Cours d'algorithmique et Algobox

Définition 1 : Un ***algorithme*** est une succession d'instructions qui prend un ensemble de valeurs comme entrée (input) suivi d'un traitement des valeurs et enfin délivre un ensemble de valeurs comme sortie (output).et ceci, afin de résoudre un problème donné.
La notion d'algorithme a été abordée pour la première fois par :
Khawarizmi mathématicien persan (Iran d'aujourd'hui) au 9ème siècle

Le but de l'enseignement d'algorithmique est de résoudre des problèmes mathématiques en créant des algorithmes.

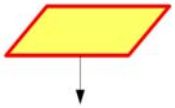
Exemple 1 :- **Plus Grand Commun Diviseur (PGCD)**
entrée (Input en anglais): deux entiers a et b ;
Traitement : celui d'Euclide (par exemple) ;
sortie (Output en anglais): la valeur de pgcd(a,b).

Organigramme

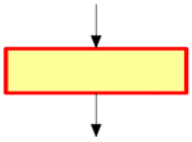
On peut présenter un algorithme sous forme graphique, par ce qu'on appelle l'organigramme :

*- Le **rectangle arrondi** pour désigner le début ou la fin du programme.*

*- Le **parallélogramme** correspond aux variables à déclarer :*



*- Les **rectangles** correspondent à des ordres à exécuter :*



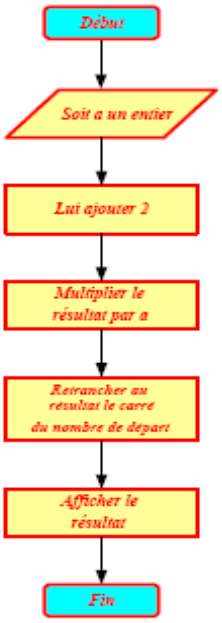
- On parcourt les instructions dans le sens des flèches :
- Les aiguillages (choix imposés) sont symbolisés par un losange (on les verra avec l'introduction des instructions conditionnelles) :

Un algorithme mathématique en langage naturel

Exemple 2 :

Un calcul simple exprimé à l'aide d'un algorithme (***en langage naturel***) de calculs en 5 étapes :

- `Lui ajouter 2 ;`
- `Multiplier cette somme par le nombre de départ ;`
- `Retrancher au résultat le carré du nombre de départ ;`
- `Afficher le résultat obtenu.`



Une autre façon pour présenter cet algorithme (forme codée) :

ou encore

variables
a, b, c : entiers ;
Début
saisir a ;
b ← *a*+2 (b prend la valeur a+2)
c ← *b* × *a*
d prend la valeur $c - a^2$
Afficher *d.*
Fin

Cet algorithme sera traité un peu plus tard à l'aide d'un programme TI.

Ce programme de calcul est un algorithme. Il comporte 3 phases :

1ère phase - Input (entrées): il faut introduire des données, c'est l'étape 1
2ème phase - Traitement : on travaille ces données, ce sont les étapes 2 à 4
3ème phase - Output (sorties): on annonce un résultat, c'est l'étape 5

La phase de ***traitement*** se structure à l'aide des instructions :

- Les ***instructions de base*** qui sont des calculs effectués à partir des entrées.
- Les ***boucles*** qui sont des actions à effectuer un certain nombre de fois.
- Les ***instructions conditionnelles*** qui dépendent de la réponse à un test.

Un algorithme mathématique en langage naturel

Voici un algorithme muni de ses instructions d'entrée et de sortie et qu'il comporte un calcul itératif, qu'on nommera une boucle qui commence par un ***pour*** et se termine par un ***fin pour***.
La variable est appelée variable de boucle et on doit définir son minimum et son maximum.
.
En général, on appelle cela ***une boucle*** « pour»

Exemple 3 – La somme S, des cubes des entiers de 1 à n(voir le programme Algobox page 8)

On appliquera l'algorithme suivant :

```
Étape 1 : Lire n.
Étape 2 : Poser S=0.
Étape 3 : Pour i variant de 1 à n, faire S=S+i^3 .
Étape 4 : Fin pour.
Étape 5 : Afficher la valeur de S.
```

Les différents langages / environnements de programmation

Définition 2 : Il s'agit d' un ensemble d'instructions et de règles utilisées afin qu'un ordinateur ou une calculatrice puisse résoudre un problème donné.

Programmer une calculatrice consiste à lui « préciser » ce qu'elle doit faire, en sachant qu'elle ne « comprend » pas notre langage, mais qu'elle peut seulement effectuer un traitement en plusieurs étapes ou encore suite d'instructions longues, exprimées par une information binaire (symbolisée par 0 ou 1) et qui s'appelle un bit (en anglais bit).

Un groupe de huit bits s'appelle un octet (en anglais, byte c'est-à-dire huit bits), en respectant de manière stricte un ensemble de conventions fixées à l'avance par un langage informatique propre à la machine.

Cette année, on travaille essentiellement avec le logiciel Algobox (1) et les calculatrices TI83+ et Ti84+.

Exercice 1:

On considère l'algorithme ci-contre
1) En introduisant x=3, quelle valeur de x s'affiche à la fin de l'algorithme ?
2) Pensez vous que x ne change de valeur, à la fin du traitement ?

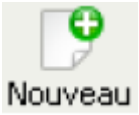
Solution :

variables
x : entiers ;
Début
Saisir *x ;*
$x \leftarrow x$-1
$x \leftarrow 2 \times x$
Afficher *x*
Fin

*(1) **Algobox** est un logiciel libre et gratuit d'aide qui nous permet d'élaborer et exécuter des algorithmes au programme de la classe de seconde en mathématiques.*

L'utilisation d'Algobox

Après avoir installé le logiciel Algobox, commencez par ouvrir le logiciel, puis créez un nouveau programme :

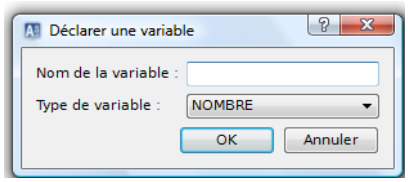
Pour créer un nouveau programme :	*Pour introduire et séparer les instructions :*	*Pour modifier un programme qui existe déjà :*	*Pour exécuter un programme :*
On utilise le bouton : 	On utilise le bouton :  à l'endroit choisit	On utilise le bouton : 	On utilise le bouton : 

Les variables

Définition 3 : Une variable correspond à un emplacement de la mémoire d'une calculatrice ou d'un ordinateur où on stocke une information modifiable.
Une variable est constituée d'un nom et d'une valeur. Toute variable doit d'abord être déclarée avant d'être utilisée.

Attention :

Le nom des variables ne doit contenir ni d'espaces ni d'accents. On ne peut pas non plus utiliser certains termes réservés aux codes Algobox comme : NOMBRE.

Pour déclarer les variables :	Pour introduire et séparer les instructions :
On utilise le bouton : 	On utilise la fenêtre : 

Entrées et sorties dans Algobox

Définition 4 : Les commandes d'entrée de valeurs sont celles, introduites par l'utilisateur, ces valeurs sont, soit un nombre, soit un texte, ou encore un caractère.

Exemple 4 : Voici l'instruction d'Algobox qui nous permet de saisir une valeur :

Commandes d'affichage

Définition 5 : Les commandes d'affichage sont utilisées affin d'afficher à l'écran la valeur d'une variable ou un texte.

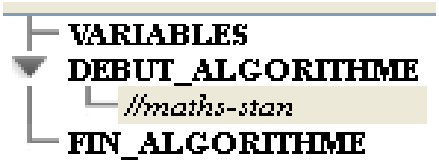
Exemple 5 :

```
AFFICHER "maths-stan"
AFFICHER "S="
AFFICHER S
```

Ici pour séparer l'affichage du texte et la valeur de S, par une ligne, il faut cocher le champ : « *Ajouter un retour à la ligne* » dans la boîte de dialogue *Afficher le message*.

Ajout de commentaires dans le code de l'algorithme

On peut ajouter un commentaire dans le code de l'algorithme à partir d'une ligne vierge grâce au bouton Commentaire.



Exercice 2 : *un algorithme erroné à corriger*

(*Exercice proposé par l'équipe de mathématiques Académie de Rouen*)

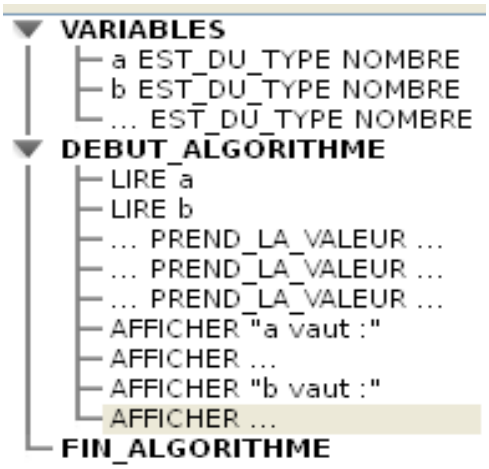
L'algorithme suivant a pour objectif de demander à l'utilisateur deux valeurs numériques stockées dans les variables *a* et *b* puis d'échanger le contenu des deux variables :

Code de l'algorithme en Algobox :

```
VARIABLES
    a EST_DU_TYPE NOMBRE
    b EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE a
    LIRE b
    a PREND_LA_VALEUR b
    b PREND_LA_VALEUR a
    AFFICHER "a vaut :"
    AFFICHER b
    AFFICHER " b vaut :"
    AFFICHER a
FIN_ALGORITHME
```

1) Cet algorithme ne donne pas satisfaction. Pourquoi ? Que fait-il réellement ?
2) Que faire pour corriger cet algorithme ?
   Faites un essaie avec *a*=5 et *b*=9.

Solution : compléter le **Code de l'algorithme** suivant, pour que le programme fonctionne :



Instructions conditionnelles TANT QUE et SI ALORS

Pour exécuter des instructions sous certaines conditions, on utilise en Algobox, les deux Instructions conditionnelles : ***TANT QUE et SI ALORS*** définies :

- TANT QUE

Cette instruction exécute la liste d'instructions ***tant que*** la condition est réalisée.

- SI <telle condition est réalisée> ***ALORS*** <exécuter telle l'action >

Exemples de syntaxe pour la condition (tirés de http://www.xm1math.net/algobox/doc.html **):**

- Pour ***vérifier si*** *x* est égal à 2, la condition à écrire est : *x*==2

Attention : La double égalité = = est pour indiquer qu'on n'est pas en présence d'une affectation de variable mais il s'agit d'un test d'égalité : la variable *x* est-elle égale à 2 ?

- Pour vérifier si *x* est ***différent de*** 2, la condition à écrire est : *x*!=2
- Pour vérifier si *x* est ***strictement inférieur à*** 2, la condition à écrire est : *x*<2
- Pour vérifier si *x* est ***inférieur ou égal à*** 2, la condition à écrire est : *x*<=2
- Pour vérifier si *x* est ***strictement supérieur à*** 2, la condition à écrire est : *x*>2
- Pour vérifier si *x* est ***supérieur ou égal à 2***, la condition à écrire est : *x*>=2

Il est possible de combiner plusieurs conditions avec ET et OU :

- La condition à écrire pour vérifier que *x* est ***strictement compris entre*** 1 et 5 est : *x*>1 **ET** *x*<5
- La condition à écrire pour vérifier que *x* ***est égal*** à 3 **OU** à 5 est : *x*==3 OU *x*==5.

Il est aussi possible d'utiliser des calculs dans les expressions conditionnelles.
Exemple de condition : *x*<sqrt(n)

En résumé :

Syntaxes Algobox pour les opérations mathématiques

syntaxe Algobox	Syntaxe des opérations mathématiques
Math.PI	π
sqrt(x)	$\sqrt{x}$
pow(x,y)	x^y
sin(x)	sin(x) (x en radians)
cos(x)	cos(x) (x en radians)
tan(x)	tan(x) (x en radians)
n%p	Reste de la division euclidienne de n par p
floor(x)	Partie entière de x
abs(x)	$\lvert x \rvert$
random()	Nombre pseudo-aléatoire compris entre 0 et 1

Exemple 6: floor(1+100* random()) permet d'obtenir un nombre entier compris entre 1 et 100 suivant la loi équirépartie.

CODE DE L'ALGORITHME :

```
1    VARIABLES
2      n EST_DU_TYPE NOMBRE
3    DEBUT_ALGORITHME
4      n PREND_LA_VALEUR floor(1+100*random())
5      AFFICHER n
6    FIN_ALGORITHME
```

RÉSULTATS :

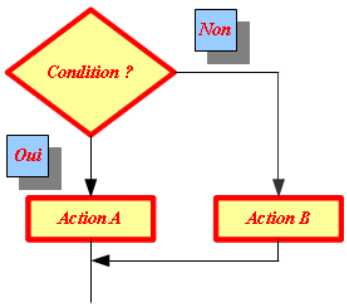
```
***Algorithme lancé***
73
```

Syntaxes Algobox pour les conditions logiques

syntaxe Algobox	Condition logique
x<y	$x < y$
x>y	$x > y$
x<=y	$x \leq y$
x>=y	$x \geq y$
x!=y	$x \neq y$

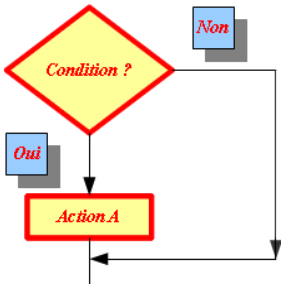
Voici quelques pistes pour réaliser des algorithmes corrects :

1° Si Condition Alors Action A sinon Action B Finsi



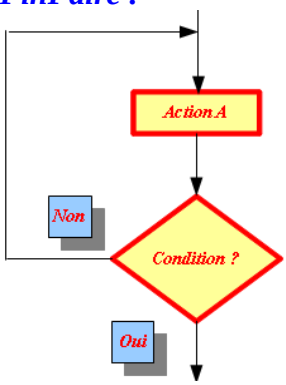
2° Si..... Alors .....

Si Condition Alors Action A sinon



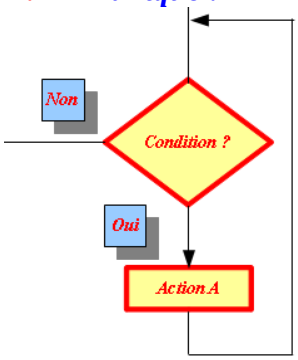
3° Faire.....jusqu'à....

Faire Action A jusqu'à. Condition FinFaire :



4° Tantque.....Faire.....

Tantque Condition Faire.. Action A. FinTantque :



Exemple 7 : On sait que la division euclidienne de l'entier a par b (a>b) a un sens que lorsque b est différent de zéro. Voici donc un algorithme pour la division euclidienne de l'entier a par b avec q comme quotient et r comme reste : $a = b \times q + r$

Entrée
Saisir a
Saisir b
Traitement
Tant que $r \geq b$, réitérer la procédure suivante
$q \leftarrow q + 1$ (q prend la valeur $q + 1$)
$r \leftarrow r - b$
Fin Tant que
Sortie
Afficher q.
Afficher r

Suite la page suivante

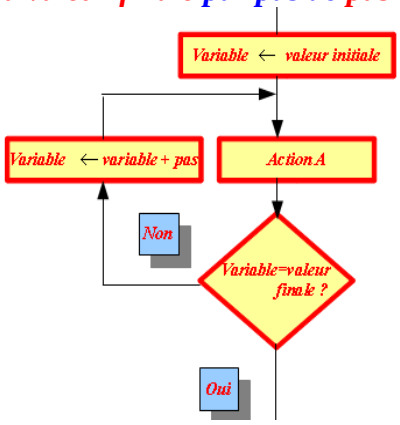
Voici l'équivalent en Algobox :

```
VARIABLES
    a EST_DU_TYPE NOMBRE
    b EST_DU_TYPE NOMBRE
    q EST_DU_TYPE NOMBRE
    r EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    AFFICHER "LA DIVISION D'UN ENTIER a PAR UN ENTIER b"
    AFFICHER "Introduire a>=0 :"
    LIRE a
    AFFICHER a
    AFFICHER "Introduire b>0 :"
    LIRE b
    AFFICHER b
    q PREND_LA_VALEUR 0
    r PREND_LA_VALEUR a
    TANT_QUE (r>=b) FAIRE
        DEBUT_TANT_QUE
        q PREND_LA_VALEUR q+1
        r PREND_LA_VALEUR r-b
        FIN_TANT_QUE
    AFFICHER "Le quotient q est :"
    AFFICHER q
    AFFICHER "Le reste r est :"
    AFFICHER r
FIN_ALGORITHME
```

5° Boucle : Pour.....Faire.....

Une telle boucle n'est utilisée que si on connaît à l'avance le nombre de répétitions à effectuer, à l'aide d'une variable (variable du type NOMBRE augmentée de 1 à chaque fois, et déclarée au préalable) qu'on utilise comme compteur.

Pour variable De valeur initiale à la valeur finale par pas de pas .Faire l'action A Fin pour



Exercice 3: Que calcule l'algorithme suivant ?

```
VARIABLES
    n EST_DU_TYPE NOMBRE
    i EST_DU_TYPE NOMBRE
    S EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
    LIRE n
    S PREND_LA_VALEUR 0
    POUR i ALLANT_DE 1 A n
        DEBUT_POUR
        S PREND_LA_VALEUR pow(i,3)+S
        FIN_POUR
    AFFICHER "maths-stan"
    AFFICHER "S="
    AFFICHER S
FIN_ALGORITHME
```

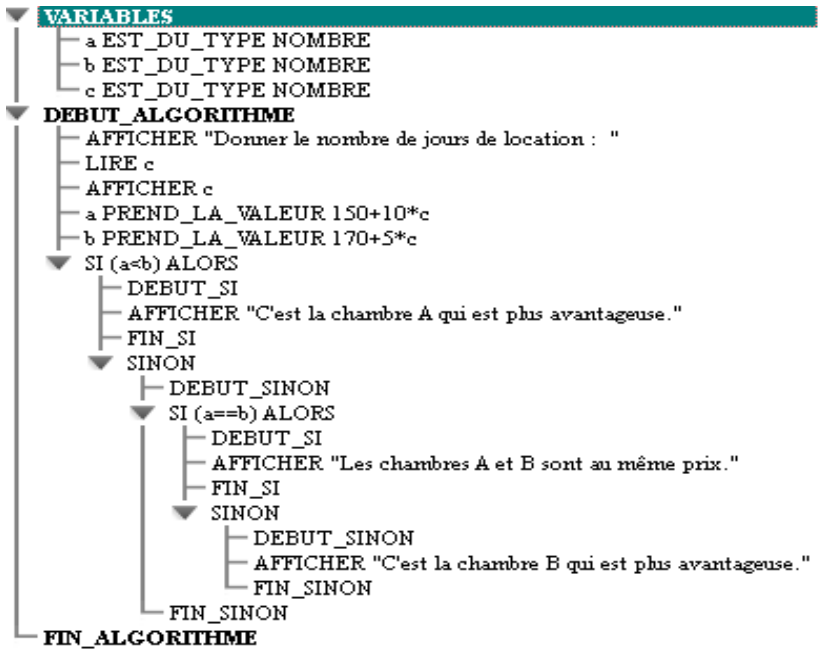
Exercice 4 : Dans un hôtel, sur les prix affichés pour deux chambres ayant les mêmes caractéristiques, on peut lire :

Chambre A : prix fixe de la chambre 150 € + 10 € de connexion internet par jour ;

Chambre B : prix fixe de la chambre 170 € + 5 € de connexion internet par jour ;

On souhaite concevoir un algorithme permettant de trouver le prix le plus avantageux entre les deux chambres en fonction du nombre de jours de location.

```
Variables
 a, b, c : entiers ;
Début
 Saisir c (nombre de jours);
 a ← 150 + 10 × c ;
 b ← 170 + 5 × c ;
 Si a<b alors afficher « Chambre A »
   Sinon Si A=B alors afficher « Chambre A  ou Chambre B » ;
            sinon afficher « Chambre B »
    FinSi ;
 FinSi ;
Fin
```

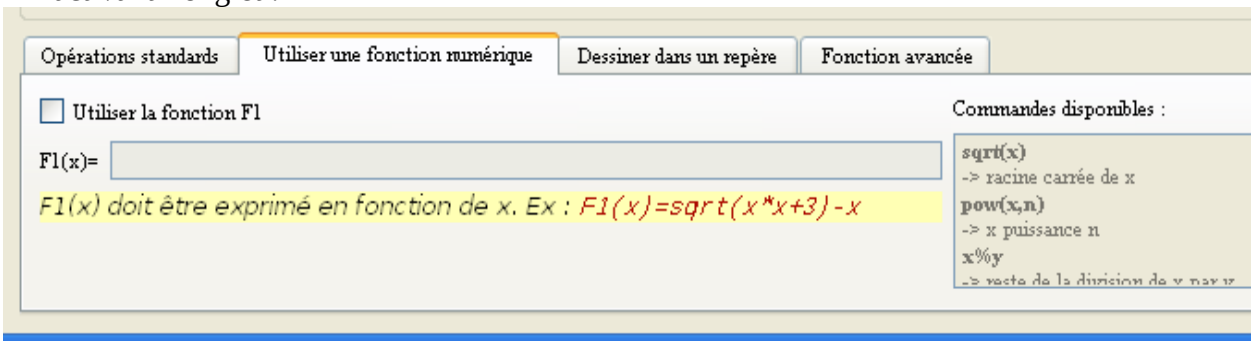


En réalisant ce programme et en le testant, trouver le nombre de jours pour lesquels la location de la chambre A qui est plus avantageuse.

Fonction et sa représentation graphique

Utilisation d'une fonction numérique:

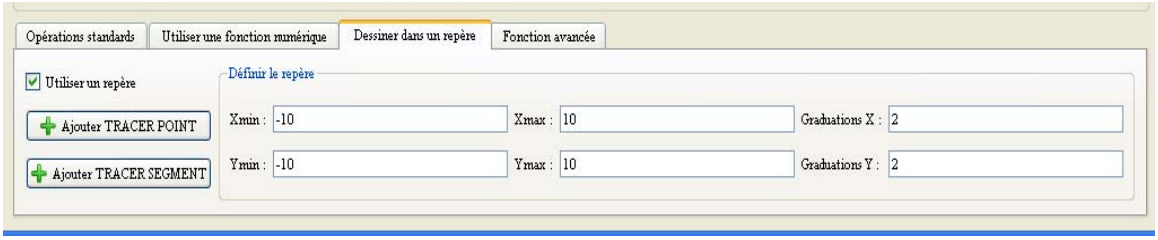
En activant l'onglet :



on peut utiliser l'image d'un 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, on peut utiliser le code : F1(nb) (cela peut se faire dans une affectation ou dans une expression conditionnelle).

Tracer des points et des segments dans un repère

En activant l'onglet :

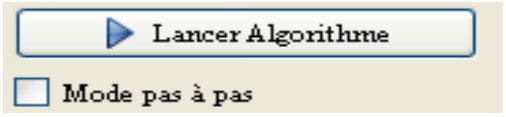


On peut alors inclure dans le code de l'algorithme des instructions pour tracer des points (qui forment la courbe) et des segments dans ce repère en utilisant les boutons **Ajouter TRACER POINT** et Ajouter **TRACER SEGMENT** .

Attention : Si les instructions à répéter comportent du tracé graphique, il faut limiter le nombre d'itérations à un nombre moins de mille, si non l'exécution du programme risque de prendre beaucoup de temps.

Exécution d'un algorithme en mode "pas à pas"

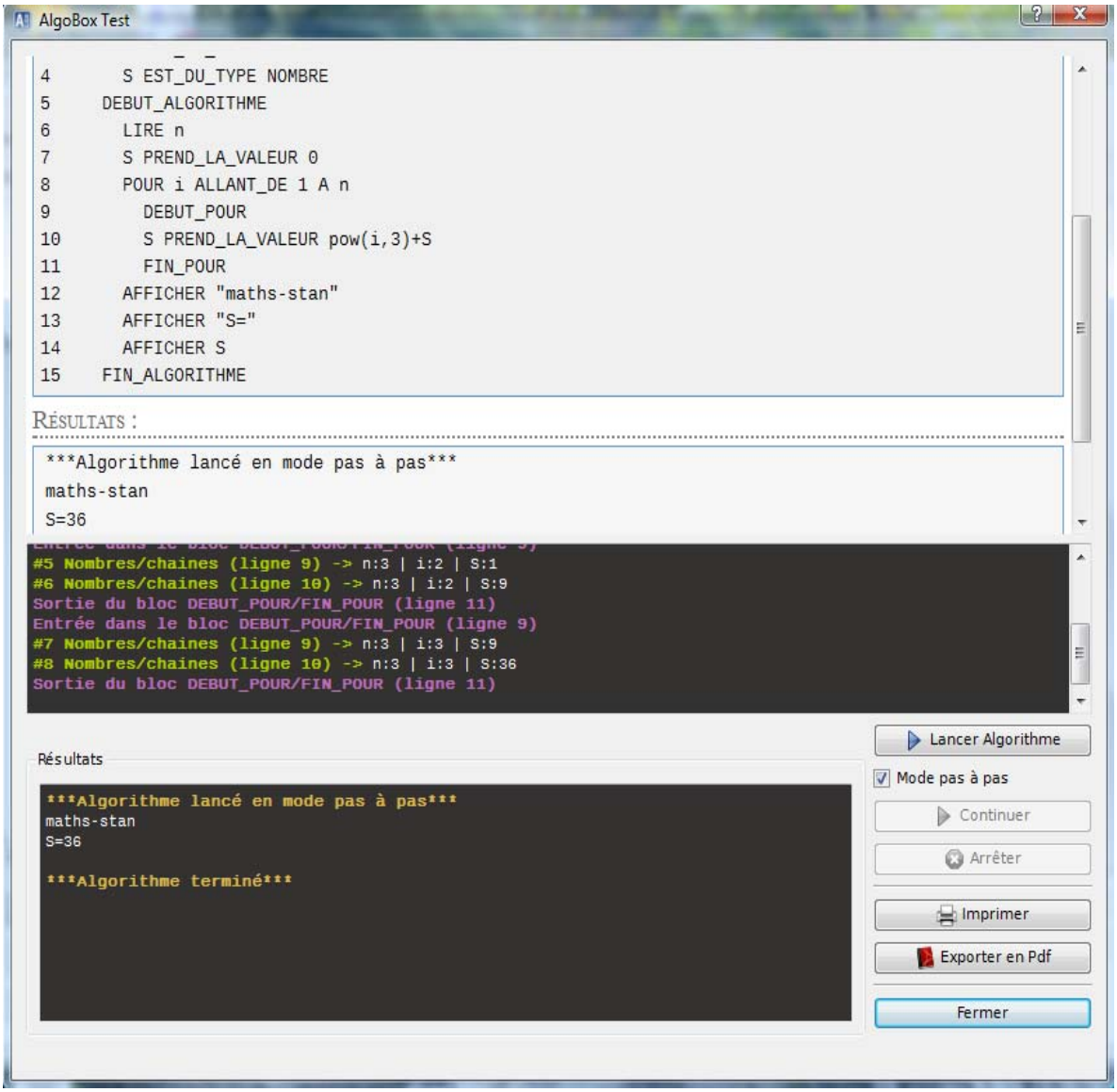
- Pour lancer le mode "pas à pas", il suffit de cocher la case correspondante avant de cliquer sur le bouton Lancer Algorithme.



- La ligne de l'algorithme correspondante est signalée en rouge.
- L'entrée et la sortie de chaque bloc **SI, SINON, TANT QUE et POUR** est systématiquement signalée.

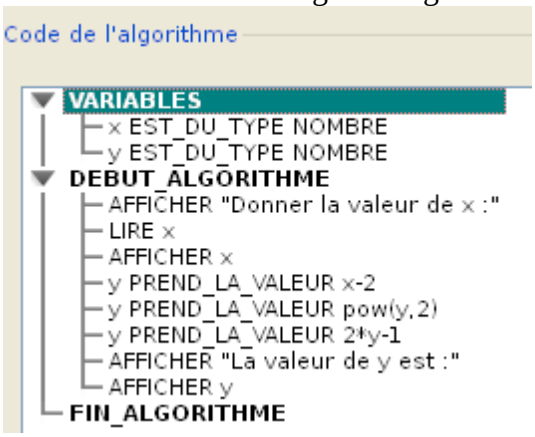
Comment rectifier les erreurs de programmation avec AlgoBox ?
Déboguer avec AlgoBox

- Avec Algobox, lors de l'arrêt de l'exécution d'un algorithme suite à une erreur, la dernière ligne exécutée est signalée automatiquement.
- Une erreur de calcul entraîne automatiquement l'arrêt de l'exécution : la ligne où figure le calcul est signalée.
- Pour déboguer un algorithme qui ne donne pas les résultats souhaités, il suffit d'exécuter l'algorithme en mode "pas à pas" , ce qui nous permet d'afficher étape pas étape le déroulement du programme.



Exercice 5:

On réalise l'algorithme suivant à l'aide du logiciel Algobox :



Attention : dans le code de l'algorithme précédent, on a utilisé la syntaxe **pow(*x*,2)** au lieu de ***x*^2** pour les puissances.

1° Utiliser l'algorithme précédent pour compléter le tableau suivant :

x	**-4**	**-2**	**0**	**1**	**3**
***f* (*x*)**					

2° Utiliser l'algorithme pour mettre en évidence l'expression finale de $f(x)$.

3° Démontrer que f est décroissante sur l'intervalle : $]-\infty,2]$ et croissante sur $[2,+\infty[$.

Exercice 6:

1) Que calcule l'algorithme suivant ?

variables

x_A , y_A , x_B , y_B , x_C , y_C , a , b c : réels ;

Début

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

$a \leftarrow (x_A - x_C)^2 + (y_A - y_C)^2$;

$b \leftarrow (x_A - x_B)^2 + (y_A - y_B)^2$;

$c \leftarrow (x_C - x_B)^2 + (y_C - y_B)^2$;

Afficher a, *b*, *c*.

Fin

2) Réaliser cet algorithme à l'aide d'Algobox.

3) Soit ***A***(3 ;8), ***B***(−1 ;0) et ***C***(−5 ;2), trois point d'un plan muni d'un repère orthonormé. Tester le programme réalisé pour les points précédents et en déduire la nature du triangle ***ABC***.

Exercice 7:

1) Que calcule l'algorithme suivant ?

```
variables
 x_u , y_u , x_v , y_v , a , b :réels ;
Début
Afficher (« Entrer les coordonnées du vecteur u ») ;
Saisir  x_u , y_u ;
Afficher (« Entrer les coordonnées du vecteur v ») ;
Saisir  x_v , y_v ;
 a ← x_u / x_v  ;
 b ← y_u / y_v  ;
Si a=b alors afficher ( « Les vecteurs u et v sont ..... ») ;
                afficher ( «avec le rapport de ..... b=») ;
Sinon afficher ( « Les vecteurs u et v ne sont pas ..... ») ;
 Finsi ;
Fin
```

2) Réaliser cet algorithme à l'aide d'Algobox.

3) Soient $\overrightarrow{u_1}\begin{pmatrix}-2\\6\end{pmatrix}$ et $\overrightarrow{v_1}\begin{pmatrix}1\\-3\end{pmatrix}$ d'une part et $\overrightarrow{u_2}\begin{pmatrix}3\\2\end{pmatrix}$ et $\overrightarrow{v_2}\begin{pmatrix}6\\-4\end{pmatrix}$ d'autre part, deux couples de vecteurs.

Tester le programme réalisé pour les deux couples de vecteurs précédents, que peut-on dire de ces deux couples de vecteurs.