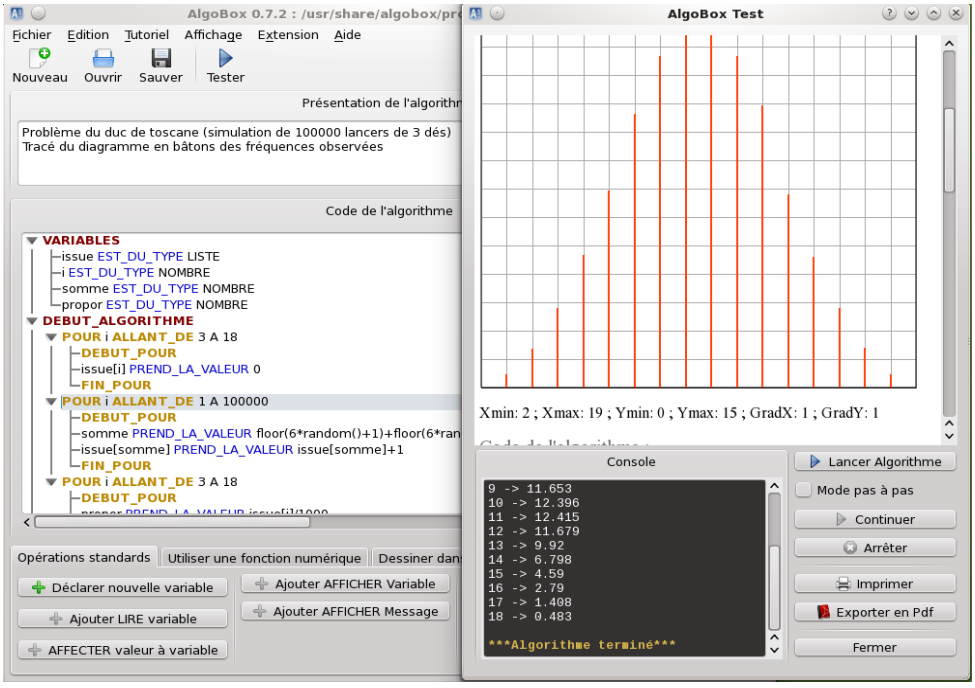
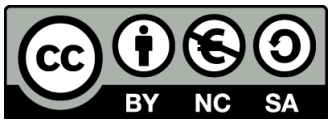
Algorithmique en classe de première avec AlgoBox



Version 1.3 - Août 2017



Cette brochure a été réalisée avec le système de composition LaTeX et l'éditeur TeXMAKER.
http://www.xm1math.net/doculatex/index.html

Sommaire

Avant-propos

Rappel des instructions officielles concernant l'algorithmique dans les programmes de mathématiques :

1. ***Instructions élémentaires (affectation, calcul, entrée, sortie).***
   Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables :
   — *d'écrire une formule permettant un calcul ;*
   — *d'écrire un programme calculant et donnant la valeur d'une fonction ;*
   — *ainsi que les instructions d'entrées et sorties nécessaires au traitement.*
2. ***Boucle et itérateur, instruction conditionnelle.***
   Les élèves, dans le cadre d'une résolution de problèmes, doivent être capables de :
   — *programmer un calcul itératif, le nombre d'itérations étant donné ;*
   — *programmer une instruction conditionnelle, un calcul itératif, avec une fin de boucle conditionnelle.*
3. ***Dans le cadre de cette activité algorithmique, les élèves sont entraînés à :***
   — *décrire certains algorithmes en langage naturel ou dans un langage symbolique ;*
   — *en réaliser quelques-uns à l'aide d'un tableur ou d'un programme sur calculatrice ou avec un logiciel adapté ;*
   — *interpréter des algorithmes plus complexes.*

Contenu de cette brochure :

— Des activités « élèves » strictement conformes aux programmes en vigueur.
— Des annexes comportant :
  — Des activités d'apprentissage des techniques de base en algorithmique avec Algobox ;
  — Un mémento sur les fonctions d'AlgoBox ;
  — Des algorithmes supplémentaires en rapport avec le contenu mathématique des programmes de première.

À propos des activités « élèves » :
Les fiches « professeurs » et « élèves » sont sur des pages différentes afin de faciliter les photocopies.
Les activités sont présentées ici sous forme d'énoncés « à trou ». Il est bien sur possible de les adapter selon sa convenance.
Adaptations possibles :
— donner l'algorithme complet et demander de décrire ce qu'il fait ;
— demander la réalisation complète de l'algorithme à partir de zéro.

Les fichiers AlgoBox des algorithmes de la partie « Activités élèves » et de l'annexe C sont disponibles en ligne à l'adresse suivante : `http://www.xm1math.net/algobox/algobook.html`

Première partie

Activités « élèves »

1 Pourcentages

Fiche professeur 1A

— Fiche élève correspondante : page 3
— *Fichier AlgoBox associé (algorithme complet) :* `algo_1A.alg`
— *Contexte (1ES/1STMG) :* Application directe du cours sur les hausses en pourcentage

Fiche professeur 1B

— Fiche élève correspondante : page 3
— *Fichier AlgoBox associé (algorithme complet) :* `algo_1B.alg`
— *Contexte (1ES/1STMG) :* Recherche du nombre de hausses nécessaires pour doubler la valeur d'une grandeur (algorithmique « utile » - résolution mathématique directe impossible en première) - Utilisation d'une boucle `TANT_QUE`

Fiche élève 1A

Compléter la ligne 8 pour que l'algorithme AlgoBox ci-dessous soit correct :

```
1: VARIABLES
2: prixHT EST_DU_TYPE NOMBRE
3: prixTTC EST_DU_TYPE NOMBRE
4: TVA EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:     LIRE prixHT
7:     LIRE TVA
8:     prixTTC PREND_LA_VALEUR prixHT*(..............)
9:     AFFICHER "Le prix TTC est égal à "
10:    AFFICHER prixTTC
11: FIN_ALGORITHME
```

Fiche élève 1B

Le PIB d'un pays émergent est de 700 milliards d'euros en 2010. Selon un modèle de prévision, il devrait augmenter de 10% par an dans les années futures.
On cherche à créer un algorithme AlgoBox qui permette de déterminer en quelle année le PIB aura doublé par rapport à 2010.
Compléter les lignes 7 et 9 ci-dessous pour que l'algorithme proposé réponde à la question.

```
1: VARIABLES
2: annee EST_DU_TYPE NOMBRE
3: PIB EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:     PIB PREND_LA_VALEUR 700
6:     annee PREND_LA_VALEUR 2010
7:     TANT_QUE (PIB.............) FAIRE
8:         DEBUT_TANT_QUE
9:         PIB PREND_LA_VALEUR PIB*...........
10:        annee PREND_LA_VALEUR annee+1
11:        FIN_TANT_QUE
12:    AFFICHER "Le PIB aura doublé en "
13:    AFFICHER annee
14: FIN_ALGORITHME
```

2 Second degré

Fiche professeur 2A

— Fiche élève correspondante : page 5
— *Fichier AlgoBox associé (algorithme complet) :* `algo_2A.alg`
— *Contexte (1S/1ES/1STL/1STMG) :* Recherche et codage des conditions pour que les valeurs d'un trinôme soient toujours strictement positives.

Fiche élève 2A

On considère la fonction f définie sur $\mathbb{R}$ par $f(x) = ax^2 + bx + c$ avec $a \neq 0$.
Compléter la ligne 11 pour que l'algorithme AlgoBox ci-dessous soit correct :

```
 1: VARIABLES
 2: a EST_DU_TYPE NOMBRE
 3: b EST_DU_TYPE NOMBRE
 4: c EST_DU_TYPE NOMBRE
 5: delta EST_DU_TYPE NOMBRE
 6: DEBUT_ALGORITHME
 7:    LIRE a
 8:    LIRE b
 9:    LIRE c
10:    delta PREND_LA_VALEUR b*b-4*a*c
11:    SI (.....................) ALORS
12:    |  DEBUT_SI
13:    |  AFFICHER "f(x) est toujours strictement positif"
14:    |  FIN_SI
15:    SINON
16:    |  DEBUT_SINON
17:    |  AFFICHER "f(x) n'est pas toujours strictement positif"
18:    |  FIN_SINON
19: FIN_ALGORITHME
```

3 Fonctions

Fiche professeur 3A

— Fiche élève correspondante : page 7
— *Fichier AlgoBox associé (algorithme complet) :* `algo_3A.alg`
— *Contexte (1S/1ES/1STL/1STMG) :* Création d'un tableau de valeurs avec un `TANT_QUE`
— *Prolongement possible :* Faire tracer les points correspondants dans un repère (il suffit d'utiliser l'onglet *Dessiner dans un repère* et d'ajouter l'instruction `TRACER_POINT (x,y)` après `AFFICHER y`).
Attention : pour tracer une courbe en joignant les points par un segment, il faut un algorithme plus complet (il faut les coordonnées des deux points à joindre - un exemple est fourni avec AlgoBox : *menu "Fichier" -> "Ouvrir un exemple"*)

Fiche professeur 3B

— Fiche élève correspondante : page 8
— *Fichier AlgoBox associé (algorithme complet) :* `algo_3B.alg`
— *Contexte (1S) :* Recherche d'une valeur approchée de l'équation $x^2\sqrt{x} = 2$ par dichotomie sur $[0;2]$.
— *Prolongements possibles :*
  — Remplacer la boucle `POUR numero_etape ALLANT_DE 1 A 4` par un `TANT_QUE` portant sur la précision souhaitée.
  — Étudier un autre cas où la fonction est strictement décroissante
  — Établir un algorithme qui fonctionne dans tous les cas (fonction strictement croissante ou strictement décroissante)

Fiche professeur 3C

— Fiche élève correspondante : page 10
— *Fichier AlgoBox associé (algorithme complet) :* `algo_3C.alg`
— *Contexte (1S) :* Recherche par « balayage » du premier point d'une courbe dont le coefficient directeur de la tangente dépasse une certaine valeur .
— *Prolongement possible :* Introduire une variable correspondante au pas et adapter l'algorithme en conséquence.

Fiche élève 3A

Soit f la fonction définie sur $[0;+\infty[$ par $f(x) = x + \sqrt{x}$.
On cherche à créer un algorithme qui permette de compléter automatiquement le tableau de valeurs suivant :

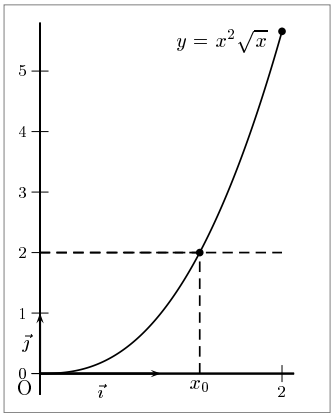
x	0	0,5	1	1,5	2	2,5	3	3,5	4	4,5	5
$y = f(x)$											

Pour cela, on utilise le principe suivant : pour chaque valeur de x, on calcule la valeur correspondante de y et on augmente la valeur de x de 0,5 **tant que** la fin du tableau n'est pas atteinte.
Compléter les lignes 6, 8 et 13 pour que l'algorithme AlgoBox ci-dessous réponde au problème :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   x PREND_LA_VALEUR 0
6:   TANT_QUE (x.....) FAIRE
7:     DEBUT_TANT_QUE
8:     y PREND_LA_VALEUR ............
9:     AFFICHER "Si x vaut "
10:    AFFICHER x
11:    AFFICHER " alors y vaut "
12:    AFFICHER y
13:    x PREND_LA_VALEUR ............
14:    FIN_TANT_QUE
15: FIN_ALGORITHME
```

Fiche élève 3B

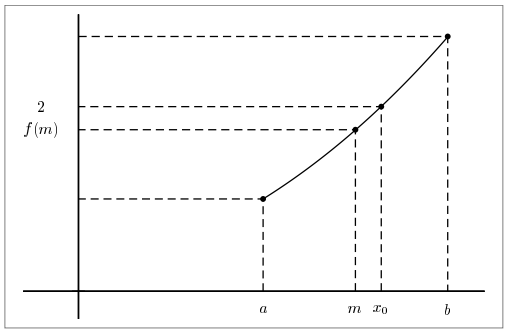
On considère la fonction f définie sur $[0\,;2]$ par $f(x) = x^2\sqrt{x}$ dont la courbe est donnée ci-dessous :



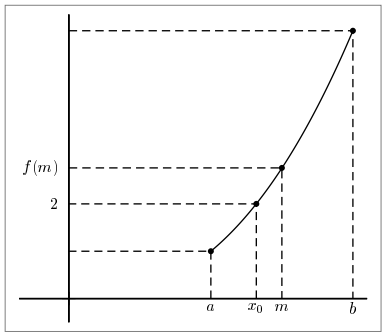
On cherche à déterminer une valeur approchée du réel x_0 tel que $f(x_0) = 2$.
On admet que f est strictement croissante sur $[0\,;2]$ et que sa courbe ne contient pas de « trous » .
Comme $f(0) = 0$ et $f(2) = 4\sqrt{2} > 2$, on sait que $x_0 \in [0\,;2]$.

Pour déterminer une valeur approchée de x_0, on utilise la méthode dite de la « dichotomie » dont le principe consiste à couper l'intervalle en deux et à regarder de quel côté se situe la solution par rapport au milieu de l'intervalle.

1. Étant donné un intervalle $[a;b]$ de milieu m et contenant x_0 (avec $a \geqslant 0$ et $b \leqslant 2$).
   - Si $f(m) < 2$, dans quel intervalle se situe x_0 ? ☐



   - Si $f(m) > 2$, dans quel intervalle se situe x_0 ? ☐



2. Compléter le tableau suivant :

Étape	Intervalle de départ $[a;b]$	milieu m	$f(m) < 2$?	Nouvel intervalle $[a;b]$
1	$a = 0$; $b = 2$	$m = 1$	OUI	$a =$; $b =$
2	$a =$; $b =$	$m =$		$a =$; $b =$
3	$a =$; $b =$	$m =$		$a =$; $b =$
4	$a =$; $b =$	$m =$		$a =$; $b =$

3. On cherche à automatiser les calculs grâce à un algorithme. Compléter les lignes 14 et 18 pour que l'algorithme AlgoBox ci-dessous réponde au problème.

```
 1: VARIABLES
 2: a EST_DU_TYPE NOMBRE
 3: b EST_DU_TYPE NOMBRE
 4: m EST_DU_TYPE NOMBRE
 5: numero_etape EST_DU_TYPE NOMBRE
 6: DEBUT_ALGORITHME
 7:     a PREND_LA_VALEUR 0
 8:     b PREND_LA_VALEUR 2
 9:     POUR numero_etape ALLANT_DE 1 A 4
10:         DEBUT_POUR
11:         m PREND_LA_VALEUR (a+b)/2
12:         SI (m*m*sqrt(m)<2) ALORS
13:             DEBUT_SI
14:             ..... PREND_LA_VALEUR m
15:             FIN_SI
16:         SINON
17:             DEBUT_SINON
18:             ..... PREND_LA_VALEUR m
19:             FIN_SINON
20:         AFFICHER a
21:         AFFICHER " <x0< "
22:         AFFICHER b
23:         FIN_POUR
24: FIN_ALGORITHME
```

Fiche élève 3C

Soit f la fonction définie sur $[0\,;3]$ par $f(x) = 10x^2\sqrt{x}$ et C_f sa courbe représentative dans un repère.

1. Dériver f et montrer que pour $x \in]0\,;3]$, on a $f\,'(x) = 25x\sqrt{x}$.
2. Calculer le coefficient directeur de la tangente à C_f au point d'abscisse 1.
3. Calculer le coefficient directeur de la tangente à C_f au point d'abscisse 2.
4. On cherche à déterminer à l'aide d'un algorithme une valeur approchée à 0,01 près du premier nombre a tel que le coefficient directeur de la tangente au point d'abscisse a soit supérieur ou égal à 50.
   On sait d'après les premières questions que a est compris entre 1 et 2. On part donc de $a = 1$ et on augmente a de 0,01 tant que le coefficient directeur ne dépasse pas 50.
   Compléter les lignes 5 et 7 pour que l'algorithme AlgoBox ci-dessous réponde au problème.

```
1:  VARIABLES
2:  a EST_DU_TYPE NOMBRE
3:  DEBUT_ALGORITHME
4:  |  a PREND_LA_VALEUR 1
5:  |  TANT_QUE (.....................) FAIRE
6:  |  |  DEBUT_TANT_QUE
7:  |  |  a PREND_LA_VALEUR ..............
8:  |  |  FIN_TANT_QUE
9:  |  AFFICHER a
10: FIN_ALGORITHME
```

4 Statistiques/Probabilités

Fiche professeur 4A

— Fiche élève correspondante : page 12
— *Fichier AlgoBox associé (algorithme complet) :* `algo_4A.alg`
— *Contexte (1S/1ES) :* Simulation de 100 000 lancers de deux dés - Calcul du gain moyen associé - Espérance d'une variable aléatoire
— *Prolongement possible :* Au lieu d'une simulation de 100 000 lancers, on peut effectuer des centaines de simulation de 1 000 lancers.

Fiche professeur 4B

— Fiche élève correspondante : page 13
— *Fichier AlgoBox associé (algorithme complet) :* `algo_4B.alg`
— *Contexte (1S/1ES/1STL/1STMG) :* Simulation d'un QCM - Calcul de la moyenne des notes obtenues
— *Prolongement possible :* Effectuer le calcul de la moyenne des notes au sein de l'algorithme.

Fiche professeur 4C

— Fiche élève correspondante : page 14
— *Fichier AlgoBox associé (algorithme complet) :* `algo_4C.alg`
— *Contexte (1S/1ES/1STL/1STMG) :* Simulation de naissances - Loi binomiale

Fiche professeur 4D

— Fiche élève correspondante : page 15
— *Fichier AlgoBox associé (algorithme complet) :* `algo_4D.alg`
— *Contexte (1S/1ES/1STL/1STMG) :* Fluctuation loi binomiale - Détermination avec un algorithme des entiers a et b.
— *Prolongement possible :* Détermination de l'intervalle de fluctuation en divisant a et b par la taille de l'échantillon.

Fiche élève 4A

Un jeu consiste à lancer deux dés, un rouge et un noir. Pour pouvoir jouer il faut payer 1 euro. On gagne 3 euros si la somme des points est supérieure ou égale à 9, 1 euro si la somme des points est inférieure ou égale à 4 et rien dans les autres cas.
On cherche à savoir combien peut-on espérer gagner en moyenne si on joue un grand nombre de fois à ce jeu. Pour cela, on appelle « gain effectif », la différence entre la somme gagnée et la somme misée

Partie A : simulation à l'aide d'un algorithme

On cherche à établir un algorithme simulant 100 000 participations à ce jeu et donnant le gain effectif moyen obtenu.

1. Compléter les lignes 13 et 17 dans l'algorithme AlgoBox ci-dessous pour qu'il réponde au problème :

```
 1: VARIABLES
 2: numero_tirage EST_DU_TYPE NOMBRE
 3: gain_moyen EST_DU_TYPE NOMBRE
 4: resultat EST_DU_TYPE NOMBRE
 5: somme_gains_effectifs EST_DU_TYPE NOMBRE
 6: DEBUT_ALGORITHME
 7:     somme_gains_effectifs PREND_LA_VALEUR 0
 8:     POUR numero_tirage ALLANT_DE 1 A 100000
 9:         DEBUT_POUR
10:         resultat PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,6)+ALGOBOX_ALEA_ENT(1,6)
11:         SI (resultat>=9) ALORS
12:             DEBUT_SI
13:             somme_gains_effectifs PREND_LA_VALEUR somme_gains_effectifs+.........
14:             FIN_SI
15:         SI (resultat>=5 ET resultat<=8) ALORS
16:             DEBUT_SI
17:             somme_gains_effectifs PREND_LA_VALEUR ..............................
18:             FIN_SI
19:         FIN_POUR
20:     gain_moyen PREND_LA_VALEUR somme_gains_effectifs/100000
21:     AFFICHER gain_moyen
22: FIN_ALGORITHME
```

2. Pourquoi l'algorithme ne traite-t-il pas le cas où `resultat` est inférieur ou égal à 4 ?
3. Que peut-on conjecturer sur la valeur du gain effectif moyen en procédant à plusieurs simulations grâce à cet algorithme ?

Partie B : analyse théorique

1. Compléter le tableau suivant en indiquant dans chaque case la somme des points obtenus :

dé rouge/dé noir	1	2	3	4	5	6
1						
2						
3						
4						
5						
6						

2. On note X la variable aléatoire qui donne le gain effectif obtenu par un joueur. À l'aide du tableau ci-dessus, déterminer la loi de probabilité associée à X et calculer son espérance.

Fiche élève 4B

Un QCM comporte cinq questions et, pour chaque question, trois réponses sont proposées dont une seule est exacte. Une bonne réponse rapporte un point, une mauvaise réponse 0.

Question 1 :	☐ Réponse 1	☐ Réponse 2	☐ Réponse 3
Question 2 :	☐ Réponse 1	☐ Réponse 2	☐ Réponse 3
Question 3 :	☐ Réponse 1	☐ Réponse 2	☐ Réponse 3
Question 4 :	☐ Réponse 1	☐ Réponse 2	☐ Réponse 3
Question 5 :	☐ Réponse 1	☐ Réponse 2	☐ Réponse 3

1. Quelles sont les notes minimale et maximale que l'on peut obtenir à ce QCM ?
2. On décide de simuler 1000 fois le fait de répondre au hasard à ce QCM à l'aide de l'algorithme AlgoBox ci-dessous (pour la simulation, on considère que la bonne réponse à chaque question est la première) :

```
 1: VARIABLES
 2: note EST_DU_TYPE NOMBRE
 3: question EST_DU_TYPE NOMBRE
 4: simulation EST_DU_TYPE NOMBRE
 5: effectif EST_DU_TYPE LISTE
 6: reponse_au_hasard EST_DU_TYPE NOMBRE
 7: DEBUT_ALGORITHME
 8:   POUR note ALLANT_DE ... A ...
 9:     DEBUT_POUR
10:     effectif[note] PREND_LA_VALEUR 0
11:     FIN_POUR
12:   POUR simulation ALLANT_DE 1 A 1000
13:     DEBUT_POUR
14:     note PREND_LA_VALEUR 0
15:     POUR question ALLANT_DE 1 A 5
16:       DEBUT_POUR
17:       reponse_au_hasard PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,3)
18:       SI (reponse_au_hasard==1) ALORS
19:         DEBUT_SI
20:         note PREND_LA_VALEUR note+1
21:         FIN_SI
22:       FIN_POUR
23:     effectif[note] PREND_LA_VALEUR effectif[note]+1
24:     FIN_POUR
25:   POUR note ALLANT_DE ... A ...
26:     DEBUT_POUR
27:     AFFICHER "On a obtenu "
28:     AFFICHER effectif[note]
29:     AFFICHER " fois la note "
30:     AFFICHER note
31:     FIN_POUR
32: FIN_ALGORITHME
```

   a) Compléter les lignes 8 et 25 de l'algorithme.
   b) Obtient-on toujours les mêmes résultats en exécutant l'algorithme ?
3. Une exécution de l'algorithme a donné les résultats suivants :

```
***Algorithme lancé***
On a obtenu 126 fois la note 0
On a obtenu 324 fois la note 1
On a obtenu 343 fois la note 2
On a obtenu 166 fois la note 3
On a obtenu 37 fois la note 4
On a obtenu 4 fois la note 5
***Algorithme terminé***
```

   a) Quelle est la moyenne des notes obtenues lors de cette simulation ?
   b) Le calcul de l'écart-type donne $\sigma \approx 1{,}036$. Parmi les 1000 notes obtenues lors de cette simulation, quelle est la proportion de celles comprises dans l'intervalle $[\bar{x} - \sigma\,;\, \bar{x} + \sigma]$?

Fiche élève 4C

On s'intéresse au nombre d'enfants d'une famille. On suppose qu'il n'y a pas de naissances multiples et qu'il y a équiprobabilité pour la naissance d'un garçon ou d'une fille.

Partie A

On suppose que la famille a eu 4 enfants et on note X le nombre de filles.

1. Justifier que X suit une loi binomiale dont on donnera les paramètres.
2. Calculer la probabilité que la famille ait eu au moins une fille.

Partie B

On cherche à déterminer le nombre minimum d'enfants que la famille devrait avoir pour que la probabilité d'avoir au moins une fille soit supérieure à 0.999 .

1. On note n le nombre d'enfants. Déterminer, en fonction de n, la probabilité d'avoir au moins une fille.
2. Montrer que répondre au problème posé revient à déterminer le premier entier n tel que $0.5^n \leqslant 0.001$.
3. Compléter la ligne pour que l'algorithme AlgoBox ci-dessous permette de déterminer cet entier. (rappel : `pow(0.5,n)` permet de calculer 0.5^n)

```
1:  VARIABLES
2:  n EST_DU_TYPE NOMBRE
3:  DEBUT_ALGORITHME
4:  |  n PREND_LA_VALEUR 1
5:  |  TANT_QUE (pow(0.5,n)...........) FAIRE
6:  |  |  DEBUT_TANT_QUE
7:  |  |  n PREND_LA_VALEUR n+1
8:  |  |  FIN_TANT_QUE
9:  |  AFFICHER n
10: FIN_ALGORITHME
```

Fiche élève 4D

On prélève un échantillon dans une population au sein de laquelle on suppose que la proportion d'un certain caractère est p.
Le prélèvement de l'échantillon étant assimilé à un tirage avec remise, le nombre X d'individus de l'échantillon ayant le caractère en question suit alors la loi binomiale de paramètres n et p, où n est égal à la taille de l'échantillon.
Conformément au programme, on note :

— a, le plus petit entier tel que $p(X \leqslant a) > 0{,}025$;
— b, le plus petit entier tel que $p(X \leqslant b) \geqslant 0{,}975$;

On cherche à déterminer les valeurs de a et b à l'aide de l'algorithme AlgoBox ci-dessous qui utilise :

— la commande `ALGOBOX_LOI_BINOMIALE(taille_echantillon,p,k)` permettant de calculer $p(X = k)$;
— une variable somme qui, au fur et à mesure que k est augmenté de 1, contient $p(X \leqslant k)$.

Compléter la ligne 21 de l'algorithme afin que celui-ci permette de déterminer la valeur de b.

```
 1: VARIABLES
 2: taille_echantillon EST_DU_TYPE NOMBRE
 3: p EST_DU_TYPE NOMBRE
 4: somme EST_DU_TYPE NOMBRE
 5: k EST_DU_TYPE NOMBRE
 6: a EST_DU_TYPE NOMBRE
 7: b EST_DU_TYPE NOMBRE
 8: DEBUT_ALGORITHME
 9:    LIRE taille_echantillon
10:    LIRE p
11:    k PREND_LA_VALEUR 0
12:    somme PREND_LA_VALEUR ALGOBOX_LOI_BINOMIALE(taille_echantillon,p,k)
13:    TANT_QUE (somme<=0.025) FAIRE
14:       DEBUT_TANT_QUE
15:       k PREND_LA_VALEUR k+1
16:       somme PREND_LA_VALEUR somme+ALGOBOX_LOI_BINOMIALE(taille_echantillon,p,k)
17:       FIN_TANT_QUE
18:    a PREND_LA_VALEUR k
19:    AFFICHER "a : "
20:    AFFICHER a
21:    TANT_QUE (somme........) FAIRE
22:       DEBUT_TANT_QUE
23:       k PREND_LA_VALEUR k+1
24:       somme PREND_LA_VALEUR somme+ALGOBOX_LOI_BINOMIALE(taille_echantillon,p,k)
25:       FIN_TANT_QUE
26:    b PREND_LA_VALEUR k
27:    AFFICHER "b : "
28:    AFFICHER b
29: FIN_ALGORITHME
```

5 Suites numériques

Remarque préalable :
Les algorithmes proposés dans les sujets du baccalauréat ne semblant pas faire appel aux structures de données du type liste, le choix a été fait aussi de ne pas les utiliser dans ce chapitre.
Il est néanmoins possible d'adapter les algorithmes proposés ici en utilisant des listes AlgoBox dont la syntaxe est très proche du formalisme utilisé en mathématiques pour les suites numériques.

Fiche professeur 5A

- Fiche élève correspondante : page 18
- *Fichier AlgoBox associé (algorithme complet) :* `algo_5A.alg`
- *Contexte (1S/1ES/1STL/1STMG) :* Calcul du terme de rang n d'une suite explicite.
- *Prolongement possible :* Faire afficher tous les termes jusqu'au rang n à l'aide d'une boucle.

Fiche professeur 5B

- Fiche élève correspondante : page 18
- *Fichier AlgoBox associé (algorithme complet) :* `algo_5B.alg`
- *Contexte (1S/1ES/1STL/1STMG) :* Reconnaitre une suite explicite à partir d'un algorithme.

Fiche professeur 5C

- Fiche élève correspondante : page 18
- *Fichier AlgoBox associé (algorithme complet) :* `algo_5C.alg`
- *Contexte (1S/1ES) :* Calcul du terme de rang n d'une suite récurrente.
- *Prolongement possible :* Faire afficher tous les termes jusqu'au rang n.

Fiche professeur 5D

- Fiche élève correspondante : page 19
- *Fichier AlgoBox associé (algorithme complet) :* `algo_5D.alg`
- *Contexte (1S/1ES/1STL/1STMG) :* Reconnaitre une suite récurrente à partir d'un algorithme.

Fiche professeur 5E

— Fiche élève correspondante : page 19
— *Fichier AlgoBox associé (algorithme complet) :* `algo_5E.alg`
— *Contexte (1S/1ES/1STMG) :* Détermination du premier rang n tel que $U_n \geqslant 20$ à l'aide d'une boucle `TANT_QUE`.

Fiche professeur 5F

— Fiche élève correspondante : page 20
— *Fichier AlgoBox associé (algorithme complet) :* `algo_5F.alg`
— *Contexte (1S/1ES/1STMG) :* Détermination du premier rang n tel que $U_n \leqslant 50$ à l'aide d'une boucle `TANT_QUE`.

Fiche professeur 5G

— Fiche élève correspondante : page 21
— *Fichier AlgoBox associé (algorithme complet) :* `algo_5G.alg`
— *Contexte (1S/1ES/1STMG) :* Utilisation d'un algorithme pour comparer l'évolution des termes d'une suite arithmétique avec ceux d'une suite géométrique.
— *Prolongement possible :* Modifier l'algorithme afin de déterminer directement à l'aide d'un `TANT_QUE` l'année à partir de laquelle la proposition 2 devient la meilleure.

Fiche professeur 5H

— Fiche élève correspondante : page 22
— *Fichier AlgoBox associé (algorithme complet) :* `algo_5H.alg`
— *Contexte (1S/1ES/1STL/1STMG) :* Analyser le fonctionnement d'un algorithme - comprendre comment modéliser de façon algorithmique le calcul d'une somme.

Fiche élève 5A

Soit (U_n) la suite définie par $U_n = \dfrac{2n}{n+3}$.

1. Calculer U_0, U_3 et U_{n+1}.
2. Compléter la ligne 6 de l'algorithme AlgoBox ci-dessous pour qu'il permette de calculer U_n après avoir entré n.

```
1: VARIABLES
2: n EST_DU_TYPE NOMBRE
3: U EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   LIRE n
6:   U PREND_LA_VALEUR ..............
7:   AFFICHER U
8: FIN_ALGORITHME
```

Fiche élève 5B

L'algorithme ci-dessous permet de calculer le terme de rang n d'une suite (U_n) définie de façon explicite.

```
1: VARIABLES
2: n EST_DU_TYPE NOMBRE
3: U EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   LIRE n
6:   U PREND_LA_VALEUR n/(n+1)
7:   AFFICHER U
8: FIN_ALGORITHME
```

1. Déterminer U_n en fonction de n.
2. Calculer U_9 et U_{n+1}.

Fiche élève 5C

Soit (U_n) la suite définie par $U_{n+1} = 5 - 3U_n$ et $U_0 = 1$.

1. Calculer U_0, U_1 et U_2.
2. Compléter la ligne 10 de l'algorithme AlgoBox ci-dessous pour qu'il permette de calculer U_n après avoir entré n.

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: U EST_DU_TYPE NOMBRE
 4: i EST_DU_TYPE NOMBRE
 5: DEBUT_ALGORITHME
 6:   U PREND_LA_VALEUR 1
 7:   LIRE n
 8:   POUR i ALLANT_DE 1 A n
 9:     DEBUT_POUR
10:     U PREND_LA_VALEUR ..............
11:     FIN_POUR
12:   AFFICHER U
13: FIN_ALGORITHME
```

Fiche élève 5D

L'algorithme ci-dessous permet de calculer le terme de rang n d'une suite (U_n) définie de façon récurrente. (U_0 étant le premier terme de la suite)

```
1: VARIABLES
2: n EST_DU_TYPE NOMBRE
3: U EST_DU_TYPE NOMBRE
4: i EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:   U PREND_LA_VALEUR 2
7:   LIRE n
8:   POUR i ALLANT_DE 1 A n
9:     DEBUT_POUR
10:    U PREND_LA_VALEUR 4*U-3
11:    FIN_POUR
12:  AFFICHER U
13: FIN_ALGORITHME
```

1. Déduire de l'algorithme la valeur de U_0 et l'expression de U_{n+1} en fonction de U_n.
2. Calculer U_1 et U_2.

Fiche élève 5E

Un produit coûte actuellement 10 euros. Il est prévu que son prix augmente de 8 % par an. On pose $U_0 = 10$ et on note U_n le prix prévu du produit au bout de n années.

1. Comment passe-t-on du prix d'une année au prix de l'année suivante ?
2. La suite (U_n) est-elle arithmétique ou géométrique ?
3. On cherche à déterminer au bout de combien d'années le prix du produit aura doublé à l'aide de l'algorithme ci-dessous :

```
1: VARIABLES
2: n EST_DU_TYPE NOMBRE
3: U EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   n PREND_LA_VALEUR 0
6:   U PREND_LA_VALEUR 10
7:   TANT_QUE (U.........) FAIRE
8:     DEBUT_TANT_QUE
9:     U PREND_LA_VALEUR U*1.08
10:    n PREND_LA_VALEUR n+1
11:    FIN_TANT_QUE
12:  AFFICHER n
13: FIN_ALGORITHME
```

Compléter la ligne 7 pour que l'algorithme réponde à la question.

Fiche élève 5F

La valeur d'une action est actuellement de 100 euros. Il est prévu que cette valeur baisse de 10 % par mois.
On pose $U_0 = 100$ et on note U_n la valeur de l'action au bout de n mois. On cherche à déterminer au bout de combien de mois le cours de l'action aura diminué de moitié à l'aide de l'algorithme ci-dessous :

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: U EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:    n PREND_LA_VALEUR 0
 6:    U PREND_LA_VALEUR 100
 7:    TANT_QUE (U.............) FAIRE
 8:       DEBUT_TANT_QUE
 9:       U PREND_LA_VALEUR U*...........
10:       n PREND_LA_VALEUR n+1
11:       FIN_TANT_QUE
12:    AFFICHER n
13: FIN_ALGORITHME
```

Compléter les lignes 7 et 9 pour que l'algorithme réponde à la question.

Fiche élève 5G

Un salarié a reçu deux propositions de salaire.

1. **Proposition 1 :** La première année, un salaire annuel de 20000 euros puis chaque année une augmentation fixe de 450 euros.
   On pose $U_0 = 20000$, U_1 le salaire au bout d'un an, ..., U_n le salaire au bout de n années.
   a) Calculer U_1 et U_2.
   b) Préciser si (U_n) est arithmétique ou géométrique et exprimer U_n en fonction de n.
2. **Proposition 2 :** La première année, un salaire annuel de 19900 euros puis chaque année une augmentation de 2%.
   On pose $V_0 = 19900$, V_1 le salaire au bout d'un an, ..., V_n le salaire au bout de n années.
   a) Calculer V_1 et V_2.
   b) Préciser si (V_n) est arithmétique ou géométrique et exprimer V_n en fonction de n.
3. Au début la proposition 1 semble meilleure que la proposition 2 pour le salarié. Afin de déterminer si cela reste le cas dans la durée, on cherche à construire un algorithme qui précise la proposition donnant le meilleur salaire annuel pour les 20 prochaines années.
   Compléter les lignes 10, 11 et 15 ci-dessous pour que l'algorithme AlgoBox proposé réponde à la question.

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: U EST_DU_TYPE NOMBRE
 4: V EST_DU_TYPE NOMBRE
 5: DEBUT_ALGORITHME
 6:     U PREND_LA_VALEUR 20000
 7:     V PREND_LA_VALEUR 19900
 8:     POUR n ALLANT_DE 1 A 20
 9:         DEBUT_POUR
10:         U PREND_LA_VALEUR U+.........
11:         V PREND_LA_VALEUR V*.........
12:         AFFICHER "Au bout de la "
13:         AFFICHER n
14:         AFFICHER " ième année, "
15:         SI (.........) ALORS
16:             DEBUT_SI
17:             AFFICHER "la proposition 1 est la meilleure"
18:             FIN_SI
19:         SINON
20:             DEBUT_SINON
21:             AFFICHER "la proposition 2 est la meilleure"
22:             FIN_SINON
23:         FIN_POUR
24: FIN_ALGORITHME
```

Fiche élève 5H

On considère l'algorithme AlgoBox ci-dessous :

```
1:  VARIABLES
2:  somme EST_DU_TYPE NOMBRE
3:  n EST_DU_TYPE NOMBRE
4:  DEBUT_ALGORITHME
5:  | somme PREND_LA_VALEUR 0
6:  | POUR n ALLANT_DE 1 A 100
7:  | | DEBUT_POUR
8:  | | somme PREND_LA_VALEUR somme+sqrt(n)
9:  | | FIN_POUR
10: | AFFICHER somme
11: FIN_ALGORITHME
```

1. Compléter les phrases ci-dessous :
   — Quand n vaut 1 et que la ligne 8 est exécutée, la variable somme a pour nouvelle valeur
   — Quand n vaut 2 et que la ligne 8 est exécutée, la variable somme a pour nouvelle valeur
   — Quand n vaut 3 et que la ligne 8 est exécutée, la variable somme a pour nouvelle valeur
2. Que permet de calculer cet algorithme ?

6 Géométrie

Fiche professeur 6A

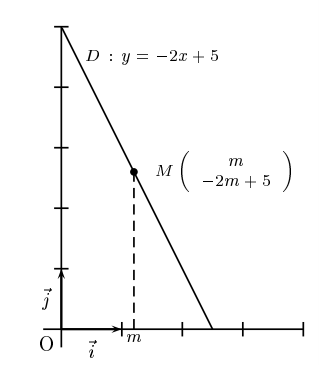
— Fiche élève correspondante : page 24
— *Fichier AlgoBox associé (algorithme complet) :* `algo_6A.alg`
— *Contexte (1S/1STL) :* Algorithme de « balayage » de la distance entre un point fixe et un point variable sur un segment de droite. Le but, d'un point de vue algorithmique, est d'habituer les élèves aux algorithmes de « balayage » (qui figurent déjà au programme à propos des fonctions) et de s'intéresser aussi à leur fiabilité. D'un point de vue mathématique, l'activité propose une initiation à la notion de distance d'un point à une droite.
— *Prolongement possible :* Modifier l'algorithme pour qu'il détermine la valeur minimale de toutes les distances calculées. On peut alors créer un algorithme classique de recherche d'un minimum.

Fiche professeur 6B

— Fiche élève correspondante : page 25
— *Fichier AlgoBox associé (algorithme complet) :* `algo_6B.alg`
— *Contexte (1S) :* Détermination des points à coordonnées entières sur un segment.
— *Prolongement possible :* Proposer un deuxième algorithme basé sur un double balayage en x et en y et comparer le coût en opérations des deux algorithmes.

Fiche élève 6A

Dans un repère orthonormé, on considère la droite D d'équation $y = -2x + 5$ et pour tout réel m compris entre 0 et 2.5, on note M le point de la droite D d'abscisse m.



Un élève prétend que la distance OM ne peut jamais être inférieure à $\sqrt{5}$ et on cherche à vérifier cette affirmation.

Partie A : approche expérimentale

On cherche à établir un algorithme qui vérifie expérimentalement l'affirmation de l'élève. Pour cela, on utilise la méthode dite du « balayage » :
Pour chaque valeur de m (en partant de 0) on calcule la distance OM correspondante et on augmente la valeur de m de 0.01 tant que m n'a pas atteint 2.5.

1. Montrer que, pour tout m, la distance OM est égale à $\sqrt{5m^2 - 20m + 25}$.
2. Compléter les lignes 6, 8 et 13 pour que l'algorithme AlgoBox ci-dessous permette de détecter s'il existe un point M pour lequel la distance OM est inférieure à $\sqrt{5}$.

```
 1: VARIABLES
 2: m EST_DU_TYPE NOMBRE
 3: distance EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:     m PREND_LA_VALEUR 0
 6:     TANT_QUE (m<=.........) FAIRE
 7:         DEBUT_TANT_QUE
 8:         distance PREND_LA_VALEUR ............................
 9:         SI (distance<sqrt(5)) ALORS
10:             DEBUT_SI
11:             AFFICHER "il y a un point pour lequel OM est inférieur à sqrt(5)"
12:             FIN_SI
13:         m PREND_LA_VALEUR ..............
14:         FIN_TANT_QUE
15: FIN_ALGORITHME
```

3. Peut-on être sur que cet algorithme puisse dire de façon totalement fiable s'il n'existe aucun point M tel que $OM < \sqrt{5}$?

Partie B : approche algébrique

1. Montrer que pour tout m compris entre 0 et 2.5, on a $OM = \sqrt{5} \times \sqrt{(m-2)^2 + 1}$.
2. En déduire la valeur minimale que peut prendre la distance OM. Pour quelle valeur de m obtient-on cette distance minimale ?

Partie C : approche géométrique

On note H le point de la droite D d'abscisse 2.

1. Montrer que H est le projeté orthogonal de O sur la droite D.
2. Calculer la distance OH. Que constate-t-on ?

Fiche élève 6B

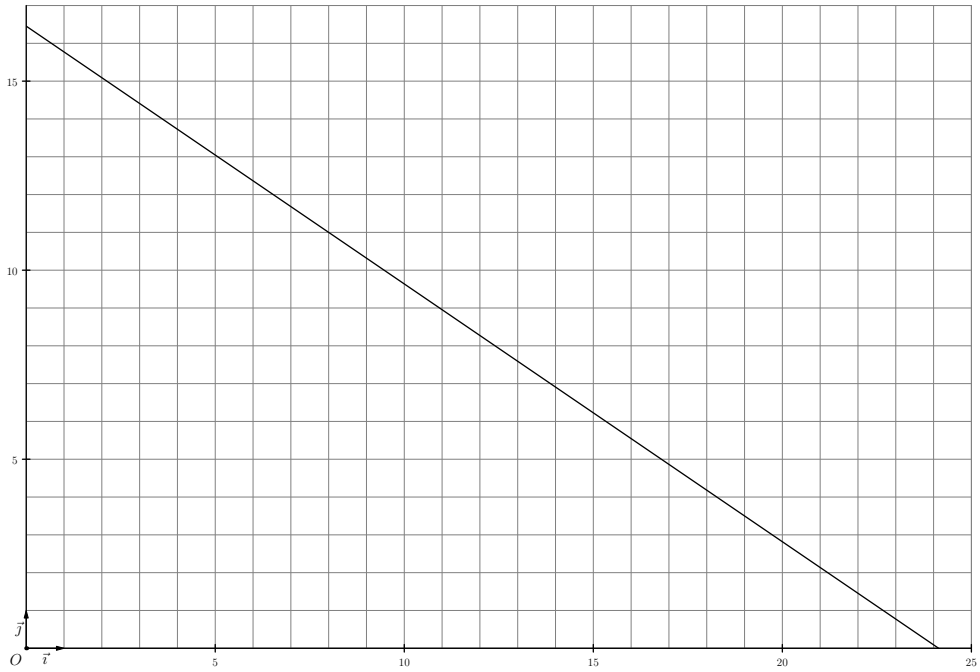
Durant une année un individu a acheté n DVD valant 15 euros pièce et p livres valant 22 euros pièce pour un montant total de 362 euros. On cherche à déterminer les valeurs possibles de n et p.

Première analyse du problème :

1. Quelle relation lie n et p ?
2. Justifier que n est nécessairement inférieur à 25.

Approche graphique :

1. Dans le plan muni d'un repère, on considère les points d'abscisse n et d'ordonnée p pouvant répondre au problème. Justifier que ces points sont situés sur une droite D dont on donnera une équation.
2. La droite D est représentée ci-dessous :



D'après le graphique, il semble que que l'on puisse avoir $n = 5$ et $p = 13$.
Est-ce réellement le cas ?

Approche algorithmique :

Pour déterminer, de façon plus rigoureuse, les valeurs possibles de n et p, on cherche à utiliser un algorithme basé sur le principe suivant : *Pour toutes les abscisses entières x possibles sur la droite D, on calcule l'ordonnée y du point de la droite d'abscisse x et, si y est un entier, on affiche x.*

Compléter les lignes 5, 7 et 8 de l'algorithme AlgoBox ci-contre pour qu'il réponde au problème. *(Note technique : pour déterminer si un nombre est un entier, on peut utiliser la fonction partie entière dont la syntaxe est `floor()` dans AlgoBox.)*

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:     POUR x ALLANT_DE ... A ....
6:         DEBUT_POUR
7:         y PREND_LA_VALEUR ..........................
8:         SI (..................) ALORS
9:             DEBUT_SI
10:            AFFICHER x
11:            FIN_SI
12:        FIN_POUR
13: FIN_ALGORITHME
```

7 Trigonométrie

Fiche professeur 7A

— Fiche élève correspondante : page 27
— *Fichier AlgoBox associé (algorithme complet) :* `algo_7A.alg`
— *Contexte (1S/1STL) :* Détermination de la mesure principale d'une mesure de la forme $\frac{n\pi}{d}$ avec $n > 0$ et $d > 0$ - Utilisation d'un TANT QUE.
— *Prolongement possible :* Déterminer le même genre d'algorithme avec $n < 0$ et $d > 0$ ou concevoir un algorithme qui fonctionne dans tous les cas. On peut aussi créer un algorithme qui permette de trouver le résultat directement sans boucle (plus difficile).

Fiche élève 7A

On cherche à déterminer la mesure principale d'un angle dont une mesure est de la forme $\frac{n\pi}{d}$ (avec $n > 0$ et $d > 0$) en enlevant 2π tant que cela est nécessaire.

1. Compléter le calcul suivant : $\frac{n\pi}{d} - 2\pi = \frac{(\qquad\qquad)\pi}{d}$.
2. Compléter les lignes 14 et 16 pour que l'algorithme AlgoBox ci-dessous réponde à la question.

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: d EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   LIRE n
 6:   LIRE d
 7:   SI (n>0 ET d>0 ) ALORS
 8:     DEBUT_SI
 9:     AFFICHER "mesure principale de "
10:     AFFICHER n
11:     AFFICHER "*pi/"
12:     AFFICHER d
13:     AFFICHER " : "
14:     TANT_QUE (.............) FAIRE
15:       DEBUT_TANT_QUE
16:       n PREND_LA_VALEUR ......................
17:       FIN_TANT_QUE
18:     AFFICHER n
19:     AFFICHER "*pi/"
20:     AFFICHER d
21:     FIN_SI
22: FIN_ALGORITHME
```

Deuxième partie

Annexes

A

Structures algorithmiques de base avec AlgoBox

Les activités proposées dans cette annexe permettent de s'initier aux structures algorithmiques de base prévues au programme quelque soit la filière (aucun pré-requis mathématique spécifique n'est nécessaire).

A.1 Variables et affectations

Les variables en algorithmique

— Les variables algorithmiques peuvent servir à stocker des données de différents types, mais nous nous contenterons ici d'utiliser des variables du type `NOMBRE`.
— La valeur d'une variable peut changer au fil des instructions de l'algorithme.
— Les opérations sur les variables s'effectuent ligne après ligne et les unes après les autres.
— Quand l'ordinateur exécute une ligne du type `mavariable PREND_LA_VALEUR un calcul`, il effectue d'abord le calcul et stocke ensuite le résultat dans `mavariable`.

► Activité n°1
On considère l'algorithme suivant :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: z EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:   x PREND_LA_VALEUR 2
7:   y PREND_LA_VALEUR 3
8:   z PREND_LA_VALEUR x+y
9: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable x contient la valeur : ☐
— La variable y contient la valeur : ☐
— La variable z contient la valeur : ☐

► Activité n°2
On considère l'algorithme suivant :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
4:   x PREND_LA_VALEUR 2
5:   x PREND_LA_VALEUR x+1
6: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable x contient la valeur : ☐

► Activité n°3
Ajoutons la ligne « `x PREND_LA_VALEUR 4*x` » à la fin du code précédent. Ce qui donne :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
4:   x PREND_LA_VALEUR 2
5:   x PREND_LA_VALEUR x+1
6:   x PREND_LA_VALEUR 4*x
7: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable x contient la valeur : ☐

► Activité n°4
On considère l'algorithme suivant :

```
1: VARIABLES
2: A EST_DU_TYPE NOMBRE
3: B EST_DU_TYPE NOMBRE
4: C EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:    A PREND_LA_VALEUR 5
7:    B PREND_LA_VALEUR 3
8:    C PREND_LA_VALEUR A+B
9:    B PREND_LA_VALEUR B+A
10:   A PREND_LA_VALEUR C
11: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable A contient la valeur : ☐

— La variable B contient la valeur : ☐

— La variable C contient la valeur : ☐

► Activité n°5
On considère l'algorithme suivant :

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: z EST_DU_TYPE NOMBRE
5: DEBUT_ALGORITHME
6:    LIRE x
7:    y PREND_LA_VALEUR x-2
8:    z PREND_LA_VALEUR -3*y-4
9:    AFFICHER z
10: FIN_ALGORITHME
```

On cherche maintenant à obtenir un algorithme équivalent sans utiliser la variable y. Compléter la ligne 6 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: z EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:    LIRE x
6:    z PREND_LA_VALEUR .....................
7:    AFFICHER z
8: FIN_ALGORITHME
```

A.2 Instructions conditionnelles

SI...ALORS...SINON

Comme nous l'avons vu ci-dessus, un algorithme permet d'exécuter une liste d'instructions les unes à la suite des autres. Mais on peut aussi "dire" à un algorithme de n'exécuter des instructions que si une certaine condition est remplie. Cela se fait grâce à la commande `SI...ALORS` :

```
SI...ALORS
  DEBUT_SI
  ...
  FIN_SI
```

Il est aussi possible d'indiquer en plus à l'algorithme de traiter le cas où la condition n'est pas vérifiée. On obtient alors la structure suivante :

```
SI...ALORS
  DEBUT_SI
  ...
  FIN_SI
  SINON
    DEBUT_SINON
    ...
    FIN_SINON
```

► Activité n°6

On cherche à créer un algorithme qui demande un nombre à l'utilisateur et qui affiche la racine carrée de ce nombre s'il est positif. Compléter la ligne 6 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
 1: VARIABLES
 2: x EST_DU_TYPE NOMBRE
 3: racine EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   LIRE x
 6:   SI (.........) ALORS
 7:     DEBUT_SI
 8:     racine PREND_LA_VALEUR sqrt(x)
 9:     AFFICHER racine
10:     FIN_SI
11: FIN_ALGORITHME
```

► Activité n°7

On cherche à créer un algorithme qui demande à l'utilisateur d'entrer deux nombres (stockés dans les variables `x` et `y`) et qui affiche le plus grand des deux. Compléter les ligne 9 et 13 dans l'algorithme ci-dessous pour qu'il réponde au problème.

```
 1: VARIABLES
 2: x EST_DU_TYPE NOMBRE
 3: y EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   LIRE x
 6:   LIRE y
 7:   SI (x>y) ALORS
 8:     DEBUT_SI
 9:     AFFICHER .......
10:     FIN_SI
11:   SINON
12:     DEBUT_SINON
13:     AFFICHER ......
14:     FIN_SINON
15: FIN_ALGORITHME
```

► Activité n°8
On considère l'algorithme suivant :

```
1: VARIABLES
2: A EST_DU_TYPE NOMBRE
3: B EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   A PREND_LA_VALEUR 1
6:   B PREND_LA_VALEUR 3
7:   SI (A>0) ALORS
8:     DEBUT_SI
9:     A PREND_LA_VALEUR A+1
10:    FIN_SI
11:  SI (B>4) ALORS
12:    DEBUT_SI
13:    B PREND_LA_VALEUR B-1
14:    FIN_SI
15: FIN_ALGORITHME
```

Après exécution de l'algorithme :

— La variable A contient la valeur : ☐

— La variable B contient la valeur : ☐

► Activité n°9
On cherche à concevoir un algorithme correspondant au problème suivant :

— on demande à l'utilisateur d'entrer un nombre (qui sera représenté par la variable x)

— si le nombre entré est différent de 1, l'algorithme doit stocker dans une variable y la valeur de 1/(x-1) et afficher la valeur de y (note : la condition x différent de 1 s'exprime avec le code x!=1). On ne demande pas de traiter le cas contraire.

Compléter l'algorithme ci-dessous pour qu'il réponde au problème.

```
1: VARIABLES
2: x EST_DU_TYPE NOMBRE
3: y EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   LIRE ......
6:   SI (........) ALORS
7:     DEBUT_SI
8:     ....... PREND_LA_VALEUR .......
9:     AFFICHER .......
10:    FIN_SI
11: FIN_ALGORITHME
```

A.3 Boucles

Boucles POUR...DE...A

— Les boucles permettent de répéter des instructions autant de fois que l'on souhaite.
— Lorsqu'on connait par avance le nombre de fois que l'on veut répéter les instructions, on utilise une boucle du type `POUR...DE...A` dont la structure est la suivante :

```
POUR...ALLANT_DE...A...
  DEBUT_POUR
  ...
  FIN_POUR
```

— Exemple : l'algorithme ci-dessous permet d'afficher la racine carrée de tous les entiers de 1 jusqu'à 50.

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: racine EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   POUR n ALLANT_DE 1 A 50
 6:     DEBUT_POUR
 7:     racine PREND_LA_VALEUR sqrt(n)
 8:     AFFICHER racine
 9:     FIN_POUR
10: FIN_ALGORITHME
```

La variable `n` est appelée « compteur de la boucle ».
— Remarques :
  — La variable servant de compteur pour la boucle doit être du type NOMBRE et doit être déclarée préalablement (comme toutes les variables).
  — Dans AlgoBox, cette variable est automatiquement augmentée de 1 à chaque fois.
  — On peut utiliser la valeur du compteur pour faire des calculs à l'intérieur de la boucle, mais les instructions comprises entre `DEBUT_POUR` et `FIN_POUR` ne doivent en aucun cas modifier la valeur de la variable qui sert de compteur.

► Activité n°10
On cherche à concevoir un algorithme qui affiche, grâce à une boucle POUR...DE...A, les résultats des calculs suivants : 8*1 ; 8*2 ; 8*3 ; 8*4 ; ... jusqu'à 8*10 .
La variable `n` sert de compteur à la boucle et la variable `produit` sert à stocker et afficher les résultats. Compléter les lignes 5 et 7 dans l'algorithme ci-dessous pour qu'il réponde au problème :

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: produit EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   POUR n ALLANT_DE .... A .....
 6:     DEBUT_POUR
 7:     produit PREND_LA_VALEUR .......
 8:     AFFICHER produit
 9:     FIN_POUR
10: FIN_ALGORITHME
```

► Activité n°11
On considère l'algorithme suivant :

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: somme EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:   somme PREND_LA_VALEUR 0
 6:   POUR n ALLANT_DE 1 A 100
 7:     DEBUT_POUR
 8:     somme PREND_LA_VALEUR somme+n
 9:     FIN_POUR
10:   AFFICHER somme
11: FIN_ALGORITHME
```

Compléter les phrases suivantes :

— Après exécution de la ligne 5, La variable `somme` contient la valeur :

— Lorsque le compteur `n` de la boucle vaut 1 et après exécution du calcul ligne 8, la variable `somme` vaut :

— Lorsque le compteur `n` de la boucle vaut 2 et après exécution du calcul ligne 8, la variable `somme` vaut :

— Lorsque le compteur `n` de la boucle vaut 3 et après exécution du calcul ligne 8, la variable `somme` vaut :

Que permet de calculer cet algorithme ?

► Activité n°12

Compléter les lignes 6 et 8 de l'algorithme ci-dessous pour qu'il permette de calculer la somme $5^2 + 6^2 + 7^2 + \cdots + 24^2 + 25^2$.

```
 1: VARIABLES
 2: n EST_DU_TYPE NOMBRE
 3: somme EST_DU_TYPE NOMBRE
 4: DEBUT_ALGORITHME
 5:     somme PREND_LA_VALEUR 0
 6:     POUR n ALLANT_DE .... A ......
 7:         DEBUT_POUR
 8:         somme PREND_LA_VALEUR somme+......
 9:         FIN_POUR
10:     AFFICHER somme
11: FIN_ALGORITHME
```

Boucles TANT QUE...

— Il n'est pas toujours possible de connaitre par avance le nombre de répétitions nécessaires à un calcul. Dans ce cas là, il est possible d'avoir recours à la structure TANT QUE... qui se présente de la façon suivante :

```
TANT_QUE...FAIRE
  DEBUT_TANT_QUE
  ...
  FIN_TANT_QUE
```

Cette structure de boucle permet de répéter une série d'instructions (comprises entre DEBUT_TANT_QUE et FIN_TANT_QUE) tant qu'*une certaine condition est vérifiée*.

— Exemple : Comment savoir ce qu'il reste si on enlève 25 autant de fois que l'on peut au nombre 583 ? Pour cela on utilise une variable n, qui contient 583 au début, à laquelle on enlève 25 tant que c'est possible, c'est à dire tant que n est supérieur ou égal à 25.

```
1: VARIABLES
2: n EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
4:     n PREND_LA_VALEUR 583
5:     TANT_QUE (n>=25) FAIRE
6:         DEBUT_TANT_QUE
7:         n PREND_LA_VALEUR n-25
8:         FIN_TANT_QUE
9:     AFFICHER n
10: FIN_ALGORITHME
```

— Remarques :
  — Si la condition du TANT QUE... est fausse dès le début, les instructions entre DEBUT_TANT_QUE et FIN_TANT_QUE ne sont jamais exécutées (la structure TANT QUE ne sert alors strictement à rien).
  — Il est indispensable de s'assurer que la condition du TANT QUE... finisse par être vérifiée (le code entre DEBUT_TANT_QUE et FIN_TANT_QUE doit rendre vraie la condition tôt ou tard), sans quoi l'algorithme ne pourra pas fonctionner.

► Activité n°13

Un individu a emprunté à un ami une somme de 2500 euros (prêt sans intérêts). Pour rembourser son ami, il prévoit de lui remettre 110 euros par mois. Mais comme cela ne correspond pas à un nombre pile de mois, il se demande quel sera le montant à rembourser le dernier mois.
Compléter la ligne 5 dans l'algorithme ci-dessous pour qu'il réponde au problème :

```
1: VARIABLES
2: montant EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
4:     montant PREND_LA_VALEUR 2500
5:     TANT_QUE (.............) FAIRE
6:         DEBUT_TANT_QUE
7:         montant PREND_LA_VALEUR montant-110
8:         FIN_TANT_QUE
9:     AFFICHER montant
10: FIN_ALGORITHME
```

► Activité n°14

On cherche à connaître le plus petit entier N tel que 2^N soit supérieur ou égal à 10000. Pour résoudre ce problème de façon algorithmique :
— On utilise une variable N à laquelle on donne au début la valeur 1.
— On augmente de 1 la valeur de N tant que 2^N n'est pas supérieur ou égal à 10000.
Une structure TANT QUE est particulièrement adaptée à ce genre de problème car on ne sait pas a priori combien de calculs seront nécessaires.
Compléter les lignes 5 et 7 de l'algorithme ci-dessous pour qu'il réponde au problème :
(remarque : `pow(2,N)` est le code AlgoBox pour calculer 2^N)

```
1: VARIABLES
2: N EST_DU_TYPE NOMBRE
3: DEBUT_ALGORITHME
4:   N PREND_LA_VALEUR 1
5:   TANT_QUE (pow(2,N).......) FAIRE
6:     DEBUT_TANT_QUE
7:     N PREND_LA_VALEUR .......
8:     FIN_TANT_QUE
9:   AFFICHER N
10: FIN_ALGORITHME
```

► Activité n°15

On considère le problème suivant :

- On lance une balle d'une hauteur initiale de 300 cm.
- On suppose qu'à chaque rebond, la balle perd 10% de sa hauteur (la hauteur est donc multipliée par 0.9 à chaque rebond).
- On cherche à savoir le nombre de rebonds nécessaire pour que la hauteur de la balle soit inférieure ou égale à 10 cm.

Compléter les lignes 7 et 10 de l'algorithme ci-dessous pour qu'il réponde au problème.

```
1: VARIABLES
2: nombre_rebonds EST_DU_TYPE NOMBRE
3: hauteur EST_DU_TYPE NOMBRE
4: DEBUT_ALGORITHME
5:   nombre_rebonds PREND_LA_VALEUR 0
6:   hauteur PREND_LA_VALEUR 300
7:   TANT_QUE (hauteur......) FAIRE
8:     DEBUT_TANT_QUE
9:     nombre_rebonds PREND_LA_VALEUR nombre_rebonds+1
10:    hauteur PREND_LA_VALEUR ...........
11:    FIN_TANT_QUE
12:  AFFICHER nombre_rebonds
13: FIN_ALGORITHME
```

Mémento sur l'utilisation d'AlgoBox

B.1 Équivalence entre « pseudo-codes »

Résumé des différences entre le pseudo-code utilisé par AlgoBox et celui que l'on peut rencontrer dans les manuels.

B.1.1 Entrée des données

Pseudo-code AlgoBox	Variantes
`LIRE...`	`Saisir...` `Entrer...`

B.1.2 Affichage des données

Pseudo-code AlgoBox	Variantes
`AFFICHER...`	`Écrire...`

B.1.3 Affecter une valeur à une variable

Pseudo-code AlgoBox	Variantes

`A PREND_LA_VALEUR 3`	`Affecter 3 à A`
	`A:=3`
	`A←3`

B.1.4 Structure SI ALORS

Pseudo-code AlgoBox	Variantes

SI...ALORS
  DEBUT_SI
  ...
  FIN_SI
	Si...alors
	  ...
	FinSi
	
	Si...
	Alors...
	FinSi

Pseudo-code AlgoBox	Variantes

SI...ALORS
  DEBUT_SI
  ...
  FIN_SI
  SINON
  DEBUT_SINON
  ...
  FIN_SINON
	Si...alors
	  ...
	  sinon
	  ...
	FinSi
	
	Si...
	Alors...
	Sinon...
	FinSi

B.1.5 Boucle POUR...

Pseudo-code AlgoBox	Variantes

POUR...ALLANT_DE...A...
  DEBUT_POUR
  ...
  FIN_POUR
	Pour...de...jusqu'à...
	  ...
	FinPour
	
	Pour...variant de...à...
	  ...
	FinPour

B.1.6 Structure TANT QUE...

Pseudo-code AlgoBox	Variantes

TANT_QUE...FAIRE
  DEBUT_TANT_QUE
  ...
  FIN_TANT_QUE
	Tant que...
	  ...
	FinTantQue

B.2 Les problèmes de syntaxe

B.2.1 Les erreurs de syntaxe les plus courantes

— *Erreur classique n°1 :* utiliser la virgule au lieu du point comme séparateur décimal
— *Erreur classique n°2 :* utiliser la syntaxe "`SI x=y`" au lieu de "`SI x==y`" pour vérifier une égalité dans une condition
— *Erreur classique n°3 :* utiliser la syntaxe `x^y` au lieu de `pow(x,y)` pour les puissances.

B.2.2 Syntaxe des opérations mathématiques

Opération mathématique	syntaxe AlgoBox
$\sqrt{x}$	`sqrt(x)`
x^y	`pow(x,y)`
$\cos(x)$ (x en radians)	`cos(x)`
$\sin(x)$ (x en radians)	`sin(x)`
$\tan(x)$ (x en radians)	`tan(x)`
e^x	`exp(x)`
$\ln x$	`log(x)`
$\|x\|$	`abs(x)`
Partie entière de x	`floor(x)`
Nombre pseudo-aléatoire compris entre 0 et 1	`random()`
Reste de la division euclidienne de n par p	`n%p`
π	`Math.PI`
$\arccos(x)$	`acos(x)`
$\arcsin(x)$	`asin(x)`
$\arctan(x)$	`atan(x)`
Valeur approchée de x à 10^{-n} près	`ALGOBOX_ARRONDIR(x,n)`

B.2.3 Syntaxe pour les conditions

Condition logique	syntaxe AlgoBox
$x = y$	`x==y`
$x \neq y$	`x!=y`
$x < y$	`x<y`
$x \leqslant y$	`x<=y`
$x > y$	`x>y`
$x \geqslant y$	`x>=y`
condition1 **ou** condition2	`condition1 OU condition2`
condition1 **et** condition2	`condition1 ET condition2`

Remarque : il est possible d'inclure des opérations mathématiques dans les conditions. (exemple : `pow(2,n)<100`)

B.2.4 Syntaxe pour les fonctions statistiques et les opérations sur les listes

Somme des termes d'une liste	`ALGOBOX_SOMME(nom_liste,rang_premier_terme,rang_dernier_terme)`
Moyenne	`ALGOBOX_MOYENNE(nom_liste,rang_premier_terme,rang_dernier_terme)`
Variance	`ALGOBOX_VARIANCE(nom_liste,rang_premier_terme,rang_dernier_terme)`
Écart-type	`ALGOBOX_ECART_TYPE(nom_liste,rang_premier_terme,rang_dernier_terme)`
Médiane	`ALGOBOX_MEDIANE(nom_liste,rang_premier_terme,rang_dernier_terme)`
Q1 (version calculatrice)	`ALGOBOX_QUARTILE1(nom_liste,rang_premier_terme,rang_dernier_terme)`
Q3 (version calculatrice)	`ALGOBOX_QUARTILE3(nom_liste,rang_premier_terme,rang_dernier_terme)`
Q1 (version « officielle »)	`ALGOBOX_QUARTILE1_BIS(nom_liste,rang_premier_terme,rang_dernier_terme)`
Q3 (version « officielle »)	`ALGOBOX_QUARTILE3_BIS(nom_liste,rang_premier_terme,rang_dernier_terme)`
Minimum d'une liste	`ALGOBOX_MINIMUM(nom_liste,rang_premier_terme,rang_dernier_terme)`
Maximum d'une liste	`ALGOBOX_MAXIMUM(nom_liste,rang_premier_terme,rang_dernier_terme)`
Rang du minimum	`ALGOBOX_POS_MINIMUM(nom_liste,rang_premier_terme,rang_dernier_terme)`
Rang du maximum	`ALGOBOX_POS_MAXIMUM(nom_liste,rang_premier_terme,rang_dernier_terme)`

B.2.5 Fonctions concernant les probabilités

entier pseudo-aléatoire entre p et n	`ALGOBOX_ALEA_ENT(p,n)`
$\binom{n}{p}$ $(n \leqslant 100)$	`ALGOBOX_COEFF_BINOMIAL(n,p)`
$p(X = k)$ pour la loi binomiale $(n \leqslant 100)$	`ALGOBOX_LOI_BINOMIALE(n,p,k)`
$p(X < x)$ pour la loi normale centrée réduite	`ALGOBOX_LOI_NORMALE_CR(x)`
$p(X < x)$ pour la loi normale	`ALGOBOX_LOI_NORMALE(esp,ecart,x)`
x tel $p(X < x) = p$ pour la loi normale centrée réduite	`ALGOBOX_INVERSE_LOI_NORMALE_CR(p)`
x tel $p(X < x) = p$ pour la loi normale	`ALGOBOX_INVERSE_LOI_NORMALE(esp,ecart,p)`
$n!$ $(n \leqslant 69)$	`ALGOBOX_FACTORIELLE(n)`

B.2.6 Fonctions concernant les chaines de caractères

Remarque : le contenu d'une chaîne doit être encadré par des guillemets.
(exemple : `machaine prend la valeur "bonjour"`)

concaténation de deux chaines A et B	`A+B`
sous-chaine de n caractères à partir de la position p	`machaine.substr(p,n)`
transformation d'un nombre n en chaine	`n.toString()`
transformation d'une chaine en entier	`parseInt(machaine)`
transformation d'une chaine en réel	`parseFloat(machaine)`
longueur d'une chaine	`machaine.length`
code ASCII du caractère situé à la position p	`machaine.charCodeAt(p)`
chaine de code ASCII p	`String.fromCharCode(p)`

B.3 Fonctionnement d'AlgoBox

B.3.1 Les deux règles fondamentales

1. Toute variable doit d'abord être déclarée avant de pouvoir être utilisée. La première chose à faire avant de concevoir un algorithme est d'ailleurs de lister toutes les variables qui seront nécessaires.
2. Une nouvelle instruction ne peut s'insérer que sur une ligne vierge.

B.3.2 Les variables

Attention

Le nom des variables ne doit pas contenir d'espaces (que l'on peut remplacer par le caractère _), ni d'accents, ni de caractères spéciaux. On ne peut pas non plus utiliser certains termes réservés : par exemple, une variable ne peut pas être appelée `NOMBRE`. Il est par contre conseillé d'utiliser des noms de variables explicites (mêmes longs) pour rendre les algorithmes plus clairs.

B.3.3 Les listes de nombres

Il est possible d'entrer directement les termes d'une liste. Pour cela, il suffit d'utiliser l'instruction `LIRE maliste[1]` (1 représentant le premier rang des termes de la liste que l'on veut entrer). Lors de l'exécution de l'algorithme, il suffira alors d'entrer toutes les valeurs souhaitées (dans l'ordre) en les séparant par : .

B.3.4 Boucle POUR...DE...A

— On n'utilise ce genre de boucles que si on connaît à l'avance le nombre de répétitions à effectuer.
— La variable servant de compteur pour la boucle doit être du type NOMBRE et doit être déclarée préalablement (comme toutes les variables).
— Dans AlgoBox, cette variable est automatiquement augmentée de 1 à chaque fois.

Attention

— On peut utiliser la valeur du compteur pour faire des calculs à l'intérieur de la boucle, mais les instructions comprises entre `DEBUT_POUR` et `FIN_POUR` ne doivent en aucun cas modifier la valeur de la variable qui sert de compteur.
— Le nombre d'itérations sur AlgoBox est limité à 500 000. En cas de dépassement, l'algorithme s'arrête et le message « ***Algorithme interrompu suite à une erreur*** » est affiché.
— Si les instructions à répéter comportent l'affichage d'une variable ou un tracé graphique, il faut limiter le nombre d'itérations à moins d'un millier (ou guère plus) : sans quoi, l'exécution du programme risque de prendre beaucoup trop de temps. Par contre, s'il n'y que des calculs à répéter on peut pousser le nombre d'itérations plus loin.

B.3.5 Structure TANT QUE

— Cette structure sert à répéter des instructions que l'on connaisse ou non par avance le nombre d'itérations à effectuer.
— Si la condition du TANT QUE... est fausse dès le début, les instructions entre `DEBUT_TANT_QUE` et `FIN_TANT_QUE` ne sont jamais exécutées (la structure TANT QUE ne sert alors strictement à rien).

Attention

— Il est indispensable de s'assurer que la condition du TANT QUE... finisse par être vérifiée (le code entre `DEBUT_TANT_QUE` et `FIN_TANT_QUE` doit rendre vraie la condition tôt ou tard), sans quoi l'algorithme va se lancer dans une « boucle infinie ».
— Afin d'éviter les boucles infinies, le nombre d'itérations est là aussi limité à 500 000. En cas de dépassement, l'algorithme s'arrête et le message « ***Algorithme interrompu suite à une erreur*** » est affiché.
— Si les instructions à répéter comportent l'affichage d'une variable ou un tracé graphique, il faut limiter le nombre d'itérations à moins d'un millier (ou guère plus) : sans quoi, l'exécution du programme risque de prendre beaucoup trop de temps. Par contre, s'il n'y que des calculs à répéter on peut pousser le nombre d'itérations plus loin.

B.3.6 Utilisation de l'onglet « Utiliser une fonction numérique »

En activant l'option *"Utiliser une fonction"* dans l'onglet *"Utiliser une fonction numérique"*, on peut utiliser l'image de n'importe quel nombre (ou variable de type nombre) par la fonction notée `F1` dans le code de l'algorithme. Il suffit pour cela d'entrer l'expression de `F1(x)` en fonction de `x` dans le champ prévu pour cela. Pour utiliser l'image d'un nombre `nb` par la fonction `F1` dans l'algorithme, il suffit d'utiliser le code : `F1(nb)` (cela peut se faire dans une affectation ou dans une expression conditionnelle). Cette option est particulièrement utile quand un algorithme nécessite le calcul de plusieurs images par une même fonction.
Un exemple classique est celui de la dichotomie :

```
1: VARIABLES
2: precision EST_DU_TYPE NOMBRE
3: a EST_DU_TYPE NOMBRE
4: b EST_DU_TYPE NOMBRE
5: m EST_DU_TYPE NOMBRE
6: DEBUT_ALGORITHME
7:    a PREND_LA_VALEUR ...
8:    b PREND_LA_VALEUR ...
9:    LIRE precision
10:   TANT_QUE (b-a>precision) FAIRE
11:      DEBUT_TANT_QUE
12:      m PREND_LA_VALEUR (a+b)/2
13:      SI (F1(m)*F1(b)>0) ALORS
14:         DEBUT_SI
15:         b PREND_LA_VALEUR m
16:         FIN_SI
17:      SINON
18:         DEBUT_SINON
19:         a PREND_LA_VALEUR m
20:         FIN_SINON
21:      AFFICHER a
22:      AFFICHER " < solution < "
23:      AFFICHER b
24:      FIN_TANT_QUE
25: FIN_ALGORITHME
```

B.3.7 Utilisation de l'onglet « Dessiner dans un repère »

En activant l'option *"Utiliser un repère"* dans l'onglet *"Dessiner dans un repère"*, un repère graphique est automatiquement ajouté dans la fenêtre de test de l'algorithme. Il est alors possible d'inclure dans le code de l'algorithme des instructions pour tracer des points et des segments dans ce repère en utilisant les boutons *"Ajouter TRACER POINT"* et *"Ajouter TRACER SEGMENT"*.

Attention

La « fenêtre » du repère est définie lors de la première utilisation des instructions `TRACER_POINT` et `TRACER_SEGMENT`. Si la fenêtre doit dépendre de la valeur de certaines variables, il faut s'assurer que celles-ci sont bien définies avant le premier tracé.

Exemple : représentation graphique d'une fluctuation d'échantillonnage (100 simulations de 1000 lancers d'une pièce)

```
VARIABLES
simulation EST_DU_TYPE NOMBRE
lancer EST_DU_TYPE NOMBRE
nb_de_piles EST_DU_TYPE NOMBRE
frequence EST_DU_TYPE LISTE
moy EST_DU_TYPE NOMBRE
ecart_type EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
| POUR simulation ALLANT_DE 1 A 100
| | DEBUT_POUR
| | nb_de_piles PREND_LA_VALEUR 0
| | POUR lancer ALLANT_DE 1 A 1000
| | | DEBUT_POUR
| | | SI (random()<0.5) ALORS
| | | | DEBUT_SI
| | | | nb_de_piles PREND_LA_VALEUR nb_de_piles+1
| | | | FIN_SI
| | | FIN_POUR
| | frequence[simulation] PREND_LA_VALEUR nb_de_piles/1000
| | TRACER_POINT (simulation,frequence[simulation])
| | FIN_POUR
| moy PREND_LA_VALEUR ALGOBOX_MOYENNE(frequence,1,100)
| TRACER_SEGMENT (1,moy)->(100,moy)
| ecart_type PREND_LA_VALEUR ALGOBOX_ECART_TYPE(frequence,1,100)
| TRACER_SEGMENT (1,moy+2*ecart_type)->(100,moy+2*ecart_type)
| TRACER_SEGMENT (1,moy-2*ecart_type)->(100,moy-2*ecart_type)
FIN_ALGORITHME
```

```
Xmin: 1 ; Xmax: 100 ; Ymin: 0.4 ; Ymax: 0.6 ; GradX: 10 ; GradY: 0.01
```

B.3.8 Utilisation d'une « Fonction locale »

Il est possible avec AlgoBox de définir et utiliser des fonctions locales que l'on peut appeler dans l'algorithme ensuite. Pour créer une nouvelle fonction locale, il suffit de se placer sur FONCTIONS_UTILISEES, de cliquer sur « Nouvelle ligne » puis sur « Déclarer nouvelle fonction » dans l'onglet « Fonctions locales ». Pour déclarer une éventuelle variable locale nécessaire à une fonction, il faut se placer sur la ligne « VARIABLES_FONCTION » correspondant à la fonction et cliquer sur « Déclarer variable locale ». Le bouton « Ajouter RENVOYER valeur » permet lui de définir ce que renverra la fonction. Quand au bouton « Appeler Fonction », il sert à appeler une fonction qui ne renvoie pas de valeur.

Exemple avec le calcul du pgcd de deux entiers (méthode récursive) :

```
FONCTIONS_UTILISEES
FONCTION pgcd(p,q)
VARIABLES_FONCTION
DEBUT_FONCTION
| SI (p==0) ALORS
| | DEBUT_SI
| | RENVOYER q
| | FIN_SI
| SI (q==0) ALORS
| | DEBUT_SI
| | RENVOYER p
| | FIN_SI
| SI (q<=p) ALORS
| | DEBUT_SI
| | RENVOYER pgcd(p-q,q)
| | FIN_SI
| RENVOYER pgcd(p,q-p)
FIN_FONCTION
VARIABLES
m EST_DU_TYPE NOMBRE
n EST_DU_TYPE NOMBRE
result EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
| LIRE m
```

```
25:     LIRE n
26:     result PREND_LA_VALEUR pgcd(m,n)
27:     AFFICHER result
28: FIN_ALGORITHME
```

B.3.9 Récupération facile d'un code AlgoBox dans un traitement de texte

Pour copier un code AlgoBox facilement sans passer par les commandes d'exportation du menu *Fichier*, il suffit de :
- lancer la fenêtre de test de l'algorithme ;
- sélectionner le code à copier dans la page web de test (la partie supérieure de la fenêtre) ;
- faire « glisser » le code sélectionné vers son traitement de texte.

B.4 Quelques techniques classiques

B.4.1 Diviseur ?

- Condition pour vérifier si un entier P est un diviseur d'un entier N : `N%P==0`

B.4.2 Entier pair ou impair ?

- Condition pour vérifier si un entier N est pair : `N%2==0`
- Condition pour vérifier si un entier N est impair : `N%2!=0` (autre condition possible : `N%2==1`)

B.4.3 Entier pseudo-aléatoire compris entre 1 et N

- Pour obtenir un entier pseudo-aléatoire compris entre 1 et N :
  `... PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,N)` ou `... PREND_LA_VALEUR floor(N*random()+1)`
- Exemple pour la face d'un dé : `... PREND_LA_VALEUR ALGOBOX_ALEA_ENT(1,6)`

B.4.4 « Balayage » d'un intervalle

Deux méthodes peuvent être utilisées pour « balayer » un intervalle $[a\,;b]$ (a et b faisant partie du traitement).
Première méthode en entrant le nombre de subdivisions et en utilisant une boucle POUR... :

```
 1: VARIABLES
 2: i EST_DU_TYPE NOMBRE
 3: nbsubdivisions EST_DU_TYPE NOMBRE
 4: pas EST_DU_TYPE NOMBRE
 5: x EST_DU_TYPE NOMBRE
 6: a EST_DU_TYPE NOMBRE
 7: b EST_DU_TYPE NOMBRE
 8: DEBUT_ALGORITHME
 9:     LIRE a
10:     LIRE b
11:     LIRE nbsubdivisions
12:     pas PREND_LA_VALEUR (b-a)/nbsubdivisions
13:     POUR i ALLANT_DE 0 A nbsubdivisions
14:         DEBUT_POUR
15:         x PREND_LA_VALEUR a+i*pas
16:         traitement....
17:         FIN_POUR
18: FIN_ALGORITHME
```

Deuxième méthode en entrant directement le pas et en utilisant une structure TANT QUE... :

```
VARIABLES
pas EST_DU_TYPE NOMBRE
x EST_DU_TYPE NOMBRE
a EST_DU_TYPE NOMBRE
b EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE a
    LIRE b
    LIRE pas
    x PREND_LA_VALEUR a
    TANT_QUE (x<=b) FAIRE
        DEBUT_TANT_QUE
        traitement....
        x PREND_LA_VALEUR x+pas
        FIN_TANT_QUE
FIN_ALGORITHME
```

B.4.5 Suites numériques

D'un strict point de vue programmation, l'utilisation d'une liste pour manipuler les termes d'une suite dans un algorithme n'est guère optimal. Par contre, d'un point de vue pédagogique, la correspondance étroite entre terme d'une suite et terme d'une liste AlgoBox peut permettre de conforter la compréhension par les élèves du formalisme sur les suites numériques.
Pour modéliser une suite (U_n) à l'aide d'un algorithme AlgoBox, on peut utiliser une variable de type LISTE notée U.
Ainsi :

- le terme U_0 de la suite sera représenté par `U[0]` dans l'algorithme ;
- le terme U_1 de la suite sera représenté par `U[1]` dans l'algorithme ;
- etc...
- le terme U_n de la suite sera représenté par `U[n]` dans l'algorithme ;
- le terme U_{n+1} de la suite sera représenté par `U[n+1]` dans l'algorithme ;

Exemple avec une suite « récurrente » :
Sans utiliser de liste :

```
VARIABLES
n EST_DU_TYPE NOMBRE
U EST_DU_TYPE NOMBRE
i EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE n
    U PREND_LA_VALEUR 1
    POUR i ALLANT_DE 0 A n-1
        DEBUT_POUR
        U PREND_LA_VALEUR 1+2*U
        FIN_POUR
    AFFICHER U
FIN_ALGORITHME
```

En utilisant une liste AlgoBox :

```
VARIABLES
n EST_DU_TYPE NOMBRE
U EST_DU_TYPE LISTE
i EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE n
    U[0] PREND_LA_VALEUR 1
    POUR i ALLANT_DE 0 A n-1
        DEBUT_POUR
        U[i+1] PREND_LA_VALEUR 1+2*U[i]
        FIN_POUR
    AFFICHER U[n]
FIN_ALGORITHME
```

B.4.6 Échanger le contenu de deux variables

Pour échanger le contenu de deux variables `A` et `B`, il suffit d'utiliser une troisième variable `temp` :

```
temp PREND_LA_VALEUR A
A PREND_LA_VALEUR B
B PREND_LA_VALEUR temp
```

Exemple : tri d'une liste de valeurs par échange du minimum

```
 1: VARIABLES
 2: listenombre EST_DU_TYPE LISTE
 3: temp EST_DU_TYPE NOMBRE
 4: i EST_DU_TYPE NOMBRE
 5: min EST_DU_TYPE NOMBRE
 6: nbtermes EST_DU_TYPE NOMBRE
 7: DEBUT_ALGORITHME
 8:   LIRE nbtermes
 9:   //Entrer les "nbtermes" valeurs sous la forme valeur1:valeur2:etc...
10:   LIRE listenombre[1]
11:   POUR i ALLANT_DE 1 A nbtermes-1
12:     DEBUT_POUR
13:     min PREND_LA_VALEUR ALGOBOX_POS_MINIMUM(listenombre,i+1,nbtermes)
14:     SI (listenombre[i]>listenombre[min]) ALORS
15:       DEBUT_SI
16:       temp PREND_LA_VALEUR listenombre[min]
17:       listenombre[min] PREND_LA_VALEUR listenombre[i]
18:       listenombre[i] PREND_LA_VALEUR temp
19:       FIN_SI
20:     FIN_POUR
21:   POUR i ALLANT_DE 1 A nbtermes
22:     DEBUT_POUR
23:     AFFICHER listenombre[i]
24:     AFFICHER " "
25:     FIN_POUR
26: FIN_ALGORITHME
```

B.4.7 Afficher un message contenant du texte et des nombres

Il suffit pour cela d'utiliser une variable du type `CHAINE`.
Exemple :

```
 1: VARIABLES
 2: a EST_DU_TYPE NOMBRE
 3: b EST_DU_TYPE NOMBRE
 4: message EST_DU_TYPE CHAINE
 5: somme EST_DU_TYPE NOMBRE
 6: DEBUT_ALGORITHME
 7:   LIRE a
 8:   LIRE b
 9:   somme PREND_LA_VALEUR a+b
10:   message PREND_LA_VALEUR a.toString()+" + "+b.toString()+" est égal à "+somme.toString()
11:   AFFICHER message
12: FIN_ALGORITHME
```

Résultat :

```
***Algorithme lancé***
Entrer a : 4
Entrer b : 7
4 + 7 est égal à 11
***Algorithme terminé***
```

C Algorithmes supplémentaires

C.1 Second degré

Solutions de l'équation du second degré $ax^2 + bx + c = 0$.

```
 1: VARIABLES
 2: a EST_DU_TYPE NOMBRE
 3: b EST_DU_TYPE NOMBRE
 4: c EST_DU_TYPE NOMBRE
 5: delta EST_DU_TYPE NOMBRE
 6: x1 EST_DU_TYPE NOMBRE
 7: x2 EST_DU_TYPE NOMBRE
 8: DEBUT_ALGORITHME
 9:   LIRE a
10:   LIRE b
11:   LIRE c
12:   SI (a!=0) ALORS
13:     DEBUT_SI
14:     delta PREND_LA_VALEUR b*b-4*a*c
15:     SI (delta<0) ALORS
16:       DEBUT_SI
17:       AFFICHER "Pas de racines réelles"
18:       FIN_SI
19:     SI (delta==0) ALORS
20:       DEBUT_SI
21:       x1 PREND_LA_VALEUR -b/(2*a)
22:       AFFICHER "Une racine double : "
23:       AFFICHER x1
24:       FIN_SI
25:     SI (delta>0) ALORS
26:       DEBUT_SI
27:       x1 PREND_LA_VALEUR (-b-sqrt(delta))/(2*a)
28:       x2 PREND_LA_VALEUR (-b+sqrt(delta))/(2*a)
29:       AFFICHER "Première racine : "
30:       AFFICHER x1
31:       AFFICHER "Deuxième racine : "
32:       AFFICHER x2
33:       FIN_SI
34:     FIN_SI
35:   SINON
36:     DEBUT_SINON
37:     AFFICHER "Pas du second degré"
38:     FIN_SINON
39: FIN_ALGORITHME
```

C.2 Paramètres d'une série statistique

Calcul des paramètres statistiques sur série (valeurs,effectifs).

```
VARIABLES
valeurs EST_DU_TYPE LISTE
effectifs EST_DU_TYPE LISTE
nb_valeurs EST_DU_TYPE NOMBRE
i EST_DU_TYPE NOMBRE
liste_complete EST_DU_TYPE LISTE
j EST_DU_TYPE NOMBRE
k EST_DU_TYPE NOMBRE
moyenne EST_DU_TYPE NOMBRE
ecart_type EST_DU_TYPE NOMBRE
mediane EST_DU_TYPE NOMBRE
premier_quartile EST_DU_TYPE NOMBRE
troisieme_quartile EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE nb_valeurs
    LIRE valeurs[1]
    LIRE effectifs[1]
    k PREND_LA_VALEUR 1
    POUR i ALLANT_DE 1 A nb_valeurs
        DEBUT_POUR
        POUR j ALLANT_DE 1 A effectifs[i]
            DEBUT_POUR
            liste_complete[k] PREND_LA_VALEUR valeurs[i]
            k PREND_LA_VALEUR k+1
            FIN_POUR
        FIN_POUR
    moyenne PREND_LA_VALEUR ALGOBOX_MOYENNE(liste_complete,1,k-1)
    AFFICHER "Moyenne : "
    AFFICHER moyenne
    ecart_type PREND_LA_VALEUR ALGOBOX_ECART_TYPE(liste_complete,1,k-1)
    AFFICHER "Ecart-type : "
    AFFICHER ecart_type
    mediane PREND_LA_VALEUR ALGOBOX_MEDIANE(liste_complete,1,k-1)
    AFFICHER "Médiane : "
    AFFICHER mediane
    premier_quartile PREND_LA_VALEUR ALGOBOX_QUARTILE1_BIS(liste_complete,1,k-1)
    AFFICHER "Q1 : "
    AFFICHER premier_quartile
    troisieme_quartile PREND_LA_VALEUR ALGOBOX_QUARTILE3_BIS(liste_complete,1,k-1)
    AFFICHER "Q3 : "
    AFFICHER troisieme_quartile
FIN_ALGORITHME
```

C.3 Tabulation loi binomiale - Intervalle de fluctuation à 95%

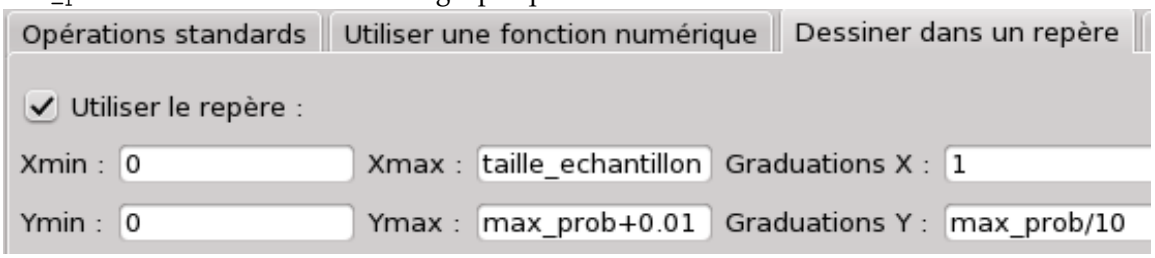
Tabulation loi binomiale et intervalle de fluctuation à 95% d'une proportion p.

```
VARIABLES
taille_echantillon EST_DU_TYPE NOMBRE
p EST_DU_TYPE NOMBRE
somme EST_DU_TYPE NOMBRE
k EST_DU_TYPE NOMBRE
a EST_DU_TYPE NOMBRE
b EST_DU_TYPE NOMBRE
fluct_min EST_DU_TYPE NOMBRE
fluct_max EST_DU_TYPE NOMBRE
prob EST_DU_TYPE NOMBRE
max_prob EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE taille_echantillon
    SI (taille_echantillon<1 OU taille_echantillon>100) ALORS
        DEBUT_SI
        AFFICHER "Calcul non autorisé"
        FIN_SI
    SINON
        DEBUT_SINON
        LIRE p
        a PREND_LA_VALEUR -1
        b PREND_LA_VALEUR -1
        max_prob PREND_LA_VALEUR 0
        POUR k ALLANT_DE 0 A taille_echantillon
            DEBUT_POUR
            prob PREND_LA_VALEUR ALGOBOX_LOI_BINOMIALE(taille_echantillon,p,k)
            SI (prob>max_prob) ALORS
                DEBUT_SI
                max_prob PREND_LA_VALEUR prob
                FIN_SI
            FIN_POUR
        AFFICHER "taille échantillon : "
        AFFICHER taille_echantillon
        AFFICHER "proportion p : "
        AFFICHER p
        AFFICHER "k -> p(X<=k) :"
        POUR k ALLANT_DE 0 A taille_echantillon
            DEBUT_POUR
            prob PREND_LA_VALEUR ALGOBOX_LOI_BINOMIALE(taille_echantillon,p,k)
            somme PREND_LA_VALEUR somme+prob
            AFFICHER k
            AFFICHER " -> "
            AFFICHER somme
            SI (somme<=0.025) ALORS
                DEBUT_SI
                TRACER_SEGMENT (k,0)->(k,prob)
                FIN_SI
            SI (somme>0.025 ET somme<0.975) ALORS
                DEBUT_SI
                TRACER_SEGMENT (k,0)->(k,prob)
                SI (a==-1) ALORS
                    DEBUT_SI
                    a PREND_LA_VALEUR k
                    FIN_SI
                FIN_SI
            SI (somme>=0.975) ALORS
                DEBUT_SI
                SI (b==-1) ALORS
                    DEBUT_SI
                    b PREND_LA_VALEUR k
                    TRACER_SEGMENT (k,0)->(k,prob)
                    FIN_SI
                SINON
                    DEBUT_SINON
                    TRACER_SEGMENT (k,0)->(k,prob)
                    FIN_SINON
                FIN_SI
            FIN_POUR
            fluct_min PREND_LA_VALEUR a/taille_echantillon
```

```
70:     AFFICHER "a : "
71:     AFFICHER a
72:     AFFICHER "b : "
73:     fluct_max PREND_LA_VALEUR b/taille_echantillon
74:     AFFICHER b
75:     AFFICHER "intervalle de fluctuation : [ "
76:     AFFICHER fluct_min
77:     AFFICHER " ; "
78:     AFFICHER fluct_max
79:     AFFICHER " ] "
80:   FIN_SINON
81: FIN_ALGORITHME
```

Notes :

— `max_prob` sert à définir la fenêtre graphique :



— Les « bâtons » situés entre `a` et `b` sont dessinés en bleu (lignes 50 et 61), les autres sont tracés en rouge (lignes 46 et 65).