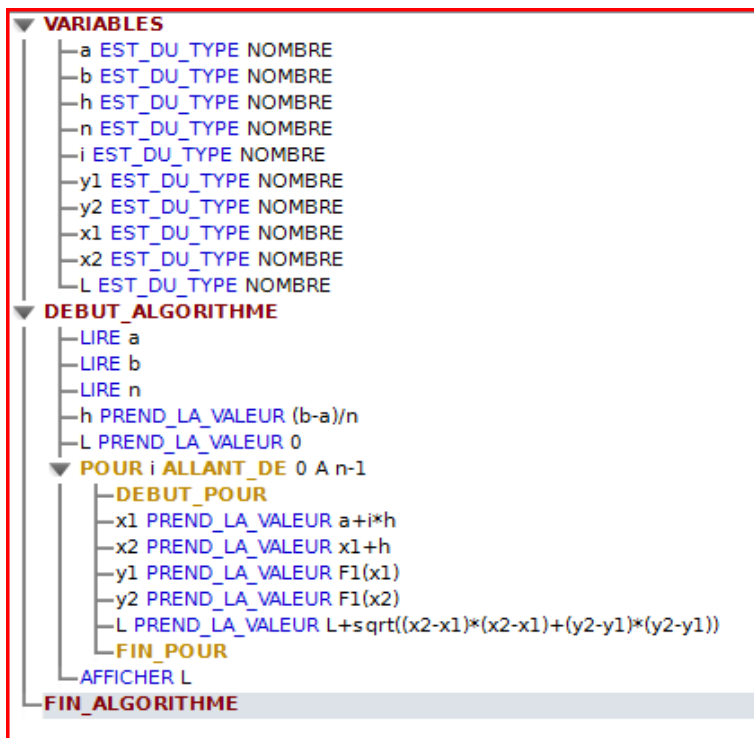


La longueur de l'arc de parabole peut être approchée par la somme des longueurs des 4 cordes a, b, c, d.

Sur le dessin ci-dessus, les 4 cordes ont pour extrémité des points de l'arc de parabole d'abscisse  $\frac{i}{4}$ ,  $i$  variant de 0 à 4.

On comprend qu'en augmentant le nombre de cordes, la précision obtenue sera meilleure.

D'où l'algorithme suivant, valable pour les fonctions continues, sur un intervalle quelconque  $[a,b]$  où la fonction  $f$  est définie.



On choisit a et b , extrémités de l'intervalle.

On choisit n, le nombre d'intervalles.

h est la largeur d'un intervalle

L est la somme des longueurs des cordes

L est initialisée à 0.

x1 et x2 sont les abscisses des extrémités d'un intervalle.

y1 et y2 sont leurs images.

Utilisation de la formule de la distance.

La dernière valeur de L est la somme des longueurs des n cordes.

La fonction F1 est définie dans l'onglet « utiliser une fonction numérique »

Ce qu'on obtient :

1°) si  $F1(x)=x^2$ ,  $a=0$ ,  $b=1$

```

***Algorithme lancé***
Entrer a : 0
Entrer b : 1
Entrer n : 1000
1.4789428
***Algorithme terminé***

```

2°) si  $F1(x)=\sqrt{1-x^2}$   $a=-1$  et  $b=1$ .

```

***Algorithme lancé***
Entrer a : -1
Entrer b : 1
Entrer n : 1000
3.1415664
***Algorithme terminé***

```

L'algorithme en langage TI :

Input A

Input B

Input N

$(B-A)/N \rightarrow H$

$0 \rightarrow L$

For(I,0,N-1)

$A+I*H \rightarrow E$

$E+H \rightarrow F$

$Y1(E) \rightarrow G$

Y1 est la fonction stockée dans la calculatrice avec la touche F1.

$Y1(F) \rightarrow K$

Pour écrire Y1 dans le programme, on fait var-Y-vars-Fonction-Y1

$L+\sqrt{(E-F)^2+(G-K)^2} \rightarrow L$

End

Disp L

## TRACER UNE COURBE POINT PAR POINT.

### *Activité proposée en seconde*

#### Exercice

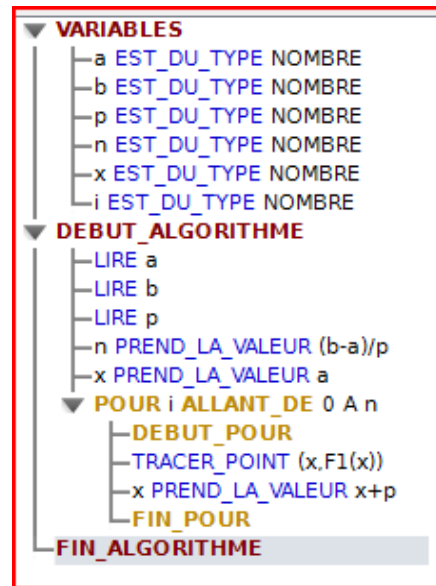
Une fonction  $f$  étant donnée ainsi qu'un intervalle  $[a;b]$ , construire point par point la courbe représentative de  $f$  sur  $[a;b]$  avec une précision  $p$ . (La précision est la différence d'abscisse de deux points consécutifs)

On peut utiliser une boucle POUR...DE ...A.

Pour cela il faut connaître le nombre  $n$  de calculs à faire. On aura ici :  $n = \frac{b-a}{p}$

On fait donc tracer tous les points de coordonnées  $(x; f(x))$  par pas de  $p$ .

Voilà ce que l'on peut obtenir avec Algobox :



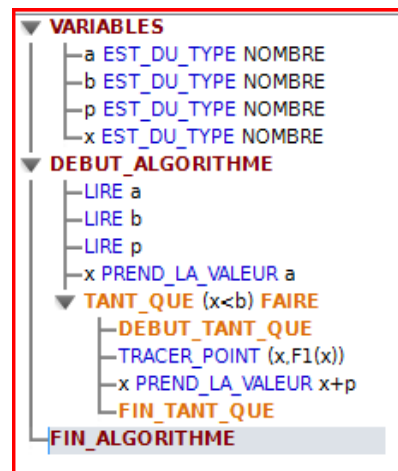
#### Remarques :

- 1) Si l'on veut diminuer le nombre de variables avec les élèves, on peut fixer  $a$ , fixer  $b$ .
- 2) la variable  $x$  est inutile, mais permet de garder  $a$  constant, ce qui est peut être plus facile à comprendre pour les élèves.

#### On peut aussi utiliser une boucle TANT QUE

Dans ce type de boucle, il est inutile à l'avance de connaître le nombre de fois que la boucle sera effectuée.

On obtient avec Algobox :



Les remarques déjà évoquées plus haut sont ici aussi valables.



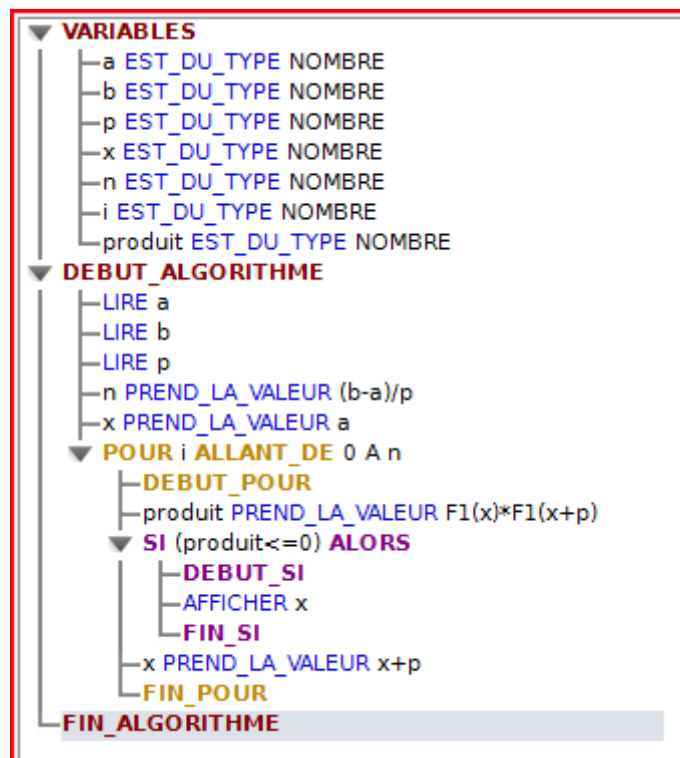
## METHODE POUR TROUVER LES SOLUTIONS DE $F(x)=0$

*Activité proposée en seconde.*

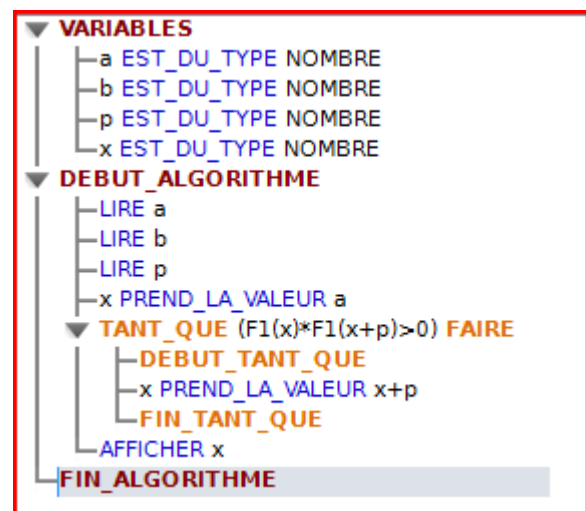
Reprenons les mêmes méthodes que précédemment pour encadrer la solution sur  $[a;b]$  d'une équation de type  $f(x)=0$ . On part de l'hypothèse que l'on a une fonction  $f$  qui s'annule une fois sur  $[a;b]$ .

On calcule les images successives des valeurs de  $x$  qui varient entre  $a$  et  $b$  par pas de  $p$ .  
Si deux images successives ont des signes différents, c'est que la solution cherchée se trouve dans l'intervalle correspondant.

Avec une boucle POUR ....DE ....A



Il est cependant préférable d'utiliser une boucle TANT ....QUE



C'est en effet beaucoup plus court.

## BOUCLES ET BOUCLES IMBRIQUEES

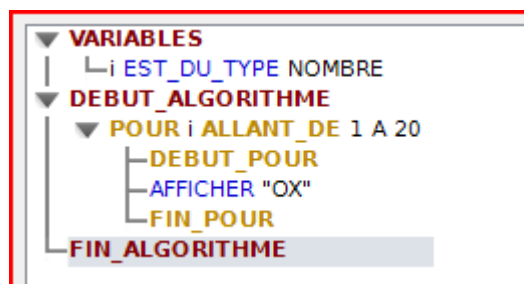
*Activité proposée en 2de*

### EXERCICE( d'après un exercice de l'IOI(olympiade internationale d'informatique))

Sur la planète Algorea, les habitants jouent à un jeu qui nécessite un plateau qui ressemble à un plateau de jeu de dames , mais de dimension 40x40. Un tel plateau de dimension 4x4 aurait cette forme :

```
OXOX
XOXO
OXOX
XOXO
```

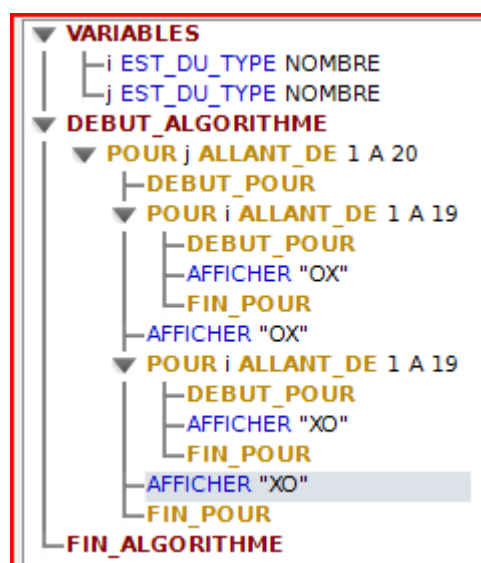
1°) Faire un algorithme avec Algobox, qui dessine la première ligne du plateau de dimension 40.



2°) Faire un algorithme avec Algobox, qui dessine le plateau 40x40 dans sa totalité.

Il va falloir recommencer 40 fois le dessin de la ligne 1, donc il va falloir mettre la boucle de l'algorithme précédent dans une autre boucle de variable j.

Mais il y a un autre problème : les lignes ne sont pas identiques, et commencent alternativement par O et X. Comment faire ? Et puis avec Algobox, il va y avoir des problèmes de retour à la ligne.



## Programmation TI

## DEGRE2

```

:Input "A=",A
:Input "B=",B
:Input "C=",C
:B^2-4*A*C→D
:Disp "DELTA=", D
:If D<0
:Then
:Disp"PAS DE SOLUTION"
:Else
:If D=0
:Then
:Disp"1 SOLUTION", (-B)/(2*A)
:Else
:Disp "2 SOLUTIONS"
:Disp "x1=", (-B-√D)/(2*A)
:Disp "x2=",(-B+√D)/(2*A)
:End
:End

```

## Programmation CASIO

## DEGRE2

```

"A=" : ? → A ↵
"B=" : ? → B ↵
"C=" : ? → C ↵
B^2-4 × A × C → D ↵
"DELTA=" : D ▲
If D<0 ↵
Then "PAS DE SOLUTION" ↵
Else ↵
If D=0 ↵
Then "1 SOLUTION" ↵
(-B) ↓ (2 × A) ▲
Else ↵
"2 SOLUTIONS" ↵
(-B-√D) ↓ (2 × A) ▲
(-B+√D) ↓ (2 × A) ▲
IfEnd ↵
IfEnd

```

## Le jeu de « Pile-Face »

Activité proposée en première S

On joue à Pile ou Face, on lance la pièce tant que les deux occurrences ne sont pas réalisées.  
On cherche, en moyenne, au bout de combien de lancers, le jeu s'arrête

Expérimentation : (travail à la maison)

Faire 10 parties en lançant une pièce et en notant le nombre de lancers nécessaires pour avoir les deux occurrences

Simulations : (en classe)

Avec la calculatrice :

Faire afficher un nombre aléatoire,  $a$ , compris entre 0 et 1, (touche "random") on convient que pour  $0 < a < 0,5$  on a Pile et que pour  $0,5 < a < 1$  on a Face. Autre méthode : la touche entAléa(0,1) et on convient que 0 correspond à Pile et 1 à Face

Faire 20 parties, noter les résultats. A quelle moyenne arrive-t-on ? (On peut regrouper les résultats de tous les élèves).

Avec un algorithme :

En décomposant les étapes de la simulation faite à la calculatrice, écrire un algorithme qui permet de simuler un nombre donné de parties et qui affiche le nombre moyen de lancers nécessaires pour obtenir les deux occurrences.

Partie théorique :

On définit et on appelle  $X$  la variable aléatoire qui compte le nombre de lancers nécessaires pour obtenir les deux occurrences.

A l'aide d'un arbre de probabilité représenter la situation donnée.

Compléter le tableau suivant qui donne la loi de la variable aléatoire  $X$  avec  $1 \leq X \leq 10$

$x_i$	1	2	3	4	5	6	7	8	9	10
$p_i$	0	$\frac{1}{2}$	$\frac{1}{4}$							

Donner la formule qui permet de calculer l'espérance de  $X$ .

En utilisant la formule de la somme des termes d'une suite géométrique et la dérivation retrouver le résultat observé.

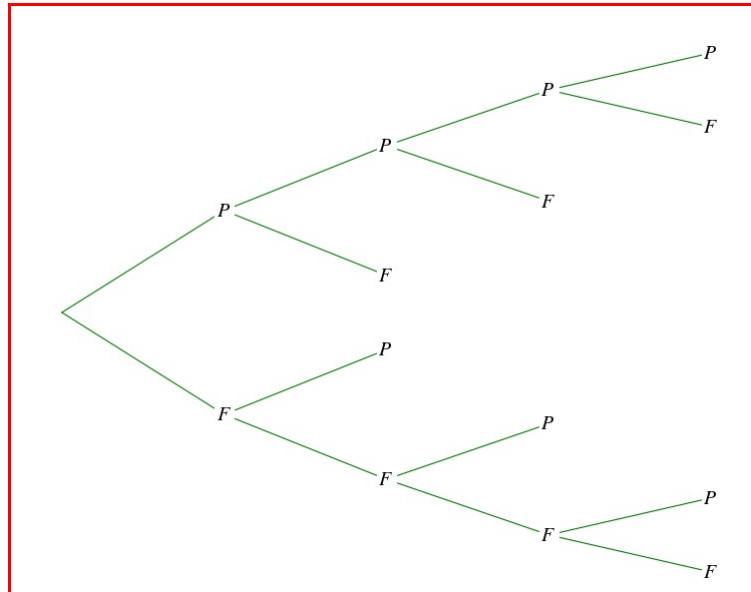
Eléments de réponse :

$$\text{On a : } E(X) = 2 \times \left(\frac{1}{2}\right)^1 + 3 \times \left(\frac{1}{2}\right)^2 + 4 \times \left(\frac{1}{2}\right)^3 + \dots + 10 \times \left(\frac{1}{2}\right)^9$$

On remarque que :

$$E(X) = (x^2 + x^3 + x^4 + \dots + x^{10})', \text{ pour } x = \frac{1}{2} \text{ or on sait que } 1 + x + x^2 + x^3 + x^4 + \dots + x^{10} = \frac{x^{11} - 1}{x - 1} \text{ en dérivant la}$$
$$\text{seconde formule on a : } E(X) = \frac{1 + 10x^{11} - 11x^{10}}{(x-1)^2} - 1 \text{ (le -1 est la dérivée de } x \text{ qui manque) et pour } x = \frac{1}{2} \text{ on a}$$
$$\text{environ } E(X) = \frac{1}{\frac{1}{4}} - 1 = 3 \text{ on peut aussi passer à la limite en faisant croître le nombre de possibilités.}$$

L'arbre utilisé :



## Un exemple de marche aléatoire.

### Activité proposée en première S

Dans le plan muni d'un repère orthonormé  $(O; \vec{i}, \vec{j})$ , un curseur situé à l'origine du repère réalise 20 déplacements sur le quadrillage de la manière suivante :

- Chaque pas correspond à un déplacement  $\vec{u} = \vec{i} + \vec{j}$  ou  $\vec{v} = \vec{i} - \vec{j}$ ,
- Les déplacements  $\vec{u}$  ou  $\vec{v}$  se font avec la même probabilité  $p = 0,5$ .

Réaliser sur une feuille quadrillée une marche possible, utiliser la touche "random" de votre calculatrice pour choisir les déplacements  $\vec{u}$  ou  $\vec{v}$  avec la probabilité de 0,5 pour chacun.

### Partie expérimentale :

On donne cet algorithme pour simuler une telle marche :

```
Nom : Marche

Variables :      i, h, p nombres entiers naturels
Initialisation : h prend la valeur 0
Traitement :     Pour i allant de 1 à 20
                  p prend la valeur 0 ou 1 avec une probabilité de 0,5
                  Si p = 0 alors
                      h prend la valeur h + 1
                      Tracer le point (i ; h)
                  Sinon
                      h prend la valeur h-1
                      Tracer le point (i ; h)
                  Fin Si
                  Fin pour
Sortie :          Afficher h
```

- 1.Expliquer comment est simulée cette marche aléatoire :  
pour ce faire, préciser le rôle de la variable  $p$  et ce que représente  $h$ .
- 2.Ecrire cet algorithme sur Algobox.
- 3.On veut maintenant savoir si le curseur atteint le point A(20 ; 0), sans refaire à chaque fois le tracé du chemin parcouru.  
Enlever et ajouter les instructions dans l'algorithme ci-dessus permettant de conclure.
- 4.On veut répéter cette marche un nombre  $n$  de fois fixé par l'utilisateur et déterminer la fréquence d'accès au point A.
  - a)Compléter l'algorithme pour obtenir cette nouvelle simulation.
  - b)Tester le pour  $n = 10$ ,  $n = 100$ ,  $n = 1000$ ,  $n = 10\,000$  puis  $n = 100\,000$ .

### Partie théorique :

Définir une variable aléatoire  $X$  qui suit une loi binomiale que vous préciserez et qui permet de calculer la probabilité que le curseur atteigne le point A.  
Comparer votre résultat avec les résultats obtenus dans la partie expérimentale.

## Les algorithmes pour les questions 3 et 4 :

Algorithme pour la question 3 proposé par un élève :

Nom : Marche

Variables :  $i, h, p$  nombres entiers naturels

Initialisation :  $h$  prend la valeur 0

Traitement : Pour  $i$  allant de 1 à 20  
                              $p$  prend la valeur 0 ou 1 avec une probabilité de 0,5  
                                     Si  $p = 0$  alors  
    $h$  prend la valeur  $h + 1$   
                                     Fin Si  
                             Fin pour

Sortie : Si  $h = 10$  alors  
                             afficher « le point A est atteint »  
                             Sinon  
                                     afficher « le point A n'est pas atteint »  
                             Fin Si

Algorithmes pour la question 4 (plus proche de celui de la partie expérimentale) :

```

1  VARIABLES
2  i EST_DU_TYPE NOMBRE
3  h EST_DU_TYPE NOMBRE
4  p EST_DU_TYPE NOMBRE
5  k EST_DU_TYPE NOMBRE
6  C EST_DU_TYPE NOMBRE
7  f EST_DU_TYPE NOMBRE
8  n EST_DU_TYPE NOMBRE
9  DEBUT_ALGORITHME
10  LIRE n
11  C PREND_LA_VALEUR 0
12  POUR k ALLANT_DE 1 A n
13  DEBUT_POUR
14  h PREND_LA_VALEUR 0
15  POUR i ALLANT_DE 1 A 20
16  DEBUT_POUR
17  p PREND_LA_VALEUR floor(0.5+random())
18  SI (p==0) ALORS
19  DEBUT_SI
20  h PREND_LA_VALEUR h+1
21  FIN_SI
22  SINON
23  DEBUT_SINON
24  h PREND_LA_VALEUR h-1
25  FIN_SINON
26  FIN_POUR
27  SI (h==0) ALORS
28  DEBUT_SI
29  C PREND_LA_VALEUR C+1
30  FIN_SI
31  FIN_POUR
32  f PREND_LA_VALEUR C/n
33  AFFICHER f
34  FIN_ALGORITHME

```

Le but de cette formule « floor(0.5+random()) » est de la généraliser à une loi binomiale : « floor( p+random()). Elle donne 1 de probabilité p et 0 de probabilité 1-p. Mais on aurait pu remplacer floor(0.5+random()) par ALGOBOX\_ALEA\_ENT(0,1)

Algorithme proposé par un élève :

Nom : Marche

Variables :  $i, h, p, k, p, C, f$  nombres entiers naturels

Entrée : Saisir  $n$

Initialisation :  $C$  prend la valeur 0

Traitement :

```
Pour  $k$  allant de 1 à  $n$ 
     $h$  prend la valeur 0
    Pour  $i$  allant de 1 à 20
         $p$  prend la valeur 0 ou 1 avec une probabilité de 0,5
        Si  $p = 0$  alors
             $h$  prend la valeur  $h + 1$ 
        Fin Si
    Fin pour
    Si  $h = 10$  alors
         $C$  prend la valeur  $C + 1$ 
    Fin Si
Fin de Pour
 $f$  prend la valeur  $C/n$ 
Afficher  $f$ 
```

Sortie :

Les algorithmes en langage TI :

Algo1 : 1 parcours

```
0→D          0 est l'ordonnée de départ du curseur
For(I,1,20)    il va faire 20 pas, son abscisse augmente de 1 à chaque pas.
EntAléat(0,1)→A  A est soit l'entier 1 soit l'entier 0
If A=0         Si A = 0
Then           alors
D+1→D         l'ordonnée du curseur augmente de 1
Else          sinon
D-1→D         l'ordonnée du curseur diminue de 1.
End
End
Disp D        D est l'ordonnée du curseur après 20 pas. Si D=0, il est en A.
```

Algo2 : 100 parcours de 10 pas

```
Input K       On entre l'ordonnée K d'arrivée.
0→C          C joue le rôle d'un compteur, initialisé à 0.
For(J,1,100)  On étudie 100 parcours du curseur
0→D          Au début de chaque parcours, l'ordonnée D est nulle.
For(I,1,10)   Le parcours fait 10 pas.
EntAléat(0,1)→A
If A=0.5
Then
D+1→D
Else
D-1→D
End
End
If D=K        Si à la fin du parcours, l'ordonnée est la valeur K choisie,
Then         Alors
C+1→C        Le compteur C augmente de 1
End
Disp C        La valeur finale de C est le nombre de parcours, sur les 100, qui finissent à
l'ordonnée K. La probabilité est alors approchée par  $C/100$ 
```



## Déplacement d'un robot sur un quadrillage

### Activité proposée en première S après la marche aléatoire.

#### Première situation :

Un robot situé sur un quadrillage muni d'un repère orthonormé  $(O, \vec{i}; \vec{j})$  se déplace aléatoirement.

Au départ le robot est en  $O$ , il peut se déplacer vers la droite avec une probabilité de 0,2, vers le haut avec une probabilité de 0,4, vers la gauche avec une probabilité de 0,3 et vers le bas avec la probabilité restante. Le robot s'arrête dès que l'une de ses coordonnées est égale à 5.

Le but de l'exercice est de modéliser cette situation à l'aide d'un algorithme, puis de faire une simulation afin de rechercher le nombre moyen de déplacements avant l'arrêt du robot.

Première partie :

Sur une feuille quadrillée, dessiner deux ou trois trajets du robot, pour cela utiliser la touche random() de votre calculatrice et des conditions sur le réel affiché pour décider du bon déplacement avec la bonne probabilité. Noter en combien d'étapes vous arrivez à l'arrêt du robot.

Deuxième partie :

On note  $(x; y)$  les coordonnées du robot.

–Démontrer qu'une condition pour arrêter le robot est :  $(x^2 - 25)(y^2 - 25) = 0$

–Déterminer les variables à utiliser

–Déterminer les tests à utiliser pour distinguer les différents cas.

–Réfléchir à la possibilité de dessiner le trajet du robot, et le traduire sur Algobox( algo1, page 34).

–Faire fonctionner cet algorithme

Troisième partie :

On veut réaliser un grand nombre d'essais pour trouver le nombre moyen de déplacements avant l'arrêt du robot.

–Déterminer les nouvelles variables à utiliser, et celles à supprimer.

–Modifier l'algorithme 1 pour compter le nombre moyen de déplacements avant l'arrêt du robot et en déduire la moyenne. ( Algo2, page 35)

#### Seconde situation :

On arrête le robot non plus à 5 mais à 2 et chaque direction a la même probabilité, à savoir  $\frac{1}{4}$ .

Refaire un nouvel algorithme qui permette de simuler "n" parties et en déduire la moyenne.

Modéliser la situation par un arbre de probabilité.

Soit  $X$  la variable aléatoire qui compte le nombre de déplacements du robot jusqu'à ce que le robot s'arrête.

Déterminer la loi de  $X$  en complétant ce tableau :

$x_i$	1	2	3	4	5	6	7	8	9	10
$p_i$	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{8}$	$\frac{1}{8}$					

Déterminer la formule qui permet d'obtenir l'espérance de  $X$ .

A l'aide d'un nouvel algorithme et/ou d'un tableur trouver le nombre moyen de déplacements avant l'arrêt du robot, on pourra considérer que l'approximation obtenue pour  $X \leq 100$  est suffisante.

## Modélisation :

Les positions du robot sont les suivantes :

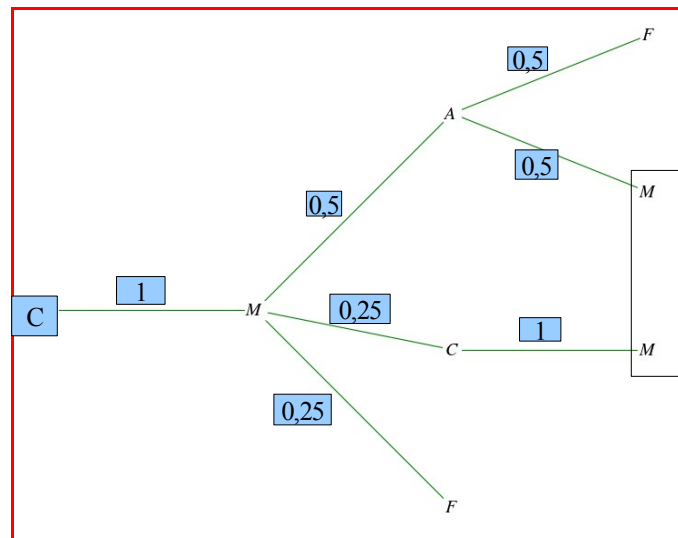
C au centre,  
M au milieu d'une arête,  
A à l'angle,  
F c'est la fin. :

A	M	A
M	C	M
A	M	A

Le robot va :

- de C en M avec une probabilité de 1,
- de M en C avec une probabilité de 0,25, de M en A avec une probabilité de 0,5 et de M à l'arrêt avec une probabilité de 0,25,
- de A en M avec une probabilité de 0,5 et de M à l'arrêt avec une probabilité de 0,5

Arbre utilisé :



Toutes les deux fois, on repart sur M et on recommence avec une probabilité divisée par 2.

## Algorithme 1

```

1  VARIABLES
2    x EST_DU_TYPE NOMBRE
3    y EST_DU_TYPE NOMBRE
4    a EST_DU_TYPE NOMBRE
5    n EST_DU_TYPE NOMBRE
6    mx EST_DU_TYPE NOMBRE
7    my EST_DU_TYPE NOMBRE
8  DEBUT_ALGORITHME
9    x PREND_LA_VALEUR 0
10   y PREND_LA_VALEUR 0
11   n PREND_LA_VALEUR 0
12   TANT_QUE
((x-5)*(x+5)*(y-5)*(y+5)!=0) FAIRE
13     DEBUT_TANT_QUE
14       a PREND_LA_VALEUR random()
15       mx PREND_LA_VALEUR x
16       my PREND_LA_VALEUR y
17       TRACER_POINT (x,y)
18       SI (a<0.2) ALORS
19         DEBUT_SI
20           x PREND_LA_VALEUR x+1
21         FIN_SI
22       SI (a>0.2 ET a<0.6) ALORS
23         DEBUT_SI
24           y PREND_LA_VALEUR y+1
25         FIN_SI
26       SI (a>0.6 ET a<0.9) ALORS
27         DEBUT_SI
28           x PREND_LA_VALEUR x-1
29         FIN_SI
30       SI (a>0.9) ALORS
31         DEBUT_SI
32           y PREND_LA_VALEUR y-1
33         FIN_SI
34       TRACER_SEGMENT (mx,my)->(x,y)
35       n PREND_LA_VALEUR n+1
36       FIN_TANT_QUE
37       AFFICHER "Le robot est au bord
au bout de "
38       AFFICHER n
39       AFFICHER " déplacements"
40     FIN_ALGORITHME

```

## Algorithme 2

```
1  VARIABLES
2    x EST_DU_TYPE NOMBRE
3    y EST_DU_TYPE NOMBRE
4    a EST_DU_TYPE NOMBRE
5    n EST_DU_TYPE NOMBRE
6    nbTrajets EST_DU_TYPE NOMBRE
7    i EST_DU_TYPE NOMBRE
8    nbMoyenDeplacements EST_DU_TYPE NOMBRE
9  DEBUT_ALGORITHME
10   n PREND_LA_VALEUR 0
11   LIRE nbTrajets
12   POUR i ALLANT_DE 1 A nbTrajets
13     DEBUT_POUR
14     x PREND_LA_VALEUR 0
15     y PREND_LA_VALEUR 0
16     TANT_QUE ((x-5)*(x+5)*(y-5)*(y+5) != 0) FAIRE
17       DEBUT_TANT_QUE
18       a PREND_LA_VALEUR random()
19       SI (a < 0.2) ALORS
20         DEBUT_SI
21         x PREND_LA_VALEUR x+1
22         FIN_SI
23       SI (a > 0.2 ET a < 0.6) ALORS
24         DEBUT_SI
25         y PREND_LA_VALEUR y+1
26         FIN_SI
27       SI (a > 0.6 ET a < 0.9) ALORS
28         DEBUT_SI
29         x PREND_LA_VALEUR x-1
30         FIN_SI
31       SI (a > 0.9) ALORS
32         DEBUT_SI
33         y PREND_LA_VALEUR y-1
34         FIN_SI
35     n PREND_LA_VALEUR n+1
36   FIN_TANT_QUE
37   FIN_POUR
38   nbMoyenDeplacements PREND_LA_VALEUR n/nbTrajets
39   AFFICHER "le nombre moyen de déplacements par trajets est "
40   AFFICHER nbMoyenDeplacements
41   AFFICHER " déplacements"
42  FIN_ALGORITHME
```

## Éléments de réponse pour la seconde situation

### Un algorithme possible pour la seconde situation

```

Code de l'algorithme

VARIABLES
x EST_DU_TYPE NOMBRE
y EST_DU_TYPE NOMBRE
a EST_DU_TYPE NOMBRE
n EST_DU_TYPE NOMBRE
mx EST_DU_TYPE NOMBRE
my EST_DU_TYPE NOMBRE
nbexp EST_DU_TYPE NOMBRE
k EST_DU_TYPE NOMBRE
nbcoup EST_DU_TYPE NOMBRE
moy EST_DU_TYPE NOMBRE

DEBUT_ALGORITHME
  LIRE nbexp
  POUR k ALLANT_DE 1 A nbexp
    DEBUT_POUR
      x PREND_LA_VALEUR 0
      y PREND_LA_VALEUR 0
      n PREND_LA_VALEUR 0
      TANT_QUE ((x-2)*(x+2)*(y-2)*(y+2)!=0) FAIRE
        DEBUT_TANT_QUE
          a PREND_LA_VALEUR floor(4*random()+1)
          mx PREND_LA_VALEUR x
          my PREND_LA_VALEUR y
          SI (a==1) ALORS
            DEBUT_SI
              y PREND_LA_VALEUR y+1
            FIN_SI
          SINON
            DEBUT_SINON
              SI (a==2) ALORS
                DEBUT_SI
                  x PREND_LA_VALEUR x-1
                FIN_SI
              SINON
                DEBUT_SINON
                  SI (a==3) ALORS
                    DEBUT_SI
                      y PREND_LA_VALEUR y-1
                    FIN_SI
                  SINON
                    DEBUT_SINON
                      x PREND_LA_VALEUR x+1
                    FIN_SINON
                FIN_SINON
              FIN_SINON
            FIN_SINON
          FIN_TANT_QUE
          TRACER_POINT (x,y)
          TRACER_SEGMENT (mx,my)->(x,y)
          n PREND_LA_VALEUR n+1
        FIN_TANT_QUE
        nbcoup PREND_LA_VALEUR nbcoup+n
      FIN_POUR
      moy PREND_LA_VALEUR nbcoup/nbexp
    AFFICHER "En moyenne, le robot est au bord au bout de "
    AFFICHER moy
    AFFICHER " déplacements"
  FIN_ALGORITHME
  
```

### Pour le calcul de l'espérance

```

Code de l'algorithme

VARIABLES
E EST_DU_TYPE NOMBRE
n EST_DU_TYPE NOMBRE
p EST_DU_TYPE NOMBRE

DEBUT_ALGORITHME
  n PREND_LA_VALEUR 2
  p PREND_LA_VALEUR 2
  TANT_QUE (n<=100) FAIRE
    DEBUT_TANT_QUE
      E PREND_LA_VALEUR E+(2*n+1)*(1/pow(2,p))
      n PREND_LA_VALEUR n+2
      p PREND_LA_VALEUR p+1
    FIN_TANT_QUE
  AFFICHER E
FIN_ALGORITHME
  
```

### Avec un tableur, (les premières lignes)

	A	B	C	D
1	rang	proba	xi pi	E
2				
3	1	0,000000000000	0,000000000000	4,5
4	2	0,250000000000	0,500000000000	
5	3	0,250000000000	0,750000000000	
6	4	0,125000000000	0,500000000000	
7	5	0,125000000000	0,625000000000	
8	6	0,062500000000	0,375000000000	
9	7	0,062500000000	0,437500000000	
10	8	0,031250000000	0,250000000000	
11	9	0,031250000000	0,281250000000	
12	10	0,015625000000	0,156250000000	
13	11	0,015625000000	0,171875000000	
14	12	0,007812500000	0,093750000000	
15	13	0,007812500000	0,101562500000	
16	14	0,003906250000	0,054687500000	
17	15	0,003906250000	0,058593750000	
18	16	0,001953125000	0,031250000000	
19	17	0,001953125000	0,033203125000	
20	18	0,000976562500	0,017578125000	
21	19	0,000976562500	0,018554687500	
22	20	0,000488281250	0,009765625000	
23	21	0,000488281250	0,010253906250	
24	22	0,000244140625	0,005371093750	
25	23	0,000244140625	0,005615234375	
26	24	0,000122070313	0,002929687500	
27	25	0,000122070313	0,003051757813	
28	26	0,000061035156	0,001586914063	
29	27	0,000061035156	0,001647949219	
30	28	0,000030517578	0,000854492188	

## Les records dans une suite de nombres

Activité proposée en première S

### EXERCICE :

Créer une suite de 100 entiers aléatoires.

Déterminer dans cette suite, les nombres, appelés « records », ceux qui sont plus grands que tous les nombres qui les précèdent.

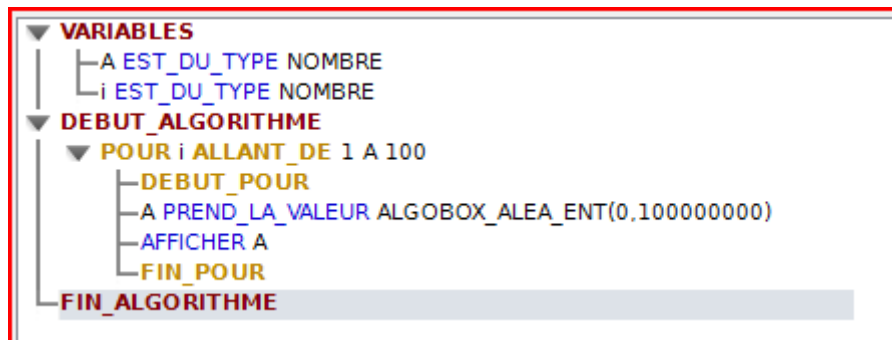
Déterminer pour un rang donné, la probabilité que le terme de ce rang soit un record.

Evaluer le nombre moyen de records dans la suite.

Exemple : dans la suite suivante, les records sont soulignés.

7 11 8 2 15 51 16 21 41 65 35 84

1) Créer une suite finie de 100 nombres aléatoires entre 0 et 100000000. Afficher chaque terme de cette suite.



Voilà l'algorithme en langage TI :

FOR(I,1,100)

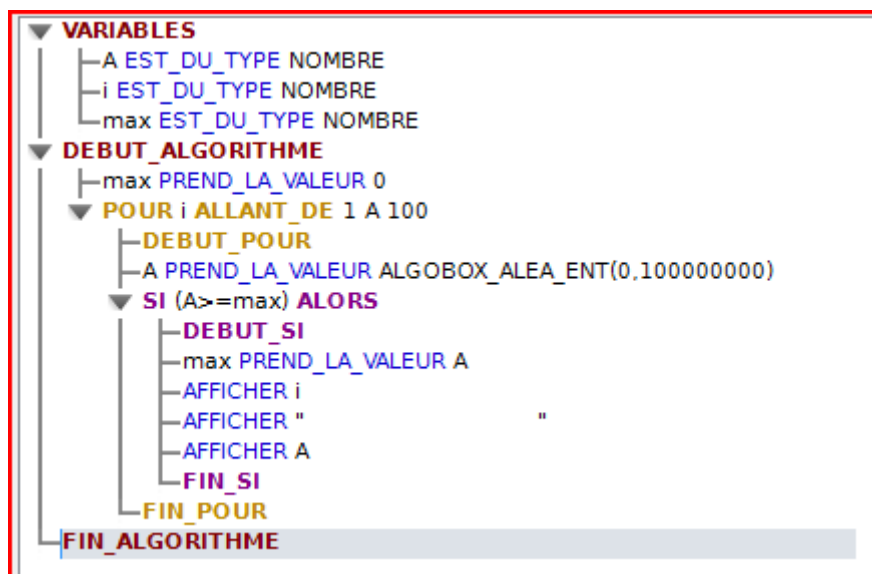
NbrAléat→A

Ici A est un réel entre 0 et 1. Le principe reste le même.

Disp A

End

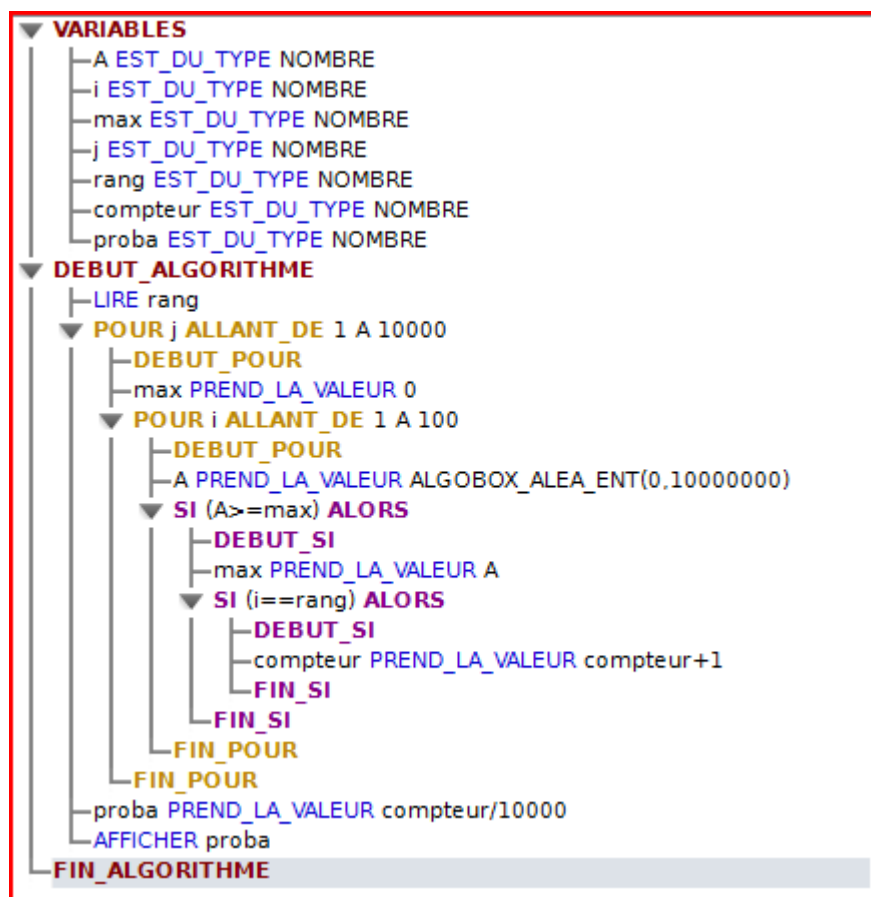
2) Repérer dans cette suite tous les nombres qui sont plus grands que ceux qui les précèdent. Ces nombres sont appelés des records. Afficher les nombres de la suite qui sont des records, ainsi que leur rang dans la suite. Pour repérer les records on utilise une variable auxiliaire max, qui augmente chaque fois qu'un nombre de la suite dépasse max.



Voila l'algorithme en langage TI :

0→M	M initialisée à 0
FOR(I,1,100)	
NbrAléat→A	
If A>M	M est le record,
Then	
A→M	M augmente au fur et à mesure que les valeurs de A sont calculées
Disp A	affichage de A seulement s'il est un record
Disp I	affichage du rang de A dans la suite .
Pause	Pause pour éviter un défilement rapide.
End	
End	

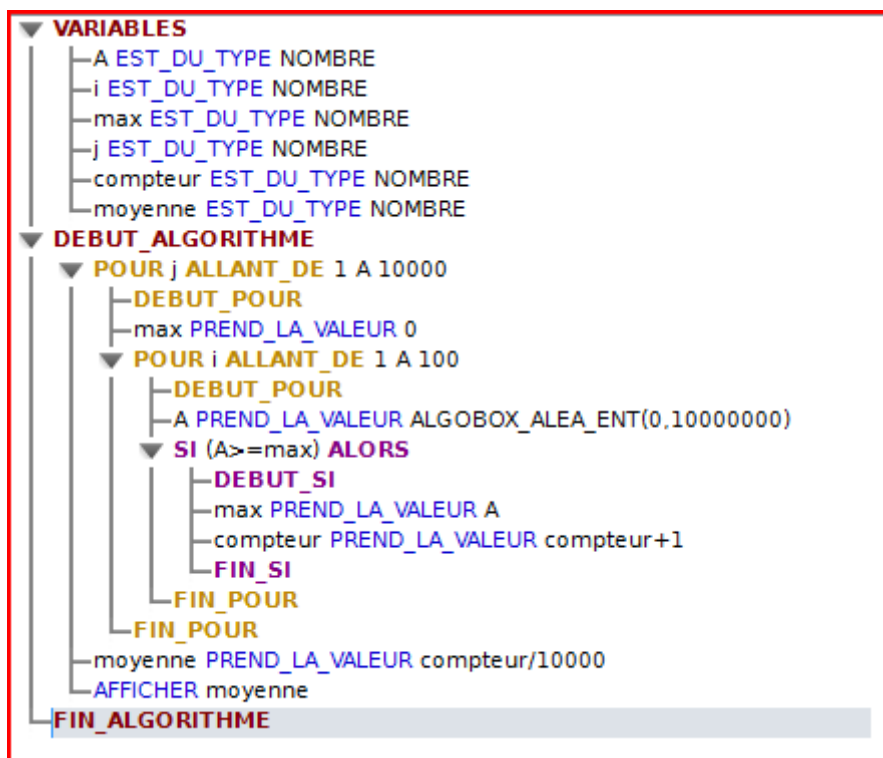
3) En répétant le tirage des 100 nombres un très grand nombre de fois, évaluer la probabilité que le le terme de rang i soit un record. Faire varier la variable « rang » de l'algorithme, pour essayer de trouver une formule pour cette probabilité.  
Pouvait-on établir cette probabilité sans expérimentation ?



Voila l'algorithme en langage TI : avec des paramètres changés à cause de la faible puissance du processeur.

INPUT R	R est le rang choisi
0→C	C est un compteur initialisé à 0
For(J,1,100)	On va tirer 100 fois une suite (et non 10000)
0→M	
For(I,1,25)	c'est une suite de 25 nombres ( et non pas 100)
NbrAléat→A	
if A>M	
Then	
A→M	
If I=R	Si le terme de rang I est un record et si I=R
Then	alors
C+1→C	le compteur augmente de 1
End	
End	
End	
End	
Disp C	C est le nombre de fois que le terme de rang R est un record sur les 100 suites testées.

5) On voudrait savoir combien on obtient de records, en moyenne, lorsque on tire aléatoirement une suite de 100 nombres.



Voila l'algorithme en langage TI :

0→C	C est un compteur initialisé à 0, qui va compter les records
For(J,1,100)	On va tirer 100 fois une suite( et non 10000)
0→M	Pour chaque suite, M est réinitialisé à 0.
For(I,1,25)	c'est une suite de 25 nombres ( et non pas 100)
NbrAléat→A	
if A>M	Si A est un record
Then	alors
A→M	M prend la valeur de A
C+1→C	le compteur augmente de 1
End	
End	
End	
Disp C	C est le nombre de records obtenu pour 100 suites de 25 nombres. En moyenne , pour une suite, cela fait C/100.

Pouvait -on prévoir le résultat, trouver une formule ?

Réponses théoriques :

Réponse à la question du paragraphe 3 : Appelons  $a_n$  la suite de nombres. Considérons le terme de rang  $i$  de la suite, soit  $a_i$ . Ce terme sera un record s'il est le plus grand des nombres parmi la liste  $\{ a_1; a_2; a_3; \dots; a_i \}$

Comme ces  $i$  nombres sont aléatoires, chacun, et en particulier  $a_i$  a une probabilité d'être le plus grand égale à  $\frac{1}{i}$ . Donc la probabilité que le terme de rang  $i$  soit un record est  $\frac{1}{i}$ .

Réponse à la question du paragraphe 5.

Soit  $X_i$  la variable aléatoire qui vaut 1 si le terme de rang  $i$  de la suite de nombres est un record, et qui vaut 0 sinon.

On considère les 100 variables aléatoires indépendantes  $(X_i)_{1 \leq i \leq 100}$ . On a

$$E(X_i) = \frac{1}{i} \times 1 + \left(1 - \frac{1}{i}\right) \times 0 = \frac{1}{i}$$

Le nombre moyen de records dans la liste de 100 nombres est l'espérance de la somme des variables aléatoires soit

$$E(X_1 + X_2 + \dots + X_{100}) = E(X_1) + E(X_2) + \dots + E(X_{100}) = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{100} \approx 5,1873775$$



## EXTRAITS D'EXERCICES POSES AU BACALAUREAT

### Exercice de bac n°1 Métropole Juin 2012

On rappelle que  $(u_n)$  est une suite définie pour tout entier strictement positif par

$$u_n = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n} - \ln n.$$

Modifiez l'algorithme suivant, afin qu'il calcule la valeur de  $u_n$ .

Variables :	$k$ et $n$ sont des entiers naturels. $S$ est un réel.
Entrée :	Demander à l'utilisateur la valeur de $n$ .
Initialisation :	Affecter à $S$ la valeur 0.
Traitement :	Pour $k$ variant de 1 à $n$ . <div style="border-left: 1px solid black; padding-left: 10px; margin-left: 20px;">Affecter à <math>S</math> la valeur <math>S + \frac{1}{k}</math>.</div>
Sortie :	Afficher $S$ .

Votre algorithme, devra lire un entier strictement positif, la valeur de  $n$ , et afficher un nombre réel, la valeur de  $u_n$ .

Tester l'algorithme obtenu sur Algobox ou calculatrice.

### Exercice de bac n°2 Antilles Guyane Juin 2012( spécialité maths)

4. On considère l'algorithme suivant :

```

A et N sont des entiers naturels
Saisir A
N prend la valeur 1
Tant que N ≤ √A
    Si A ≡ 0 mod N alors
        Afficher N et A/N
    Fin si
    N prend la valeur N + 1
Fin Tant Que
    
```

- a. Quels résultats affiche cet algorithme pour  $A = 12$  ?
- b. Que donne cet algorithme dans le cas général ?
- c. **Non demandé au BAC**

Dans certains cas (par exemple pour  $A = 4$ ), cet algorithme affiche deux fois certains nombres. Modifiez le pour qu'il affiche exactement les mêmes nombres mais jamais plusieurs fois le même.

Tester éventuellement cet algorithme avec Algobox ou calculatrice.

### Exercice de bac n°3 Antilles guyane juin 2012

5. On considère l'algorithme :

```

A et C sont des entiers naturels,
C prend la valeur 0
Répéter 9 fois
    A prend une valeur aléatoire entière entre 1 et 7.
    Si A > 5 alors C prend la valeur de C + 1
    Fin Si
Fin répéter
Afficher C.
    
```

Dans l'expérience aléatoire simulée par l'algorithme précédent, on appelle  $X$  la variable aléatoire prenant la valeur  $C$  affichée.

Quelle loi suit la variable  $X$ ? Préciser ses paramètres.

### Exercice de bac n°4 Polynésie juin 2012

#### Partie A

On considère l'algorithme suivant :

Les variables sont le réel  $U$  et les entiers naturels  $k$  et  $N$ .

#### Entrée

Saisir le nombre entier naturel non nul  $N$ .

#### Traitement

```
Affecter à  $U$  la valeur 0
Pour  $k$  allant de 0 à  $N - 1$ 
    Affecter à  $U$  la valeur  $3U - 2k + 3$ 
Fin pour
```

#### Sortie

Afficher  $U$

Quel est l'affichage en sortie lorsque  $N = 3$  ?

### Exercice n°5 Pondichéry Avril 2012

#### Partie A

Un groupe de 50 coureurs, portant des dossards numérotés de 1 à 50, participe à une course cycliste qui comprend 10 étapes, et au cours de laquelle aucun abandon n'est constaté.

À la fin de chaque étape, un groupe de 5 coureurs est choisi au hasard pour subir un contrôle antidopage. Ces désignations de 5 coureurs à l'issue de chacune des étapes sont indépendantes. Un même coureur peut donc être contrôlé à l'issue de plusieurs étapes.

1. À l'issue de chaque étape, combien peut-on former de groupes différents de 5 coureurs ?
2. On considère l'algorithme ci-dessous dans lequel  $\text{rand}(1, 50)$  permet d'obtenir un nombre entier aléatoire appartenant à l'intervalle  $[1; 50]$ .

<b>Variables</b>	$a, b, c, d, e$ sont des variables du type entier
<b>Initialisation</b>	Affecter à $a$ la valeur 0. Affecter à $b$ la valeur 0. Affecter à $c$ la valeur 0. Affecter à $d$ la valeur 0. Affecter à $e$ la valeur 0.
<b>Traitement</b>	Tant que $(a = b)$ ou $(a = c)$ ou $(a = d)$ ou $(a = e)$ ou $(b = c)$ ou $(b = d)$ ou $(b = e)$ ou $(c = d)$ ou $(c = e)$ ou $(d = e)$ Début du tant que Affecter à $a$ la valeur $\text{rand}(1, 50)$ Affecter à $b$ la valeur $\text{rand}(1, 50)$ Affecter à $c$ la valeur $\text{rand}(1, 50)$ Affecter à $d$ la valeur $\text{rand}(1, 50)$ Affecter à $e$ la valeur $\text{rand}(1, 50)$ Fin du tant que
<b>Sortie</b>	Afficher $a, b, c, d, e$

- a. Parmi les ensembles de nombres suivants, lesquels ont pu être obtenus avec cet algorithme :

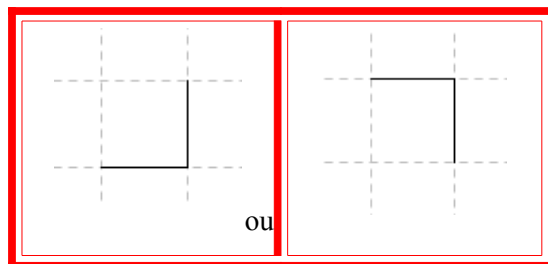
- $L_1 = \{2, 11, 44, 2, 15\}$
- $L_2 = \{8, 17, 41, 34, 6\}$
- $L_3 = \{12, 17, 23, 17, 50\}$
- $L_4 = \{45, 19, 43, 21, 18\}$

- b. Que permet de réaliser cet algorithme concernant la course cycliste ?

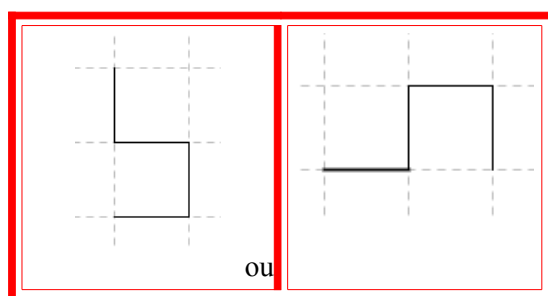
## La courbe du « Dragon », une fractale obtenue par pliages successifs.

### Activité post-bac

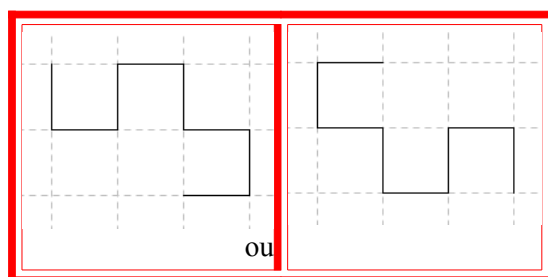
Prendre une feuille, la plier en deux et l'ouvrir à demi, regarder "la tranche" vous devez voir :



Refermer et replier en deux, dans le même sens, ouvrir "à demi" vous devez voir :



Refermer et replier en deux, dans le même sens, ouvrir "à demi" vous devez voir :



Et si vous pouviez faire vingt pliages, que verriez-vous ? (Le dessin ne tenant pas compte qu'à chaque pliage la dimension de chaque segment est divisé en deux, on garde la même taille tout au long du tracé de la courbe).

Peut-être qu'un algorithme pourrait vous aider ! Si on regarde bien, soit on "tourne" à gauche, soit on tourne à droite, reste à savoir quand.

Une aide, tout entier  $n$  non-nul peut s'écrire :  $n = 2^p \times k$  où  $k$  désigne un nombre impair et  $p$  le plus grand possible.

La fractale obtenue s'appelle la courbe du "Dragon".

Remerciements à Jean Claude Loulmet professeur au Lycée Guez de Balzac pour cette idée.

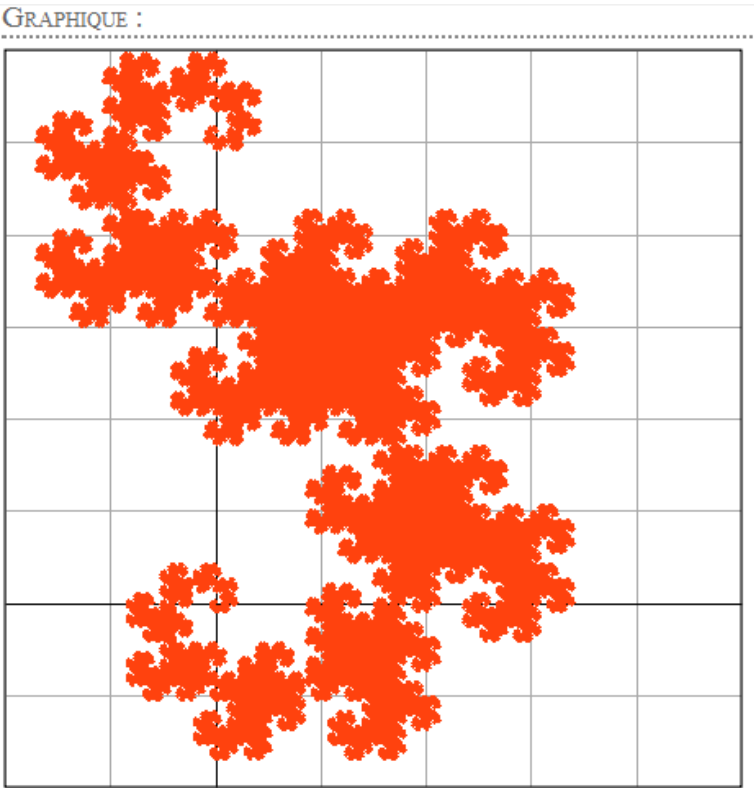
Le début de la recherche :

Numéro du pliage	1	2	3	4	5	6	7	8	9	10	11	12
Décomposition	$2^0 \times 1$	$2^1 \times 1$	$2^0 \times 3$	$2^2 \times 1$	$2^0 \times 5$	$2^1 \times 3$	$2^0 \times 7$	$2^3 \times 1$	$2^0 \times 9$	$2^1 \times 5$	$2^0 \times 11$	$2^2 \times 3$
Droite ou gauche	D	D	G	D	D	G	G	D	D	D	G	G

Si on observe bien, à chaque fois que le reste de la division par 4 de  $k$  est 1 on tourne dans un sens et à chaque fois que le reste est 3 on tourne dans l'autre sens.  
On suppose que ce résultat est toujours vrai, on peut observer que les séquences 1 et 3 sont inversées, 1,2, 3 et 5,6,7 le sont aussi et ainsi de suite, on pourrait envisager une récurrence pour mieux expliquer le résultat.

On a donc tous les éléments pour construire un algorithme.

En voici un qui donne la courbe pour un nombre de plis au plus égal à  $2^{18}$  et voici le résultat :



Maintenant, à vous !

Une solution possible au problème, ce n'est certainement pas la seule et il doit y avoir plus simple.

Détails de l'algorithme :

#### ▼ VARIABLES

```
├-xa EST_DU_TYPE NOMBRE
├-xb EST_DU_TYPE NOMBRE
├-ya EST_DU_TYPE NOMBRE
├-yb EST_DU_TYPE NOMBRE
├-n EST_DU_TYPE NOMBRE
├-fi EST_DU_TYPE NOMBRE
├-rfipq EST_DU_TYPE NOMBRE
└-NB EST_DU_TYPE NOMBRE
```

NB est le nombre de plis choisi.

n est le numéro du pli

fi est le facteur impair de n

rfipq est le reste de la division de fi par quatre

les autres variables sont les coordonnées des extrémités des segments successifs.

#### ▼ DEBUT\_ALGORITHME

```
├-xa PREND_LA_VALEUR 0
├-xb PREND_LA_VALEUR 1
├-ya PREND_LA_VALEUR 0
├-yb PREND_LA_VALEUR 0
├-LIRE NB
└-POUR n ALLANT_DE 1 A NB
    └-DEBUT_POUR
        └-TRACER_SEGMENT (xa,ya)->(xb,yb)
        └-fi PREND_LA_VALEUR n
        └-TANT_QUE (fi%2==0) FAIRE
            └-DEBUT_TANT_QUE
                └-fi PREND_LA_VALEUR fi/2
            └-FIN_TANT_QUE
        └-rfipq PREND_LA_VALEUR fi%4
```

Le début de l'algorithme consiste à donner les extrémités du premier segment, à lire le nombre NB de plis souhaité.

Ensuite on entre dans une boucle qui a pour but de tracer les NB segments en tournant correctement suivant le numéro  $n$  du pli.

Cette opération est assurée par le calcul du reste par 4 du facteur impair de la décomposition de  $n$ .

#### ▼ SI (xb==xa+1 ET rfipq==1) ALORS

```
├-DEBUT_SI
├-xa PREND_LA_VALEUR xb
├-ya PREND_LA_VALEUR yb
├-yb PREND_LA_VALEUR yb+1
├-TRACER_SEGMENT (xa,ya)->(xb,yb)
└-FIN_SI
```

On entre alors dans une série de test pour savoir la direction empruntée à l'avant dernier pli et en fonction de cette direction et du reste de la division on va modifier les coordonnées des extrémités et tracer le nouveau segment.

#### ▼ SINON

##### └-DEBUT\_SINON

#### ▼ SI (xb==xa+1 ET rfipq==3) ALORS

```
├-DEBUT_SI
├-xa PREND_LA_VALEUR xb
├-ya PREND_LA_VALEUR yb
├-yb PREND_LA_VALEUR yb-1
├-TRACER_SEGMENT (xa,ya)->(xb,yb)
└-FIN_SI
```

#### ▼ SINON

##### └-DEBUT\_SINON

Cette séquence de tests est répétée quatre fois :

→  $x_B = x_A + 1$  : segment horizontal dirigé vers la droite,

→  $x_B = x_A - 1$  : segment horizontal dirigé vers la gauche,

→  $y_B = y_A + 1$  : segment vertical dirigé vers le haut,

→  $y_B = y_A - 1$  : segment vertical dirigé vers le bas.