

L'objectif est de rendre les élèves capables :

- De décrire certains algorithmes en langage naturel.
- D'en réaliser quelques uns, tableur ou petit programme sur la calculatrice.
- D'interpréter quelques algorithmes plus complexes.

Le logiciel utilisé pour les exemples suivants est le logiciel ALGOBOX. C'est un logiciel très simple et très rapide de prise en main et qui semble parfaitement adapté à la découverte de l'algorithmique.

Les activités suivantes sont de difficultés progressives et qui permettent à un élève de seconde de découvrir l'algorithmique.

Il ne faut pas hésiter à utiliser le mode « pas à pas » du logiciel AlgoBox car il permet de faire un parfait débogage.

Chacune des activités proposées commence par la mise en place d'un algorithme suivi de quelques exercices.

Pour prolonger ces activités et faire de la programmation à un autre niveau avec des constructions de procédures de fonctions etc, il faudra utiliser des langages standards comme le VISUAL BASIC ou le PASCAL avec la plateforme DELPHI ou bien encore avec le langage C avec C++.

Une fois l'algorithme terminé dans AlgoBox, si celui-ci ne fonctionne pas, n'hésitez pas à utiliser le mode PAS A PAS de ce logiciel pour voir comment évolue le contenu de toutes vos variables.

Pour bien comprendre, prenons l'exemple suivant :

on souhaite préparer des biscottes beurrées avec de la confiture. On est obligé pour cela de procéder comme suit :

Prendre les biscottes et les poser sur la table
Étaler du beurre sur les biscottes
Étaler de la confiture sur les biscottes beurrées

Il est bien évident qu'on ne peut pas intervertir les étapes... ou bien attention au résultat !

- On a ainsi écrit, sans le savoir, ce que l'on appelle un algorithme.

▶ Un algorithme est un processus permettant d'aboutir de façon automatique au résultat d'une action ou à la résolution d'un problème **en un nombre fini** d'étapes.

- Les algorithmes sont particulièrement utiles en Mathématiques, pour effectuer des calculs, par exemple.
- Écrire un algorithme consiste à donner une méthode détaillée décrivant toutes les étapes d'une tâche à accomplir, réparties en trois phases :

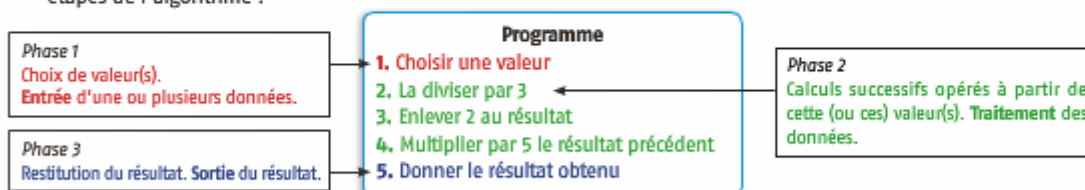
- L'élément (ou les éléments) dont on part : **les entrées**
- Les actions à effectuer sur ces éléments : **le traitement**
- Le résultat obtenu : **les sorties**

- Voici un programme de calcul vu au collège :

1. Choisir une valeur
2. La diviser par 3
3. Enlever 2 au résultat
4. Multiplier par 5 le résultat précédent
5. Donner le résultat obtenu

Ce programme de calcul est un algorithme, puisqu'on a décrit les différentes étapes d'un calcul permettant d'obtenir un résultat. La description de cet algorithme permet d'automatiser le calcul proposé grâce à une calculatrice ou un logiciel.

- Ce programme de calcul se décompose également en trois *phases*. Chacune d'elles correspond à une ou plusieurs étapes de l'algorithme :



- Ces trois phases, notamment la phase du traitement, se structurent à l'aide d'instructions :
 - Les **instructions de base** (calculs à partir des entrées), détaillées **page 8**.
 - Les **instructions conditionnelles** (qui dépendent de la réponse à un test), détaillées en **page 9**.
 - Les **boucles et itérateurs** (actions à effectuer un certain nombre de fois), détaillées **page 10**.

Présentation de l'algorithme

Algo vu au college|

Code de l'algorithme

```

▼ VARIABLES
  A EST_DU_TYPE NOMBRE
  Resultat EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  AFFICHER "Donner la valeur de A"
  LIRE A
  AFFICHER A
  A PREND_LA_VALEUR A/3
  AFFICHER "La valeur de A/3 est"
  AFFICHER A
  A PREND_LA_VALEUR A-2
  AFFICHER "La valeur de A/3-2 est"
  AFFICHER A
  Resultat PREND_LA_VALEUR A*5
  AFFICHER "La valeur de (A/3-2)*5 est"
  AFFICHER Resultat
FIN_ALGORITHME

```

```

#1 Nombres/chaines (ligne 6) -> A:18 | Resultat:0
#2 Nombres/chaines (ligne 8) -> A:6 | Resultat:0
#3 Nombres/chaines (ligne 11) -> A:4 | Resultat:0
#4 Nombres/chaines (ligne 14) -> A:4 | Resultat:20

```

Résultats

```

La valeur de A/3 est
6
La valeur de A/3-2 est
4
La valeur de (A/3-2)*5 est
20

***Algorithme terminé***

```

SOMMAIRE.

- I. Première approche : La division.
- II. PGCD de deux nombres entiers.
- III. Travailler sur quatre semaines.
- IV. Les chaînes de caractères : les palindromes.
- V. Le jeu du c'est plus – c'est moins.
- VI. Dans la foulée : la dichotomie.
- VII. Moyenne, écart-type, tri et médiane.
- VIII. Un peu de probabilités : la somme de deux dés.

I°) Première approche : la division.

Entrer deux nombres entiers A et B et de récupérer le quotient Q de ces deux nombres.

Méthode :

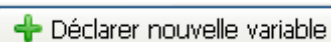
Demander la saisie du nombre A.
 Demander la saisie du nombre B.
 Calculer le quotient A/B et le mettre dans Q.
 Afficher la valeur de Q.

Variables utilisés :

Les trois variables numériques A, B et Q.

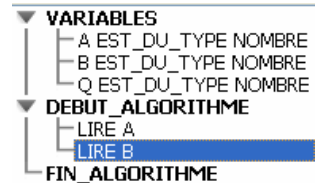
Réalisation de l'algorithme :

- 1) Lancer le logiciel Algobox
- 2) Cliquer sur déclarer une nouvelle variable :

 Déclarer nouvelle variable

, ce qui donne

- 3) Taper la variable A en laissant bien NOMBRE comme TYPE DE VARIABLE, comme indiquée ci-dessus.
- 4) Cliquer alors sur le bouton Nouvelle ligne ou bien appuyer sur la combinaison de touche [CTRL] [Entrée], puis cliquer sur le bouton lire variable pour demander la saisie du nombre A .
- 5) Faire la même opération pour la saisie du nombre B. A cette étape votre algorithme doit être comme représentée sur la figure ci-contre. Il nous faut , maintenant que les deux variables A et B sont bien en mémoire , calculer le quotient en prenant bien soin que le diviseur B ne soit pas nul.
- 6) Cliquer sur nouvelle ligne puis sur le bouton Affecter valeur à variable. Puis choisir la variable Q et à la suite de prend la valeur taper A/B, et il nous reste à afficher le quotient obtenu.
- 7) Rajouter une nouvelle ligne, puis cliquer sur ajouter afficher message et taper alors le message suivant « La valeur du quotient A/B est ». Ajouter une nouvelle ligne puis cliquer alors sur ajouter afficher variable et choisir la variable Q, en ajoutant un retour à la ligne.



L'algorithme final est alors le suivant :

Présentation de l'algorithme

Quotient de deux entiers.

Code de l'algorithme

```

VARIABLES
├── A EST_DU_TYPE NOMBRE
├── B EST_DU_TYPE NOMBRE
└── Q EST_DU_TYPE NOMBRE
DEBUT_ALGORITHMME
├── LIRE A
├── LIRE B
├── Q PREND_LA_VALEUR A/B
├── AFFICHER "La valeur du quotient A/B est "
├── AFFICHER Q
└── FIN_ALGORITHMME

```

- 8) Il s'agit maintenant de tester cet algorithme , pour cela il faut cliquer sur le bouton Tester Algorithme, puis sur le bouton Lancer l'algorithme et le résultat s'affiche comme dans la fenêtre ci-dessous après avoir entrée les nombres 10 et 3 :

Résultats

```

***Algorithme lancé***
La valeur du quotient A/B est
3.333333
***Algorithme terminé***

```

Quelques améliorations pour cet algorithme :

Il serait agréable d'avoir une petite explication avant de saisir les deux entiers afin d'éviter de taper un diviseur égal à zéro. Pour cela on introduit une nouvelle ligne deux fois et on obtient l'algorithme ci-dessous :

Quotient de deux entiers.

Code de l'algorithme

```

VARIABLES
├── A EST_DU_TYPE NOMBRE
├── B EST_DU_TYPE NOMBRE
└── Q EST_DU_TYPE NOMBRE
DEBUT_ALGORITHMME
├── AFFICHER "Pour le quotient des deux nombres, donner votre dividende A"
├── LIRE A
├── AFFICHER "Donner votre diviseur B"
├── LIRE B
├── Q PREND_LA_VALEUR A/B
├── AFFICHER "La valeur du quotient A/B est "
├── AFFICHER Q
└── FIN_ALGORITHMME

```

Résultats

```

***Algorithme lancé***
Pour le quotient des deux nombres, donner votre dividende A
Donner votre diviseur B
La valeur du quotient A/B est
3.333333
***Algorithme terminé***

```

La structure algorithmique TANT QUE :

Il est assez risqué de laisser l'utilisateur saisir un diviseur égal à zéro, pour cela nous allons demander de lire B tant que la valeur de B est de zéro.

Méthode :

Insérer une ligne après la ligne AFFICHER « Donner votre diviseur B », puis cliquer sur le bouton AJOUTER TANT QUE. Comme condition taper $B == 0$, en faisant bien attention de mettre deux signes = !.

Ce double = est pour indiquer que nous sommes pas sur une affectation de variable mais sur un test d'égalité : la variable B est-elle égale à zéro ?

Entre les deux instructions de DEBUT et FIN il faut ajouter la ligne LIRE B et supprimer l'ancienne ligne, et on obtient alors l'algorithme suivant

Présentation de l'algorithme

Quotient de deux entiers.

Code de l'algorithme

```

VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  Q EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER "Pour le quotient des deux nombres, donner votre dividende A"
  LIRE A
  AFFICHER "Donner votre diviseur B"
  TANT_QUE (B==0) FAIRE
    DEBUT_TANT_QUE
      LIRE B
    FIN_TANT_QUE
  Q PREND LA_VALEUR A/B
  AFFICHER "La valeur du quotient A/B est "
  AFFICHER Q
FIN_ALGORITHME

```

Ensuite il ne reste qu'à tester ce nouvel algorithme en essayant par exemple de rentrer pour B la valeur 0, ce qui donne :

ALGODOX : DIVISION D

AlgoBox

Entrer B :
(utiliser le . comme séparateur décimal)
(exemples de syntaxe possible : -3 ; 2,6 ; 4/3 ; 1+sqrt(3) ; ...)

0

OK Annuler

```

1  VARIABLES
2  A EST_DU_TYPE NOMBRE
3  B EST_DU_TYPE NOMBRE
4  Q EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6  AFFICHER "Pour le quotient des deux nombres, d
7  LIRE A
8  AFFICHER "Donner votre diviseur B"

```

Résultats

```

***Algorithme lancé***
Pour le quotient des deux nombres, donner votre dividende A
Donner votre diviseur B

```

Nouvelle amélioration : Fixer le nombres de décimales à afficher :

Ce problème devient un problème mathématique, c'est-à-dire de la réponse

3,257257257, il faut arriver à 3,26 par exemple où 3,25.

En fait, une solution simple consiste à utiliser la fonction **partie entière** en procédant de la manière suivante :

$3,257257257 \times 100$ devient 325,7257257, dont on prend la partie entière 325, et il suffit alors de diviser ce nouveau résultat par 100 pour obtenir 3,25.

La fonction partie entière, comme dans beaucoup de langages informatiques est donnée par l'instruction suivante :

$\text{floor}()$

Et notre ligne de calcul devient alors $Q = \text{floor}(A / B * 100) / 100$

Au niveau de l'algorithme, il suffit de se positionner sur la ligne Q PREND LA VALEUR A/B puis de cliquer sur le bouton MODIFIER LIGNE, puis remplacer alors A/B par la formule $\text{floor}(A / B * 100) / 100$, ce qui donne le nouvel algorithme ci-dessous :

Présentation de l'algorithme

Quotient de deux entiers.

Code de l'algorithme

```

▼ VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  Q EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  AFFICHER "Pour le quotient des deux nombres, donner votre dividende A"
  LIRE A
  AFFICHER "Donner votre diviseur B"
  TANT_QUE (B==0) FAIRE
    DEBUT_TANT_QUE
    LIRE B
    FIN_TANT_QUE
  Q PREND LA_VALEUR floor(A/B*100)/100
  AFFICHER "La valeur du quotient A/B est "
  AFFICHER Q
  FIN_ALGORITHME

```

Résultats

```

***Algorithme lancé***
Pour le quotient des deux nombres, donner votre dividende A
Donner votre diviseur B
La valeur du quotient A/B est
3.33
***Algorithme terminé***

```

Exercice 1 : Modifier l'algorithme pour obtenir un affichage à 3 décimales.

Il s'agit de rajouter une variable pour demander à l'utilisateur le nombre de décimales souhaité.

Note pour cet exercice, il faut utiliser la fonction puissance $\text{pow}(x, n)$ qui correspond à la puissance nième de x .

Solutions :

Présentation de l'algorithme

Affichage avec 3 décimales

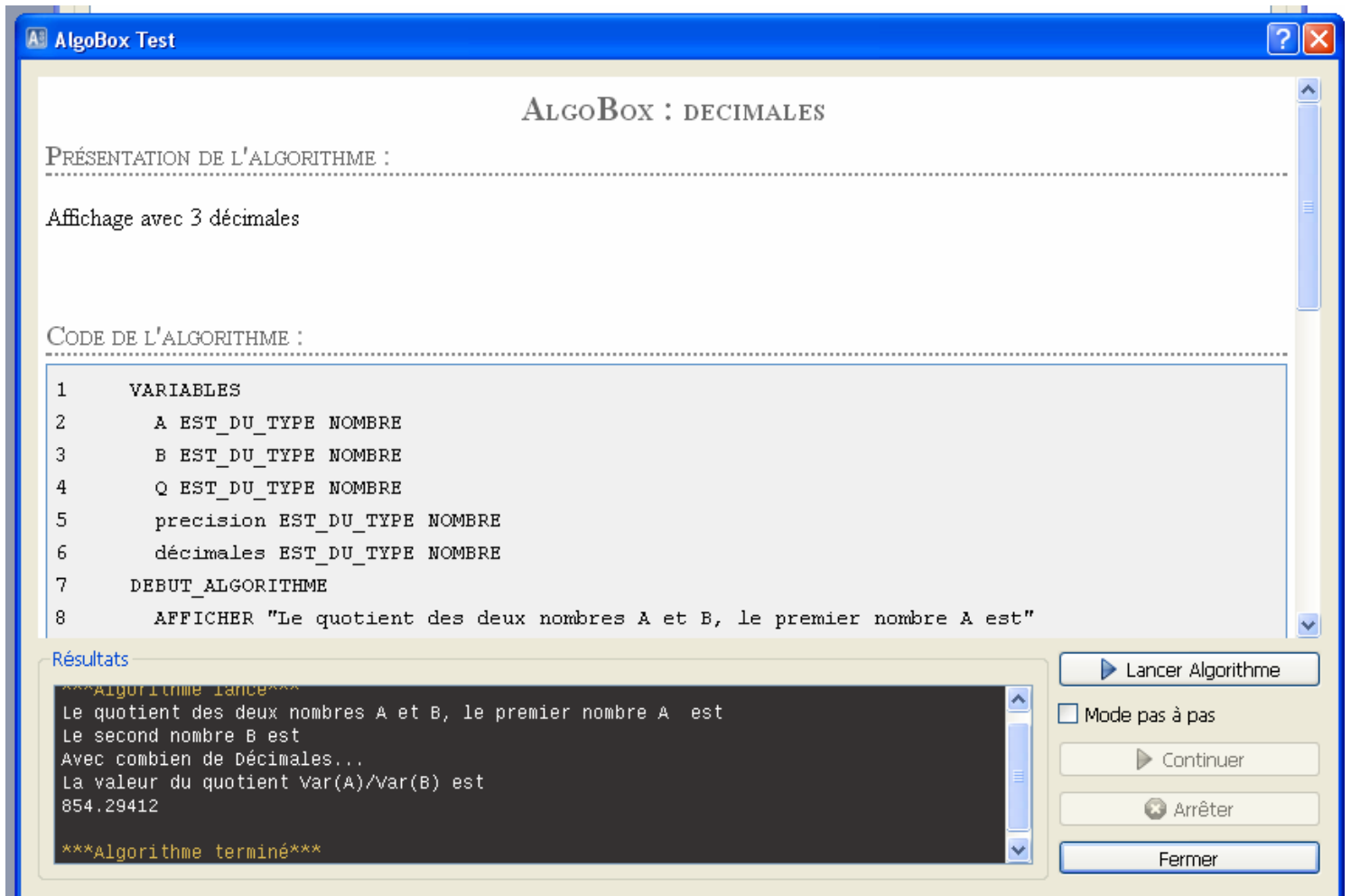
Code de l'algorithme

```

▼ VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  Q EST_DU_TYPE NOMBRE
  precision EST_DU_TYPE NOMBRE
  décimales EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  AFFICHER "Le quotient des deux nombres A et B, le premier nombre A est"
  LIRE A
  AFFICHER "Le second nombre B est"
  TANT_QUE (B==0) FAIRE
    DEBUT_TANT_QUE
    LIRE B
    FIN_TANT_QUE
  AFFICHER "Avec combien de Décimales..."
  LIRE décimales
  precision PREND_LA_VALEUR pow(10,décimales)
  Q PREND_LA_VALEUR floor(A/B*precision)/precision
  AFFICHER "La valeur du quotient Var(A)/Var(B) est"
  AFFICHER Q
  FIN_ALGORITHME

```

Ce qui donne après utilisation : en prenant par exemple $A = 14\,523$, puis $B = 17$ et enfin décimales = 11



Exercice 2 : La division à l'ancienne. Les premiers processeurs ne faisaient que des additions et des soustractions ! Comment alors peut-on réaliser une division ?

Solutions :

Voici la méthode directe employée pour faire la division avec un quotient entier de deux nombres entiers A et B. On soustrait B de a tant que c'est possible et on compte le nombre de soustractions faites, ce qui est assez simple :

Un exemple, on cherche le quotient entier de 11 par 3.

On calcule donc $11 - 3 = 8$, et on compte 1 (une soustraction).

Puis $8 - 3 = 5$ et on compte 2

Puis $5 - 3 = 2$ et on compte 3

Puis $2 - 3 = -1$ ce qui est impossible dans \mathbb{N} , donc on s'arrête de compter, et alors le quotient entier de 11 par 3 est donc 3.

Ci-dessous l'algorithme avec AlgoBox est :

Présentation de l'algorithme

Division à l'ancienne

Code de l'algorithme

```

▼ VARIABLES
  | A EST_DU_TYPE NOMBRE
  | B EST_DU_TYPE NOMBRE
  | Q EST_DU_TYPE NOMBRE
  | diff EST_DU_TYPE NOMBRE
  | R EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  | AFFICHER "Division entiere de A par B"
  | LIRE A
  | LIRE B
  | //Ne pas oublier de mettre le Quotient à zéro
  | Q PREND_LA_VALEUR 0
  | diff PREND_LA_VALEUR A-B
  | ▼ TANT_QUE (diff >= 0) FAIRE
  |   | DEBUT_TANT_QUE
  |   | //On incrémente le quotient de 1 à chaque passage
  |   | Q PREND_LA_VALEUR Q+1
  |   | //On va calculer la différence et la remettre dans diff
  |   | diff PREND_LA_VALEUR diff-B
  |   | //Et on incrémente le quotient de 1 à chaque passage
  |   | FIN_TANT_QUE
  |   | AFFICHER Q
  | FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

Division à l'ancienne

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  A EST_DU_TYPE NOMBRE
3  B EST_DU_TYPE NOMBRE
4  Q EST_DU_TYPE NOMBRE
5  diff EST_DU_TYPE NOMBRE
6  R EST_DU_TYPE NOMBRE
7  DEBUT_ALGORITHME
8  AFFICHER "Division entiere de A par B"

```

Résultats

```

***Algorithme lancé***
Division entiere de A par B
3
***Algorithme terminé***

```

Exercice 3 : Que se passe-t-il si les lignes Q PREND LA VALEUR Q+1 et diff PREND LA VALEUR diff – B sont inversées ? L’algorithme fonctionne –t-il encore correctement ?

Solutions :

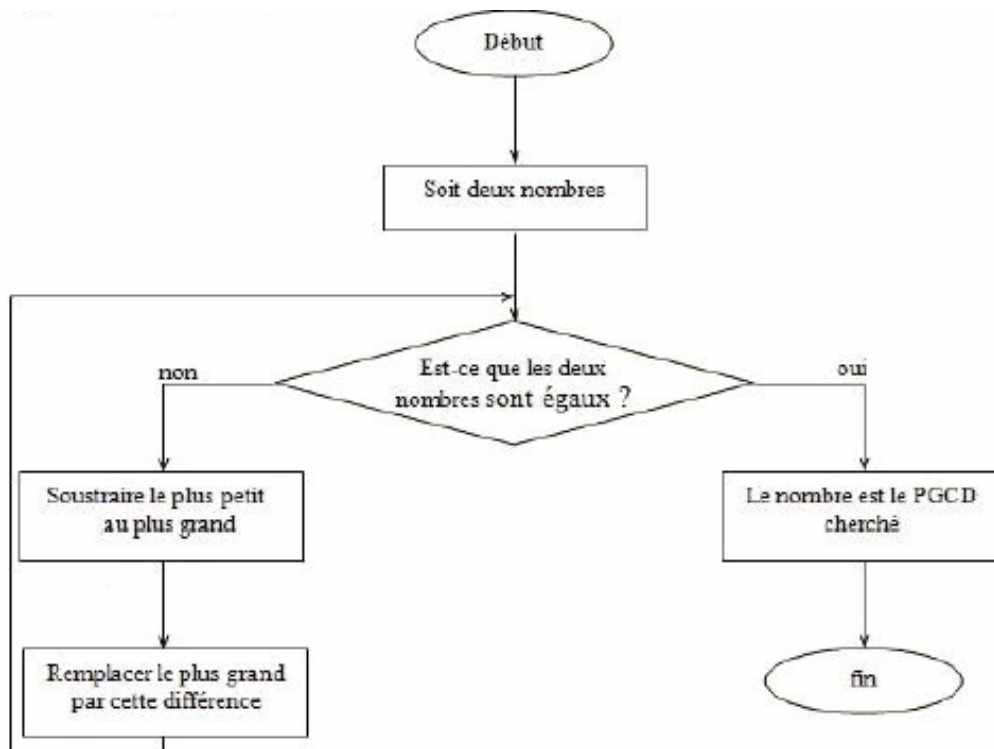
A faire dans le programme

Exercice 4 : Utiliser la variable R pour afficher le reste dans la division de A par B à la place du quotient Q.

II°) Le pgcd de deux nombres entiers.

Rappelons que le plus grand diviseur commun aux deux nombres entiers 12 et 18 est 6 car les diviseurs de 12 sont {1 ; 2 ; 3 ; 4 ; 6 ; 12} et les diviseurs de 18 sont {1 ; 2 ; 3 ; 6 ; 9 ; 18}, et les diviseurs communs aux deux entiers sont {1 ; 2 ; 3 ; 6}, et le plus grand entier est bien 6.

Voici ci-dessous un algorithme pour déterminer le PGCD de deux nombres entiers A et B : (Algorithmes des soustractions successives).



Ce qui donne avec Algobox :

Puis l'exécution de ce programme en prenant A=12 et B=18, donne bien comme PGCD 6.

Présentation de l'algorithme

PGCD

Code de l'algorithme

```

VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  diff EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE A
  LIRE B
  //noter la façon de tester la non égalité de deux valeurs
  TANT_QUE (A!=B) FAIRE
    DEBUT_TANT_QUE
      SI (A>B) ALORS
        DEBUT_SI
          A PREND_LA_VALEUR A-B
        FIN_SI
      SINON
        DEBUT_SINON
          B PREND_LA_VALEUR B-A
        FIN_SINON
    FIN_TANT_QUE
  AFFICHER A
FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

PGCD

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      A EST_DU_TYPE NOMBRE
3      B EST_DU_TYPE NOMBRE
4      diff EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6      LIRE A
7      LIRE B
8      //noter la façon de tester

```

Résultats

```

***Algorithme lancé***
6
***Algorithme terminé***

```

Comme exercice supplémentaire, on peut par exemple se servir de cet algorithme pour déterminer si deux entiers sont premiers entre eux. A titre de rappel, deux entiers naturels sont premiers entre eux si et seulement si leur PGCD vaut 1

III°) Travailler sur quatre semaines.

Dans certains emplois du temps il faut découper sur quatre semaines plutôt que sur deux. C'est le cas pour certains élèves de seconde en informatique dans un lycée quelconque : par exemple en semaine 41, nous sommes en semaine 1, en semaine 42 nous sommes en semaine 2, en semaine 43 nous sommes en semaine 3, en semaine 44 nous sommes en semaine 4, en semaine 45 nous sommes en semaine 1 et ainsi de suite.

On voudrait faire afficher, à partir du numéro de semaine du calendrier, le numéro de semaine de l'emploi du temps, c'est-à-dire 1, 2, 3 ou 4.

La méthode :

Prendre le numéro de semaine du calendrier dans la variable semaine, calculer le reste dans la division entière par 4 de cette semaine et le stocker dans semaine_info.

Si semaine_info vaut zéro le mettre à 4. Les variables à utiliser seront donc semaine, de type numérique, et semaine_info également de type numérique.

Et il faudra utiliser la fonction **A%B** qui donne le reste dans la division euclidienne de A par B, et l'algorithme est alors le suivant :

Présentation de l'algorithme

Travailler sur quatre semaines.

Code de l'algorithme

```

▼ VARIABLES
  semaine EST_DU_TYPE NOMBRE
  semaine_info EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  LIRE semaine
  //il s'agit maintenant de calculer le reste de ce nombre dans la division par 4
  semaine_info PREND_LA_VALEUR semaine%4
  //probleme avec le reste zéro qui est peu agréable comme numero de semaine
  ▼ SI (semaine_info==0) ALORS
    DEBUT_SI
    semaine_info PREND_LA_VALEUR 4
    FIN_SI
  AFFICHER semaine_info
FIN_ALGORITHME

```

1°) Exercice :

On sait que les cours d'informatique de la classe de seconde 12 sont en semaine 4, écrire un algorithme permettant d'afficher toutes les semaines 4 de l'année.

Il faudra donc utiliser une boucle pour explorer les 53 semaines de 1 à 53, et cette boucle peut avoir la structure suivante :

POUR semaine ALLANT DE 1 A 53

DEBUT POUR

-
-
-

FIN POUR

Solution de l'exercice 1 :

Présentation de l'algorithme

Afficher toutes les semaines 4.

Code de l'algorithme

```

VARIABLES
├── semaine EST_DU_TYPE NOMBRE
├── semaine_info EST_DU_TYPE NOMBRE
└── DEBUT_ALGORITHME
    ├── POUR semaine ALLANT_DE 1 A 53
    │   ├── DEBUT_POUR
    │   ├── semaine_info PREND_LA_VALEUR semaine%4
    │   └── SI (semaine_info==0) ALORS
    │       ├── DEBUT_SI
    │       ├── semaine_info PREND_LA_VALEUR 4
    │       ├── FIN_SI
    │       └── SI (semaine_info==4) ALORS
    │           ├── DEBUT_SI
    │           ├── AFFICHER "La semaine numero"
    │           ├── AFFICHER semaine
    │           ├── AFFICHER "est une semaine 4"
    │           ├── FIN_SI
    │           └── FIN_POUR
    └── FIN_ALGORITHME
  
```

PRÉSENTATION DE L'ALGORITHME :

Afficher toutes
les semaines 4.

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      semaine EST_DU_TYPE NOMBRE
3      semaine_info EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5      POUR semaine ALLANT_DE 1 A 53
6          DEBUT_POUR
7              semaine_info PREND_LA_VALEUR semaine%4
  
```

Résultats

```

La semaine numero
48
est une semaine 4
La semaine numero
52
est une semaine 4

***Algorithme terminé***
  
```

2°) Exercice :

La même problématique, mais cette fois pour une des quatre semaines au choix de l'utilisateur. On va donc créer une nouvelle variable et demander à l'utilisateur quelle liste de semaines il veut, soit 1, 2, 3 ou 4.

Solution de l'exercice 2 :

Présentation de l'algorithme

Afficher une des 4 semaines|

Code de l'algorithme

```

VARIABLES
├── semaine EST_DU_TYPE NOMBRE
├── semaine_info EST_DU_TYPE NOMBRE
├── numero EST_DU_TYPE NOMBRE
└── DEBUT_ALGORITHME
    ├── LIRE numero
    ├── POUR semaine ALLANT_DE 1 A 53
    │   ├── DEBUT_POUR
    │   ├── semaine_info PREND_LA_VALEUR semaine%4
    │   └── SI (semaine_info==0) ALORS
    │       ├── DEBUT_SI
    │       ├── semaine_info PREND_LA_VALEUR 4
    │       ├── FIN_SI
    │       └── SI (semaine_info==numero) ALORS
    │           ├── DEBUT_SI
    │           ├── AFFICHER "La semaine numero"
    │           ├── AFFICHER semaine
    │           ├── AFFICHER "est une semaine "
    │           ├── LIRE numero
    │           ├── FIN_SI
    │           └── FIN_POUR
    └── FIN_ALGORITHME
  
```

PRÉSENTATION DE L'ALGORITHME :

Afficher une des
4 semaines

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      semaine EST_DU_TYPE NOMBRE
3      semaine_info EST_DU_TYPE NOMBRE
4      numero EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6      LIRE numero
7      POUR semaine ALLANT_DE 1 A 53
  
```

Résultats

```

La semaine numero
10
est une semaine
La semaine numero
13
est une semaine

***Algorithme terminé***
  
```

IV°) Les chaînes de caractères : les palindromes.

Le travail avec les variables alphanumériques dites CHAINES DE CARACTERES est toujours assez difficile en programmation. Les mots, les phrases, les structures n'obéissant pas du tout à des règles mathématiques mais étant bourrée d'exceptions.

Ici, il s'agit de créer un algorithme qui renverse un mot ou une expression pour vérifier s'il s'agit d'un palindrome, comme LAVAL, ERDRE, RADAR)

Méthode :

Un mot est saisi dans la variable CHAINE Palind.

La variable CHAINE Résult va récupérer une par une les lettres de palind à partir de la dernière etc...

Il est utile pour cet algorithme de connaître les fonctions de manipulation de chaînes suivante :

palind.length : renvoi la longueur de la chaîne (par exemple 5 si on tape LAVAL)

palind.substr(3,1) : renvoi la chaîne de longueur 1 située en 3° position (par exemple V pour LAVAL)

Attention :

En déclarant les variables il faut évidemment choisir le type chaîne dans AlgoBox.

Ce qui donne l'algorithme suivant :

```

Présentation de l'algorithme
Palindrome

Code de l'algorithme
▼ VARIABLES
  x EST_DU_TYPE NOMBRE
  palind EST_DU_TYPE CHAINE
  result EST_DU_TYPE CHAINE
  longueur EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  //Vidons la chaîne result
  result PREND_LA_VALEUR ""
  LIRE palind
  //on a besoin de la longueur de la chaîne palind
  longueur PREND_LA_VALEUR palind.length
  POUR x ALLANT_DE 1 A longueur+1
  DEBUT_POUR
    //le dernier caractère est mis en premier dans result puis l'avant dernier en second etc
    result PREND_LA_VALEUR result+palind.substr(longueur-x+1,1)
  FIN_POUR
  AFFICHER result
FIN_ALGORITHME

```

L'exécution de cet algorithme avec LAVAL, RADAR, puis JEAN donne :

```

Résultats
***Algorithme lancé***
LAVAL
***Algorithme terminé***

```

```

Résultats
***Algorithme lancé***
RADAR
***Algorithme terminé***

```

```

Résultats
***Algorithme lancé***
NAEJ
***Algorithme terminé***

```

Sous forme d'exercice, essayer de compléter cet algorithme en ajoutant un test SI ALORS pour afficher un message de réussite si le mot tape est bien un palindrome ou non.

Ce qui donne

Présentation de l'algorithme

Palindrome

Code de l'algorithme

```

▼ VARIABLES
  x EST_DU_TYPE NOMBRE
  palind EST_DU_TYPE CHAINE
  result EST_DU_TYPE CHAINE
  longueur EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  //Vidons la chaine result
  result PREND_LA_VALEUR ""
  LIRE palind
  //on a besoin de la longueur de la chaine palind
  longueur PREND_LA_VALEUR palind.length
  ▼ POUR x ALLANT_DE 1 A longueur+1
    DEBUT_POUR
    //le dernier caractere est mis en premier dans result puis l'avant dernier en second etc
    result PREND_LA_VALEUR result+palind.substr(longueur-x+1,1)
    FIN_POUR
  AFFICHER result
  ▼ SI (result==palind) ALORS
    DEBUT_SI
    AFFICHER "BRAVO C'EST UN PALINDROME"
    FIN_SI
  ▼ SINON
    DEBUT_SINON
    AFFICHER "NO, CE N'EST PAS UN PALINDROME"
    FIN_SINON
  FIN_ALGORITHME

```

L'exécution de cet algorithme avec RADAR, puis JEAN donne :

Résultats

```

***Algorithme lancé***
RADAR
BRAVO C'EST UN PALINDROME

***Algorithme terminé***

```

Résultats

```

***Algorithme lancé***
NAEJ
NO, CE N'EST PAS UN PALINDROME

***Algorithme terminé***

```

V°) Le jeu du c'est plus.....c'est moins.

Classique jeu du c'est plus c'est moins avec un nombre entre 1 et 100 choisi au hasard par la machine.

Pour calculer un nombre au hasard entrée 1 et 100 on utilise la fonction **random()** qui renvoie une valeur décimale entre 0 et 0,9999999 !

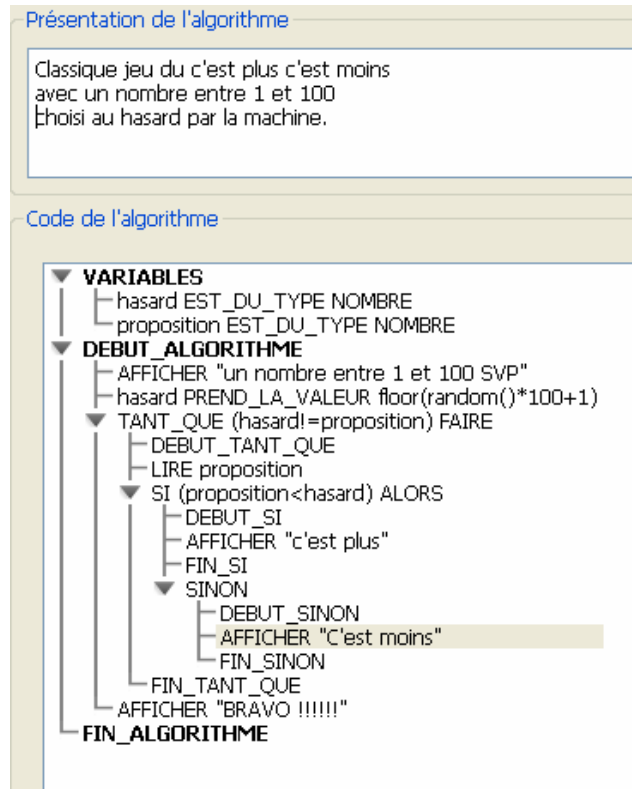
La formule classique pour obtenir un ombre entre 1 et 100 sera la suivante dans Albox : **floor(random()*100+1)**

Explication :

random() sort un nombre décimal entre 0 et 0,9999999.

random()*100 sort un nombre décimal entre 0 et 99,99999.

random()*100 + 1 sort un nombre décimal entre 1 et 100,99999, dont il faut prendre la partie entière en utilisant l'instruction floor(), ce qui donne l'algorithme suivant :

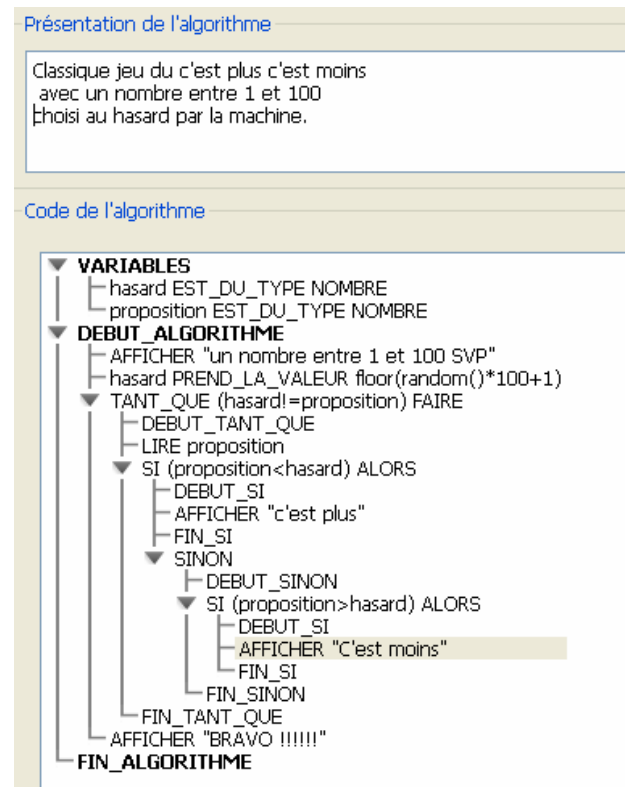


Prolongement :

Nous avons la un problème algorithmique très intéressant car nous utilisons une instruction TANT QUE dans laquelle nous avons un test (proposition !=hasard) en début de boucle alors qu'il aurait fallu ce test en fin de boucle pour ne pas avoir a faire un passage inutile !

Si bien que nous nous trouvons souvent avec un « c'est moins en trop ».

Nous devons donc ajouter un test dans le SINON pour empêcher l'affichage du « c'est moins » en cas d'égalité, ce qui nous donne l'algorithme suivant :



Autre prolongement :

Problème qui aurait put être solutionné aussi avec une variable SORTIE qui prend la valeur 1 en cas d'égalité comme dans l'algorithme ci-contre. Lorsque vous modifiez votre algorithme vous pouvez aussi utiliser les raccourcis clavier ci-après pour les lignes ou les blocs

[Ctrl] [C] pour copier

[Ctrl] [X] pour couper

[Ctrl] [V] pour coller

N'oubliez pas dans les recherches ci-dessous d'utiliser lors de l'exécution des algorithmes dans Algobox la fonction PAS A PAS qui vous permet de surveiller la valeur de vos variables à chaque passage d'une boucle.

Présentation de l'algorithme

Classique jeu du c'est plus
c'est moins avec un nombre
entre 1 et 100 choisi au hasard
par la machine.

Code de l'algorithme

```

VARIABLES
├─ hasard EST_DU_TYPE NOMBRE
├─ proposition EST_DU_TYPE NOMBRE
└─ sortie EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
├─ AFFICHER "un nombre entre 1 et 100 SVP"
├─ sortie PREND_LA_VALEUR 0
├─ hasard PREND_LA_VALEUR floor(random()*100+1)
├─ TANT_QUE (sortie==0) FAIRE
│   ├── DEBUT_TANT_QUE
│   ├── LIRE proposition
│   ├── SI (proposition<hasard) ALORS
│   │   ├── DEBUT_SI
│   │   ├── AFFICHER "c'est plus"
│   │   └─ FIN_SI
│   ├── SINON
│   │   ├── DEBUT_SINON
│   │   ├── SI (proposition>hasard) ALORS
│   │   │   ├── DEBUT_SI
│   │   │   ├── AFFICHER "c'est moins"
│   │   │   └─ FIN_SI
│   │   └─ SINON
│   │       ├── DEBUT_SINON
│   │       ├── sortie PREND_LA_VALEUR 1
│   │       └─ FIN_SINON
│   └─ FIN_SINON
└─ FIN_TANT_QUE
AFFICHER "BRAVO !!!!!!"
FIN_ALGORITHME

```

EXERCICE 1 :

En dernière ligne proposer à l'utilisateur le nombre de coups qui ont été nécessaires avec un affichage du genre : BRAVO, vous avez trouvé en 7 coups.

CORRECTION DE L'EXERCICE 1 :

Présentation de l'algorithme

Classique jeu du c'est plus
c'est moins avec un nombre
entre 1 et 100 choisi au hasard
par la machine.

Code de l'algorithme

```

▼ VARIABLES
  |— hasard EST_DU_TYPE NOMBRE
  |— proposition EST_DU_TYPE NOMBRE
  |— compt EST_DU_TYPE NOMBRE
  |— min EST_DU_TYPE NOMBRE
  |— max EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  |— compt PREND_LA_VALEUR 0
  |— min PREND_LA_VALEUR 0
  |— max PREND_LA_VALEUR 100
  |— AFFICHER "un nombre entre "
  |— AFFICHER min
  |— AFFICHER " et "
  |— AFFICHER max
  |— AFFICHER " VSP "
  |— hasard PREND_LA_VALEUR floor(random()*100+1)
  |— proposition PREND_LA_VALEUR 0
  |— TANT_QUE (proposition!=hasard) FAIRE
    |— DEBUT_TANT_QUE
    |— compt PREND_LA_VALEUR compt+1
    |— LIRE proposition
    |— SI (proposition<hasard) ALORS
      |— DEBUT_SI
      |— min PREND_LA_VALEUR proposition
      |— AFFICHER "c'est plus , c'est entre "
      |— AFFICHER min
      |— AFFICHER " et "
      |— AFFICHER max
      |— FIN_SI
    |— SINON
      |— DEBUT_SINON
      |— max PREND_LA_VALEUR proposition
      |— AFFICHER "c'est moins , c'est entre "
      |— AFFICHER min
      |— AFFICHER " et "
      |— AFFICHER max
      |— FIN_SINON
    |— FIN_TANT_QUE
  |— AFFICHER "BRAVO ! vous avez réussi en "
  |— AFFICHER compt
  |— AFFICHER " coups."
FIN_ALGORITHME

```

EXERCICE 2 :

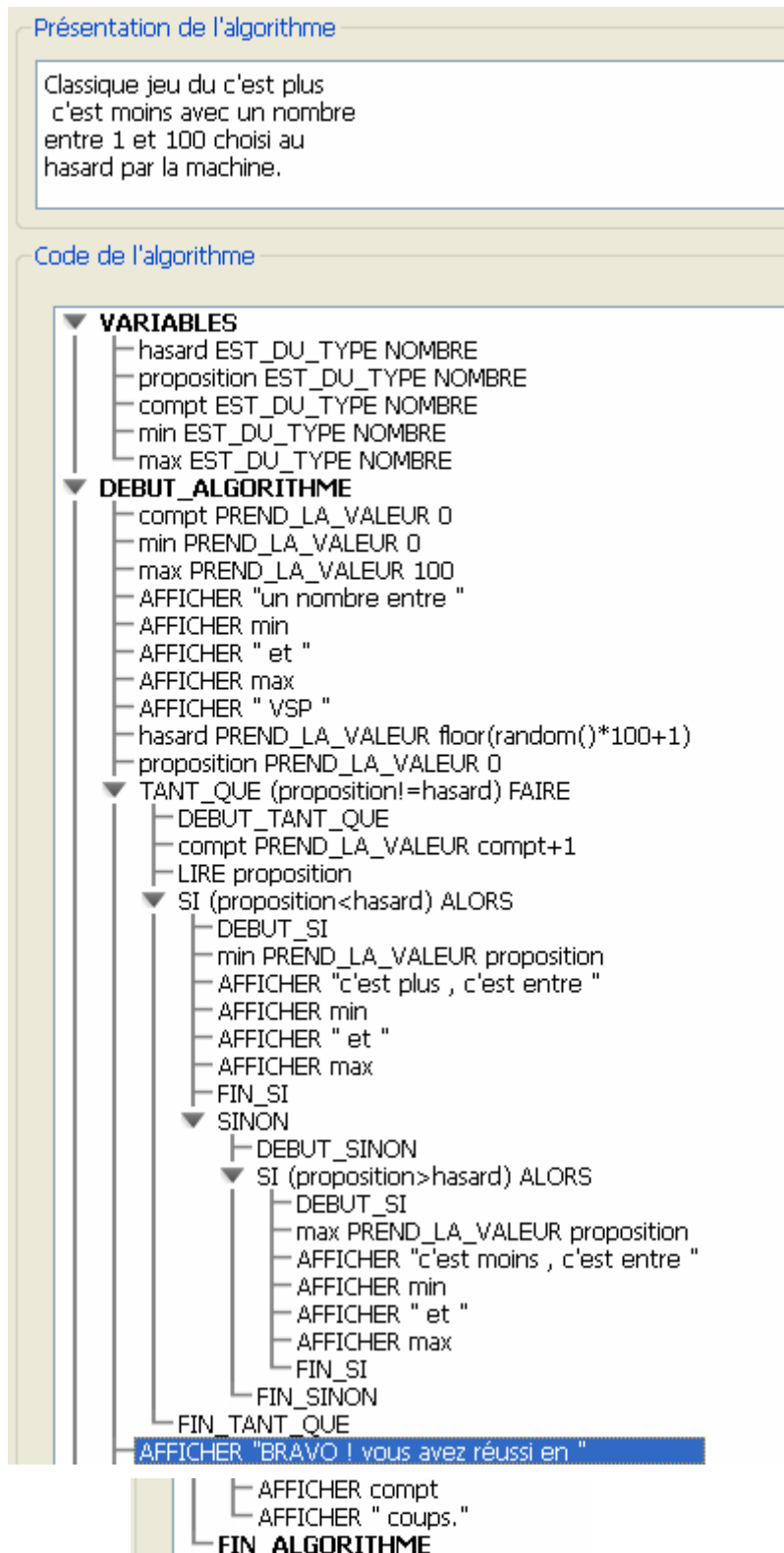
Afficher l'intervalle dans lequel se trouve le nombre pour faciliter la tâche à l'utilisateur !

Par exemple : après avoir proposé le nombre 50 l'utilisateur verra s'afficher un message du genre :

C'est moins, le nombre est situé entre 0 et 50.

Puis après avoir proposé 20, il aura un message du genre :

C'est plus, le nombre est situé entre 20 et 50 etc.....

CORRECTION DE L'EXERCICE 2 :

VI° Dans la foulée, la dichotomie.

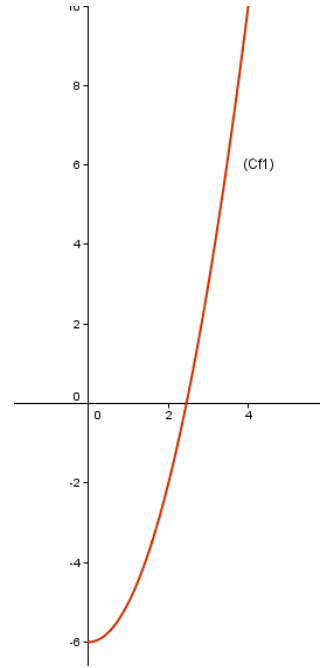
Il s'agit de calculer la valeur (souvent approchée) qui annule une fonction donnée $f_1(x)$.

La méthode est simple : on se place sur un intervalle de continuité monotone et on calcule l'image du milieu de cet intervalle par cette fonction.

En fonction du signe de cette image ce milieu remplace une des deux bornes et ainsi de suite.

Exemple :

On considère la fonction $f_1(x) = x^2 - 6$, et sur le graphique ci-contre sur l'intervalle $[0; 4]$. Nous remarquons que le milieu est 2 et que $f_1(2) = -2 < 0$, donc notre intervalle de travail va devenir $[2; 4]$, puis de même si on remarque que $f_1(3) = 3 > 0$, notre intervalle de travail devient donc $[2; 3]$, etc



L'algorithme :

Attention, il faut définir une fonction en utilisant l'onglet UTILISER UNE FONCTION NUMERIQUE. Cette fonction aura pour nom prédéfini F1(x).

Dans l'exemple, ci vous désirez utiliser la fonction ci-dessus $f_1(x) = x^2 - 6$, il faut taper **pow(x,2)-6**, ou bien tout simplement **x*x-6**.

Vous pouvez aussi utiliser l'onglet DESSINER DANS UN REPERE pour voir le parcours fait par le point de recherche F1(middle).

Ce qui donne l'algorithme suivant :

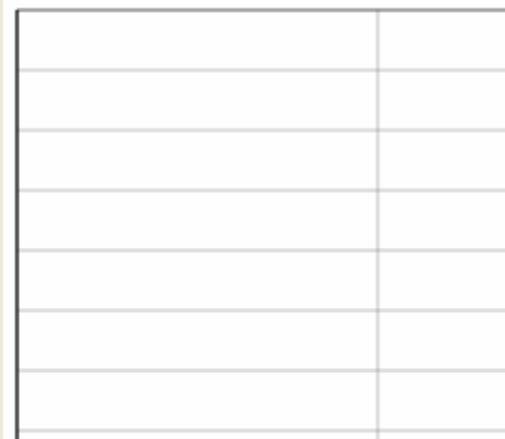
```

Présentation de l'algorithme
Dichotomie

Code de l'algorithme
VARIABLES
  a EST_DU_TYPE NOMBRE
  min EST_DU_TYPE NOMBRE
  max EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
  middle EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  min PREND_LA_VALEUR 0
  max PREND_LA_VALEUR 4
  a PREND_LA_VALEUR min
  b PREND_LA_VALEUR max
  //on défini une précision de 0.01
  TANT_QUE (abs(a-b)>0.01) FAIRE
    DEBUT_TANT_QUE
      //on met dans middle le point central de l'intervalle
      middle PREND_LA_VALEUR (b+a)/2
      TRACER_POINT (middle,F1(middle))
      //si l'image de middle c'est middle qui devient la borne supérieure de l'intervalle
      //sinon middle devient la borne inférieure de l'intervalle
      SI (F1(middle)>0) ALORS
        DEBUT_SI
          b PREND_LA_VALEUR middle
        FIN_SI
      SINON
        DEBUT_SINON
          a PREND_LA_VALEUR middle
        FIN_SINON
      FIN_TANT_QUE
    AFFICHER a
    AFFICHER " <solution< "
    AFFICHER b
  FIN_ALGORITHME

```

GRAPHIQUE :



Résultats

```

***Algorithme lancé***
2.4453125 <solution< 2.453125
***Algorithme terminé***

```

A titre d'exercice supplémentaire, vous pouvez demander à l'utilisateur les deux valeurs de min et de max, mais il n'est pas possible, avec ce programme de demander à l'utilisateur de définir lui-même la fonction F1(x).

VII°) Moyenne, écart-type, tri et médiane.

Pour calculer la moyenne d'un certain nombre de notes (**combien** par exemple) nous devons utiliser une liste (mathématiquement on dirait plutôt un vecteur), note[1] contient la première note, puis note[2] qui contient la seconde et ainsi de suite.

Nous pouvons ainsi saisir les notes au sein d'une boucle de longueur **combien**.

Il ne reste qu'à faire la somme des notes et diviser par **combien**.

Ce qui donne l'algorithme suivant :

Présentation de l'algorithme

Moyenne de notes|

Code de l'algorithme

```

▼ VARIABLES
  | note EST_DU_TYPE LISTE
  | i EST_DU_TYPE NOMBRE
  | combien EST_DU_TYPE NOMBRE
  | sigma EST_DU_TYPE NOMBRE
  | moyenne EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  | sigma PREND_LA_VALEUR 0
  | LIRE combien
  | //une boucle pour demander toutes les notes
  | ▼ POUR i ALLANT_DE 1 A combien
  |   | DEBUT_POUR
  |   | LIRE note[i]
  |   | FIN_POUR
  |   | //calculons la moyenne
  |   | ▼ POUR i ALLANT_DE 1 A combien
  |   |   | DEBUT_POUR
  |   |   | sigma PREND_LA_VALEUR sigma+note[i]
  |   |   | FIN_POUR
  |   | moyenne PREND_LA_VALEUR sigma/combien
  |   | AFFICHER "la moyenne des notes est de "
  |   | AFFICHER moyenne
  | FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

.....

Moyenne de notes

CODE DE L'ALGORITHME :

.....

```

1  VARIABLES
2  note EST_DU_TYPE LISTE
3  i EST_DU_TYPE NOMBRE
4  combien EST_DU_TYPE NOMBRE
5  sigma EST_DU_TYPE NOMBRE
6  moyenne EST_DU_TYPE NOMBRE
7  DEBUT_ALGORITHME
8  sigma PREND_LA_VALEUR 0

```

Résultats

```

***Algorithme lancé***
la moyenne des notes est de 10.75
***Algorithme terminé***

```

Exercice : faire afficher la moyenne, la variance et l'écart type de la série de notes :

Solution :

Utilisation, en prenant 10 notes : soit {12;11;8;15;12;14;17;2;6;13} , on obtient le résultat ci-dessous :

Présentation de l'algorithme

Moyenne , variance
et écart type

Code de l'algorithme

```

VARIABLES
├─ note EST_DU_TYPE LISTE
├─ i EST_DU_TYPE NOMBRE
├─ combien EST_DU_TYPE NOMBRE
├─ sigma EST_DU_TYPE NOMBRE
├─ moyenne EST_DU_TYPE NOMBRE
├─ variance EST_DU_TYPE NOMBRE
└─ ecart_type EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
├─ sigma PREND_LA_VALEUR 0
├─ LIRE combien
├─ //une boucle pour demander toutes les notes
├─ POUR i ALLANT_DE 1 A combien
│   ├── DEBUT_POUR
│   ├── LIRE note[i]
│   └─ FIN_POUR
├─ //calculons la moyenne
├─ POUR i ALLANT_DE 1 A combien
│   ├── DEBUT_POUR
│   ├── sigma PREND_LA_VALEUR sigma+note[i]
│   └─ FIN_POUR
├─ moyenne PREND_LA_VALEUR floor(sigma/combien*100)/100
├─ AFFICHER "la moyenne des notes est de "
├─ AFFICHER moyenne
├─ //calculons la variance
├─ sigma PREND_LA_VALEUR 0
├─ POUR i ALLANT_DE 1 A combien
│   ├── DEBUT_POUR
│   ├── sigma PREND_LA_VALEUR sigma+pow(note[i],2)/combien
│   └─ FIN_POUR
├─ variance PREND_LA_VALEUR sigma-pow(moyenne,2)
├─ AFFICHER "la Variance est "
├─ AFFICHER variance
├─ //calculons l'écart type
├─ ecart_type PREND_LA_VALEUR sqrt(variance)
├─ AFFICHER "l'écart type est donc de "
└─ AFFICHER ecart_type
FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

Moyenne , variance
et écart type

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      note EST_DU_TYPE LISTE
3      i EST_DU_TYPE NOMBRE
4      combien EST_DU_TYPE NOMBRE
5      sigma EST_DU_TYPE NOMBRE
6      moyenne EST_DU_TYPE NOMBRE
7      variance EST_DU_TYPE NOMBRE

```

Résultats

```

***Algorithme lancé***
la moyenne des notes est de 11
la variance est 18.2
l'écart type est donc de 4.2661458
***Algorithme terminé***

```

Prolongement :

Détermination de la médiane de la série de notes, et pour déterminer cette médiane, il faut d'abord trier dans l'ordre croissant la série, qui est une opération pas évidente au sein d'un algorithme. Il existe de nombreuses façons de réaliser un tri. Nous allons choisir la plus simple mais peut être pas la plus rapide au niveau de très grandes séries statistiques.

Réalisons un tri dans l'ordre croissant de la série de notes ci-dessus.

Le principe est le suivant : on fait parcourir la série et dès que l'on trouve deux notes consécutives qui ne sont pas dans l'ordre on les inverse et on refait le parcours de la série.

La variable PERMUTE est mise à zéro puis on fait le parcours de la série de 1 à combien – 1.

Si on trouve $note[i] < note[i+1]$ dans la série on met la variable PERMUTE à 1 et on inverse les deux notes.

Quand la variable PERMUTE restera à zéro, il n'y aura plus rien à inverser et donc la série sera bien triée dans l'ordre croissant.

On utilise un repère pour tracer l'histogramme de la série. Il faut pour cela cliquer sur DESSINER DANS UN REPERE pour pouvoir utiliser la fonction TRACER_SEGMENT de la couleur choisie et après avoir définis les intervalle X et y du repère.

A titre d'exercice supplémentaire, faire un tri décroissant de la série de notes.

Une fois la série triée il ne reste plus qu'à calculer la médiane. Pour éviter d'avoir à tester la parité du nombre de notes on utilise une petite astuce :

On calcule le centre de la série et on fait la moyenne des valeurs entourant ce centre. Si la série est impaire ces deux nombres seront les mêmes sinon ils entoureront le centre.

Nous calculons $med1 = floor((1 + combien) / 2)$ puis $med2 = floor((1 + combien) / 2 + 0,5)$, le milieu sera $(med1 + med2) / 2$, ($med1 = med2$ si on a un nombre impair) et la médiane sera $(note[med1] + note[med2]) / 2$.

Et il ne reste qu'à tracer la médiane après avoir tracer la série.

```
Code de l'algorithme
VARIABLES
- note EST_DU_TYPE LISTE
- i EST_DU_TYPE NOMBRE
- combien EST_DU_TYPE NOMBRE
- tampon EST_DU_TYPE NOMBRE
- permute EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
- sigma PREND_LA_VALEUR 0
- LIRE combien
- POUR i ALLANT_DE 1 A combien
  - DEBUT_POUR
  - LIRE note[i]
  - FIN_POUR
- permute PREND_LA_VALEUR 1
- TANT_QUE (permute==1) FAIRE
  - DEBUT_TANT_QUE
  - i PREND_LA_VALEUR 1
  - permute PREND_LA_VALEUR 0
  - TANT_QUE ((i<combien) ET (permute==0)) FAIRE
    - DEBUT_TANT_QUE
    - SI (note[i]>note[i+1]) ALORS
      - DEBUT_SI
      - permute PREND_LA_VALEUR 1
      - tampon PREND_LA_VALEUR note[i]
      - note[i] PREND_LA_VALEUR note[i+1]
      - note[i+1] PREND_LA_VALEUR tampon
      - FIN_SI
    - i PREND_LA_VALEUR i+1
    - FIN_TANT_QUE
  - FIN_TANT_QUE
- POUR i ALLANT_DE 1 A combien
  - DEBUT_POUR
  - sigma PREND_LA_VALEUR sigma+note[i]
  - TRACER_SEGMENT (i,0)->(i,note[i])
  - FIN_POUR
FIN_ALGORITHME
```

[Solution du tracer de la médiane de la série de notes :](#)

Présentation de l'algorithme

Médiane de la
série de notes.

Code de l'algorithme

```

▼ VARIABLES
  | note EST_DU_TYPE LISTE
  | i EST_DU_TYPE NOMBRE
  | combien EST_DU_TYPE NOMBRE
  | tampon EST_DU_TYPE NOMBRE
  | permute EST_DU_TYPE NOMBRE
  | med1 EST_DU_TYPE NOMBRE
  | med2 EST_DU_TYPE NOMBRE
  | mediane EST_DU_TYPE NOMBRE
  | milieu EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  | sigma PREND_LA_VALEUR 0
  | LIRE combien
  ▼ POUR i ALLANT_DE 1 A combien
    | DEBUT_POUR
    | LIRE note[i]
    | FIN_POUR
  | permute PREND_LA_VALEUR 1
  ▼ TANT_QUE (permute==1) FAIRE
    | DEBUT_TANT_QUE
    | i PREND_LA_VALEUR 1
    | permute PREND_LA_VALEUR 0
    ▼ TANT_QUE ((i<combien) ET (permute=0)) FAIRE
      | DEBUT_TANT_QUE
      ▼ SI (note[i]>note[i+1]) ALORS
        | DEBUT_SI
        | permute PREND_LA_VALEUR 1
        | tampon PREND_LA_VALEUR note[i]
        | note[i] PREND_LA_VALEUR note[i+1]
        | note[i+1] PREND_LA_VALEUR tampon
        | FIN_SI
      | i PREND_LA_VALEUR i+1
      | FIN_TANT_QUE
    | FIN_TANT_QUE
  ▼ POUR i ALLANT_DE 1 A combien
    | DEBUT_POUR
    | sigma PREND_LA_VALEUR sigma+note[i]
    | TRACER_SEGMENT (i,0)->(i,note[i])
    | FIN_POUR

  | med1 PREND_LA_VALEUR floor((1+combien)/2)
  | med2 PREND_LA_VALEUR floor((1+combien)/2+0.5)
  | milieu PREND_LA_VALEUR (med1+med2)/2
  | mediane PREND_LA_VALEUR (note[med1]+note[med2])/2
  | TRACER_SEGMENT (milieu,0)->(milieu,mediane)
FIN_ALGORITHME

```

VIII°) Un peu de probabilités : la somme de deux dés.

Nous savons que la probabilité d'apparition de la somme 7 lors du jet de deux dés de couleurs différentes est de $1/6$ c'est-à-dire à peu près 17 %.

Nous pouvons utiliser un algorithme pour vérifier si ce nombre théorique se retrouve sur 1000 lancers aléatoires de la machine. Utilisons comme nous l'avons déjà fait la fonction RANDOM() qui génère un décimal entre 0 et 0,9999999.

Fabriquons la formule :

RANDOM()*6 donne un nombre entre 0 et 5,999999

RANDOM()*6+1 est donc entre 1 et 6,999999

Et il suffit d'utiliser la fonction partie entière floor() pour achever la formule, soit FLOOR(RANDOM()*6+1).

Ce qui donne l'algorithme suivant :

Présentation de l'algorithme

La somme de deux dés.

Code de l'algorithme

```

VARIABLES
  dés_rouge EST_DU_TYPE NOMBRE
  dés_bleu EST_DU_TYPE NOMBRE
  les_deux EST_DU_TYPE NOMBRE
  n EST_DU_TYPE NOMBRE
  compteur EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  compteur PREND_LA_VALEUR 0
  AFFICHER "Somme de deux dés de couleurs différentes"
  POUR n ALLANT_DE 1 A 1000
    DEBUT_POUR
      dés_rouge PREND_LA_VALEUR floor(random()*6+1)
      dés_bleu PREND_LA_VALEUR floor(random()*6+1)
      les_deux PREND_LA_VALEUR dés_rouge+dés_bleu
      SI (les_deux==7) ALORS
        DEBUT_SI
          compteur PREND_LA_VALEUR compteur+1
        FIN_SI
    FIN_POUR
  AFFICHER "la somme 7 a été trouvée "
  AFFICHER compteur
  AFFICHER " fois sur 1000 lancers"
FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

La somme de deux dés.

CODE DE L'ALGORITHME :

```

1  VARIABLES
2    dés_rouge EST_DU_TYPE NOMBRE
3    dés_bleu EST_DU_TYPE NOMBRE
4    les_deux EST_DU_TYPE NOMBRE
5    n EST_DU_TYPE NOMBRE
6    compteur EST_DU_TYPE NOMBRE
7  DEBUT_ALGORITHME
8    compteur PREND_LA_VALEUR 0

```

Résultats

```

***Algorithme lancé***
Somme de deux dés de couleurs différentes
la somme 7 a été trouvée 158 fois sur 1000 lancers
***Algorithme terminé***

```

Quelques exercices supplémentaires.

EXERCICE 1 :

Demander à l'utilisateur quelle somme il veut tester, entre 2 et 12 bien sur, et sur combien de lancers il veut faire ce test.

CORRECTION DE L'EXERCICE 1 :

Présentation de l'algorithme

Quelle somme de deux dés. |

Code de l'algorithme

```

▼ VARIABLES
  - des1 EST_DU_TYPE NOMBRE
  - des2 EST_DU_TYPE NOMBRE
  - sigma EST_DU_TYPE NOMBRE
  - combien EST_DU_TYPE NOMBRE
  - i EST_DU_TYPE NOMBRE
  - compteur EST_DU_TYPE NOMBRE
  - quelle_somme EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  - LIRE combien
  - LIRE quelle_somme
  - //bien penser à mettre le compteur à zéro
  - compteur PREND_LA_VALEUR 0
  - //on réalise combien fois le tirage des deux dés
  - POUR i ALLANT_DE 1 A combien
    - DEBUT_POUR
    - des1 PREND_LA_VALEUR floor(random()*6+1)
    - des2 PREND_LA_VALEUR floor(random()*6+1)
    - sigma PREND_LA_VALEUR des1+des2
    - //on incrémente compteur uniquement si la somme des dés est 7
    - SI (sigma==quelle_somme) ALORS
      - DEBUT_SI
      - compteur PREND_LA_VALEUR compteur+1
      - FIN_SI
    - FIN_POUR
  - //on réalise l'affichage des résultats
  - AFFICHER "on a trouvé la somme "
  - AFFICHER quelle_somme
  - AFFICHER " "
  - AFFICHER compteur
  - AFFICHER " fois sur "
  - AFFICHER combien
FIN_ALGORITHME

```

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      des1 EST_DU_TYPE NOMBRE
3      des2 EST_DU_TYPE NOMBRE
4      sigma EST_DU_TYPE NOMBRE
5      combien EST_DU_TYPE NOMBRE
6      i EST_DU_TYPE NOMBRE
7      compteur EST_DU_TYPE NOMBRE
8      quelle_somme EST_DU_TYPE NOMBRE

```

Résultats

```

***Algorithme lancé***
on a trouvé la somme 9  0 fois sur 4
***Algorithme terminé***

```

EXERCICE 2 :

Demander le nombre de lancers désiré et faire afficher toutes les sommes de 2 à 12, pour cela il faudra utiliser évidemment une liste.

CORRECTION DE L'EXERCICE 2 :

Présentation de l'algorithme

Nombre de lancers|

Code de l'algorithme

```

▼ VARIABLES
  des1 EST_DU_TYPE NOMBRE
  des2 EST_DU_TYPE NOMBRE
  sigma EST_DU_TYPE NOMBRE
  somme EST_DU_TYPE NOMBRE
  combien EST_DU_TYPE NOMBRE
  apparition EST_DU_TYPE LISTE
▼ DEBUT_ALGORITHME
  LIRE combien
  ▼ POUR somme ALLANT_DE 2 A 12
    DEBUT_POUR
    apparition[somme] PREND_LA_VALEUR 0
    ▼ POUR i ALLANT_DE 1 A combien
      DEBUT_POUR
      des1 PREND_LA_VALEUR floor(random()*6+1)
      des2 PREND_LA_VALEUR floor(random()*6+1)
      sigma PREND_LA_VALEUR des1+des2
      ▼ SI (sigma==somme) ALORS
        DEBUT_SI
        apparition[somme] PREND_LA_VALEUR apparition[somme]+1
        FIN_SI
      FIN_POUR
    FIN_POUR
  ▼ POUR somme ALLANT_DE 2 A 12
    DEBUT_POUR
    AFFICHER "on a trouvé la somme "
    AFFICHER somme
    AFFICHER " "
    AFFICHER apparition[somme]
    AFFICHER " fois sur "
    AFFICHER combien
    FIN_POUR
  FIN_ALGORITHME

```

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  des1 EST_DU_TYPE NOMBRE
3  des2 EST_DU_TYPE NOMBRE
4  sigma EST_DU_TYPE NOMBRE
5  somme EST_DU_TYPE NOMBRE
6  combien EST_DU_TYPE NOMBRE
7  apparition EST_DU_TYPE LISTE
8  DEBUT_ALGORITHME

```

Résultats

```

on a trouvé la somme 7  2 fois sur 4
on a trouvé la somme 8  0 fois sur 4
on a trouvé la somme 9  2 fois sur 4
on a trouvé la somme 10 0 fois sur 4
on a trouvé la somme 11 0 fois sur 4
on a trouvé la somme 12 0 fois sur 4

***Algorithme terminé***

```

EXERCICE 3 :

Demander le nombre de lancers désiré et faire afficher toutes les sommes de 2 à 12, pour cela il faudra utiliser évidemment une liste.

CORRECTION DE L'EXERCICE 3 :

En utilisant la partie DESSINER DANS UN REPERE vous pouvez, à partir de l'algorithme ci-dessus, faire un histogramme des pourcentages.

Il est intéressant d'étudier la régularité et la symétrie de l'histogramme en fonction du nombre de lancers demandés.

Présentation de l'algorithme

Histogramme des pourcentages.

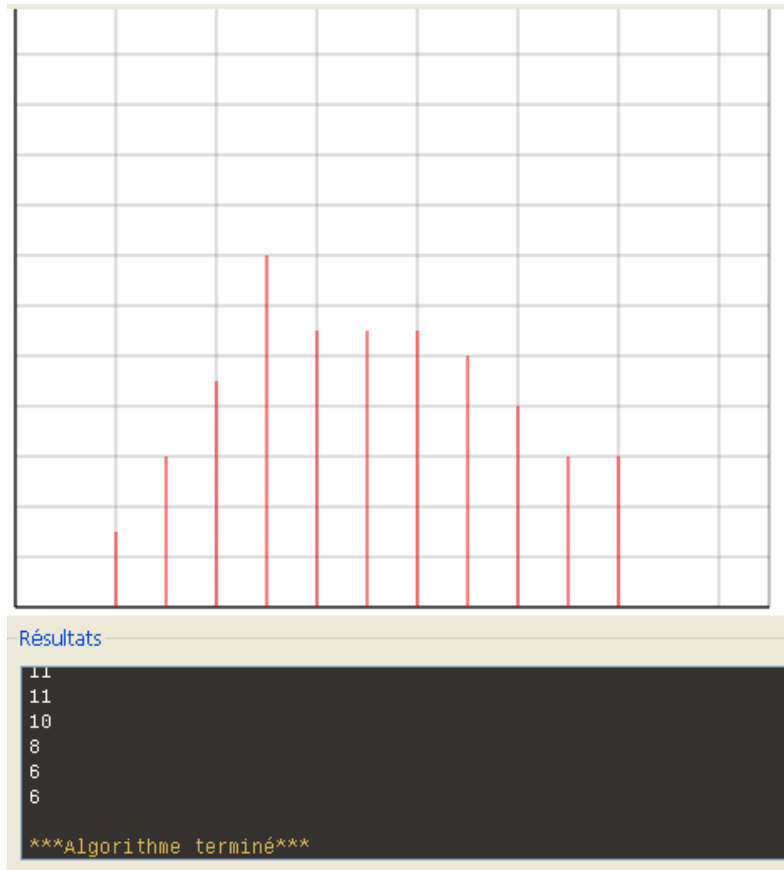
Code de l'algorithme

```

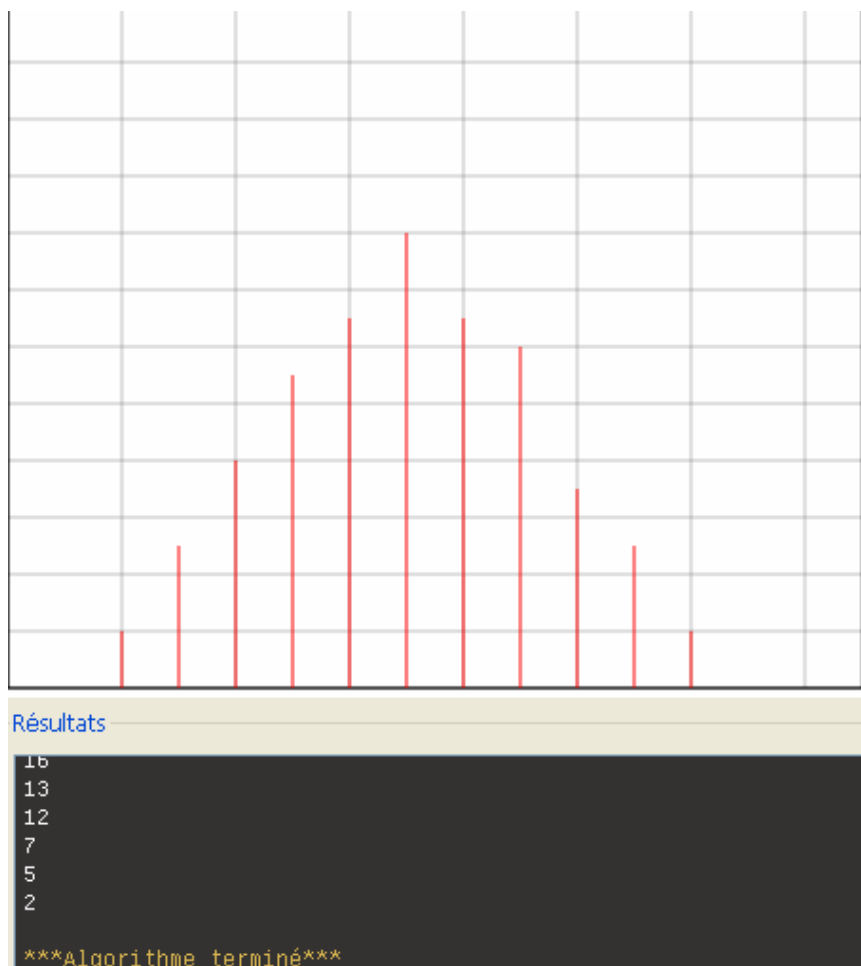
▼ VARIABLES
  | des1 EST_DU_TYPE NOMBRE
  | des2 EST_DU_TYPE NOMBRE
  | sigma EST_DU_TYPE NOMBRE
  | i EST_DU_TYPE NOMBRE
  | compteur EST_DU_TYPE NOMBRE
  | somme EST_DU_TYPE NOMBRE
  | combien EST_DU_TYPE NOMBRE
  | apparition EST_DU_TYPE LISTE
  | pourcent EST_DU_TYPE LISTE
▼ DEBUT_ALGORITHME
  | LIRE combien
  ▼ POUR somme ALLANT_DE 2 A 12
    | DEBUT_POUR
    | apparition[somme] PREND_LA_VALEUR 0
    ▼ POUR i ALLANT_DE 1 A combien
      | DEBUT_POUR
      | des1 PREND_LA_VALEUR floor(random()*6+1)
      | des2 PREND_LA_VALEUR floor(random()*6+1)
      | sigma PREND_LA_VALEUR des1+des2
      ▼ SI (sigma==somme) ALORS
        | DEBUT_SI
        | apparition[somme] PREND_LA_VALEUR apparition[somme]+1
        | FIN_SI
      | FIN_POUR
    | FIN_POUR
  ▼ POUR somme ALLANT_DE 2 A 12
    | DEBUT_POUR
    | pourcent[somme] PREND_LA_VALEUR floor((apparition[somme]*100)/combien)
    | AFFICHER pourcent[somme]
    | TRACER_SEGMENT (somme,0)->(somme,pourcent[somme])
    | FIN_POUR
  FIN_ALGORITHME

```

Voici le graphe pour 100 lancers :



Voici le graphe pour 1000 lancers :



I) Ce que disent les nouveaux programmes.Algorithmes, objectifs pour le lycée, extrait du programme.

La démarche algorithmique est, depuis les origines, une composante essentielle de l'activité mathématique. Au collège, les élèves ont rencontré des algorithmes (Algorithmes opératoires, algorithmes des différences, algorithmes d'Euclide, algorithmes de construction en géométrie.....)

Ce qui est proposé dans les programmes est une formalisation en langage naturel propre à donner lieu à traduction sur une calculatrice ou à l'aide d'un logiciel. Il s'agit de familiariser les élèves avec les trois grands principes de l'organisation d'un algorithme :

- Gestion des entrées – sorties.
- Affectation d'une valeur
- Mise ne forma d'un calcul.

Dans le cadre de cette activité logarithmique les élèves sont entraînés :

- A décrire certains algorithmes en langage naturel ou dans un langage symbolique ;
- A en réaliser quelques uns à l'aide d'un tableur ou d'un petit programme réalisé sur une calculatrice ou avec un logiciel adapté.
- A interpréter des algorithmes plus complexes.

Aucun langage n aucun logiciel n'est imposé.

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 calculatoire et donnant la valeur d'une fonction.
- D'utiliser les instructions d'entrées et sorties nécessaires au traitement.

Boucle itérative, 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.
- D'écrire un programme calculant et donnant la valeur d'une fonction.
- De programmer une instruction conditionnelle, un calcul itératif, avec une fin de boucle conditionnelle.

L'algorithmique a une place naturelle dans tous les champs des mathématiques et les problèmes posés doivent être en relation avec les autres parties du programme : (Fonctions, géométrie, statistiques et probabilités, logique.) mais aussi avec les autres disciplines ou bien dans la vie courante.

A l'occasion de l'écriture d'algorithmes et de petits programmes, il convient de donner aux élèves de bonnes habitudes de rigueur et de les entraîner aux pratiques systématiques de vérification et de contrôle.

II) Qu'est ce qu'un algorithme ?

Une définition :

« Un algorithme est une suite finie de règles à appliquer dans un ordre déterminé à un nombre fini de données pour arriver, en un nombre fini d'étapes, à un certain résultat et cela indépendamment des données. »

Cette notion est très répandue dans la vie courante, un algorithme peut par exemple y prendre la forme :

- d'une recette de cuisine.
- D'un mode d'emploi.
- D'une notice de montage.
- D'une partition musicale.
- D'un texte de loi.
- D'un itinéraire routier.

III) La démarche algorithmique.

La résolution d'un problème passe par deux étapes:

2. Concevoir une procédure qui une fois appliquée amènera à une solution du problème posé : c'est ce qu'on appelle l'*algorithmique*.
3. Résoudre effectivement le problème posé en appliquant cet algorithme

Quelques algorithmes en Mathématiques connus des élèves:

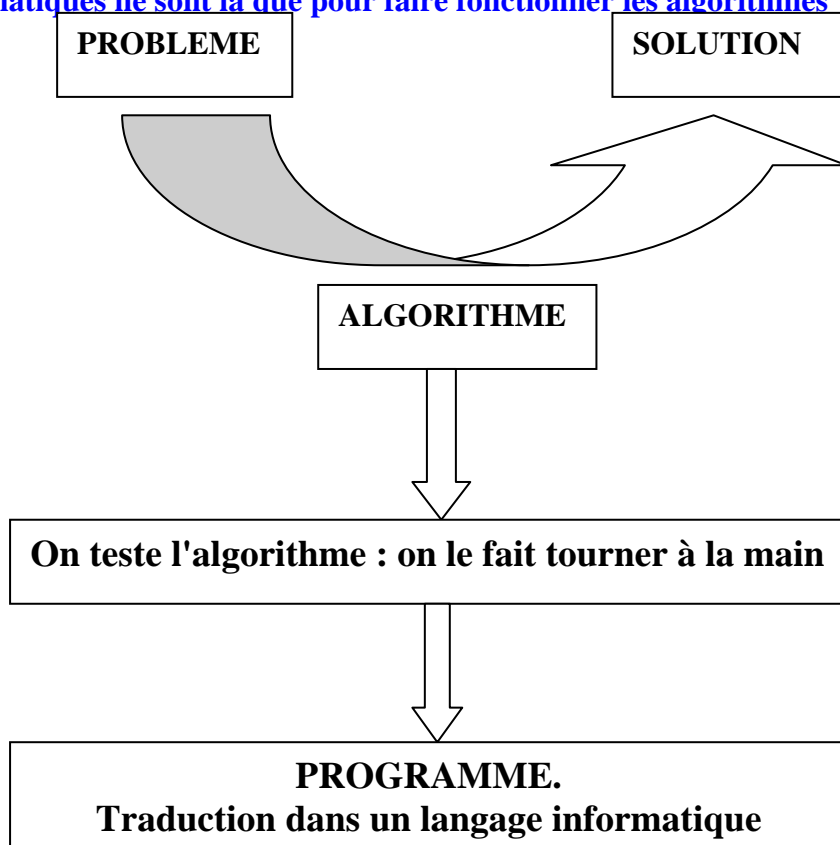
- algorithmes opératoires
- algorithmes de constructions géométriques (notamment avec un logiciel de géométrie)
- algorithme d'Euclide
- algorithme des différences
- Et bien d'autres encore...

Un algorithme décrit les différentes étapes qui permettent de résoudre un problème.

Il est donc écrit dans un langage naturel. Néanmoins, l'étape 2 conditionne souvent la façon de concevoir l'algorithme. Tout dépend de la personne ou de la machine qui va l'utiliser: tout dépend de son niveau de compréhension

Dans le cas de l'utilisation d'une machine (ordinateur ou calculatrice) l'algorithme doit être rendu **compréhensible** par la machine que nous allons utiliser pour résoudre effectivement le problème. Le résultat de la traduction de l'algorithme dans un langage connu de la machine est appelé un *programme*.

Les langages informatiques ne sont là que pour faire fonctionner les algorithmes



Un algorithme bien conçu peut être traduit dans la plupart des langages informatiques courants

Les principaux logiciels gratuits sont les suivants :

- **ALGOBOX**
- **SCRATCH**
- **XCAS**
- **SCILAB**
- **PYTHON**

Dans la suite, nous utiliserons le logiciel ALGOBOX pour traduire et exécuter les algorithmes.

Ce logiciel gratuit peut se télécharger à l'adresse suivante:

<http://www.xm1math.net/algobox/index.html>

AlgoBox est un logiciel libre, multi-plateforme et **gratuit** d'aide à l'élaboration et à l'exécution d'algorithmes dans **l'esprit du nouveau programme de seconde**. Il a été conçu par Pascal Brachet, qui est l'auteur de PstPlus et de Texmaker, logiciels forts utiles pour les utilisateurs de LaTeX.

IV) Pourquoi choisir AlgoBox ?

Avantages

- Les commandes sont en Français
- L'interface est simple et limpide ce qui évite à l'élève tout problème d'apprentissage de syntaxe et lui permet donc de se concentrer sur la logique algorithmique.
- Il répond parfaitement à une initiation à l'algorithmique et aucune connaissance d'un langage particulier n'est nécessaire.

Inconvénients: langage assez limité

Par exemple ne permet pas de faire des sous programmes (procédures ou fonctions), graphiques

V) Comment concevoir un algorithme ?

Un algorithme se compose en général de trois parties :



C'est un schéma simplifié. Il arrive d'ailleurs que les sorties se fassent au cours du traitement.

Pour écrire un algorithme, il faut commencer par l'analyse du problème

Il s'agit de déterminer les données utilisées et les opérations à effectuer sur ces données et de trouver la démarche pour résoudre le problème.

VI) Les variables.

Un algorithme manipule des données, données initiales ou résultats intermédiaires.

Ces données pour pouvoir être utilisées par un algorithme doivent être rangées dans des **VARIABLES**

Attention: Une variable en algorithmique est différente d'une variable en mathématiques (ce qui peut poser un problème pour les élèves)

Du point de vue de l'**ordinateur**, une variable est une zone de mémoire au contenu de laquelle on accède via un identificateur. Du point de vue **algorithmique**, une variable est caractérisée par:

- **son nom**
- **sa structure** : variable simple, liste, tableau, fichier
- **sa nature**: numérique, chaîne de caractères, booléen
- **son contenu** qui peut changer au cours de l'exécution de l'algorithme

BOOLEEN : Une **variable booléenne** est une variable qui ne peut prendre que deux valeurs **VRAI** ou **FAUX**.

Son intérêt: nécessite moins de place en mémoire qu'une variable numérique

Ce type de variable n'existe pas dans tous les langages.

LISTE – TABLEAUX : Les listes et les tableaux sont des suites de données indicées caractérisées par un **nom**, le **nombre d'éléments** et la **nature** des données qu'ils contiennent. On accède à un élément de la liste ou du tableau en indiquant sa position :

LISTE [I] est le terme de rang I de la liste nommée LISTE

TABLEAU [I,J] est le terme situé ligne I, colonne J du tableau à deux dimensions

En algorithmique, les **listes** et les **tableaux** sont deux structures de données différentes. Une **liste** est une variable dynamique: la taille n'est pas donnée lors de la déclaration, on peut ajouter des éléments à une liste (augmenter le nombre d'éléments). Un **tableau** est une variable statique: il a des dimensions prédéfinies lors de sa déclaration ; on peut seulement modifier les éléments qu'il contient.

Dans la plupart des langages de programmation, il est nécessaire de déclarer toutes les variables utilisées dès le début du programme La déclaration de variables n'initialise pas les variables.

Une variable ne peut être utilisée que si elle est initialisée c'est-à-dire si elle a un contenu.

Certaines erreurs dans les algorithmes viennent donc parfois de la :

- *Non déclaration de la variable*
- *Non initialisation de la variable.*

Il peut être préférable pour les élèves d'utiliser des noms de variables significatifs qui ont un lien avec leur contenu.

Par exemple:

salaire au lieu de S

longueur au lieu de L

Les opérations que l'on peut effectuer sur les données dépendent de la nature de ces variables.

VII) Les instructions.

Les instructions sont les « règles » que l'on applique aux données

Instruction d'entrée

Il s'agit de demander à l'utilisateur une valeur que l'on va attribuer à une variable

Plusieurs formulations sont possibles:

Demander A

Lire A

Saisir A

Instruction de sortie

Il s'agit d'afficher (ou d'imprimer) les résultats, c'est-à-dire le contenu de certaines variables ou bien des messages

Afficher A

Afficher « message »

Instruction d'affectation

Il s'agit d'affecter c'est-à-dire d'attribuer une valeur à une variable. Cette valeur peut être une constante ou le résultat d'un calcul.

Plusieurs formulations sont possibles:

Donner à A la valeur 2

A prend la valeur 2

Affecter à A la valeur 2

Stocker dans A la valeur 2

Pour modifier le contenu par un calcul

augmenter A de 2

remplacer A par la valeur A+2

Donner à A la valeur A+2

Affecter à A la valeur A+2

Stocker dans A la valeur A+2

A prend la valeur A+2

A A+2

Les deux premières sont plus parlantes pour les élèves mais ainsi formulées, ne sont pas comprises par les langages de programmation

VIII) Les premiers exemples.

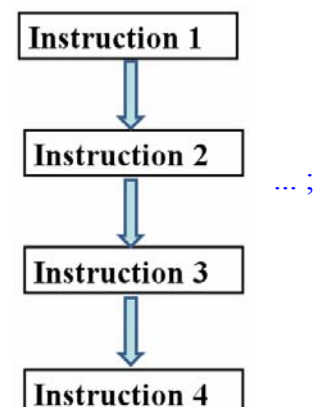
Dans les algorithmes qui suivent le traitement est un traitement séquentiel

Les instructions y sont exécutées les unes après les autres par ordre d'apparition.

instruction 1;

instruction 2;

instruction n.



Il s'agit d'exercices qui n'utilisent que :

- Des instructions d'entrée et de sortie
- Des instructions d'affectation
- Des calculs

Exercice 1 :

Un algorithme important

Ecrire un algorithme qui échange le contenu de deux variables A et B

Il permet de comprendre le fonctionnement des variables.

C'est le fonctionnement du presse papiers des ordinateurs.

Corrections de l'exercice 1 :

On va faire l'algorithme avec des variables numériques. Il est évident qu'il marche avec n'importe quel autre type de variable

Variables:

- A un nombre
- B un nombre
- C un nombre

Entrée

- Demander A
- Demander B

Traitement

- Donner à C la valeur de A
- Donner à A la valeur de B
- Donner à B la valeur de C

Sortie

- Afficher « la valeur de A est: », A
- Afficher « la valeur de B est: », B

Comment se présente un algorithme dans ALGOBOX ?

The screenshot shows the ALGOBOX software interface. At the top, there is a menu bar with 'Fichier', 'Edition', 'Tutoriel', 'Affichage', and 'Aide'. Below the menu bar is a toolbar with icons for 'Nouveau', 'Ouvrir', 'Sauver', and 'Tester'. The main workspace is divided into two sections: 'Présentation de l'algorithme' (empty) and 'Code de l'algorithme' (containing a tree view with 'VARIABLES', 'DEBUT_ALGORITHME', and 'FIN_ALGORITHME'). To the right of the code editor is a panel with buttons for 'Modifier Ligne', 'Supprimer Ligne/Bloc', and a yellow note box containing instructions: '- Pour utiliser une variable, il faut d'abord la déclarer (bouton "Déclarer nouvelle variable").' and '- Pour ajouter un nouvel élément à l'algorithme, il faut d'abord insérer une nouvelle ligne (bouton "Nouvelle Ligne"), puis cliquer sur un des boutons disponibles dans le panneau "Ajouter code".'. Below the code editor are buttons for 'Tester Algorithme' and 'Nouvelle Ligne'. At the bottom, there is a palette of operations: 'Opérations standards', 'Utiliser une fonction numérique', and 'Dessiner dans un repère'. The 'Opérations standards' section includes buttons for 'Déclarer nouvelle variable', 'Ajouter LIRE variable', 'AFFECTER valeur à variable', 'Ajouter AFFICHER Variable', 'Ajouter AFFICHER Message', 'Ajouter SI...ALORS', 'Ajouter POUR...DE...A', and 'Ajouter TANT QUE...'. To the right of this palette are buttons for 'Pause' and 'Commentaire'.

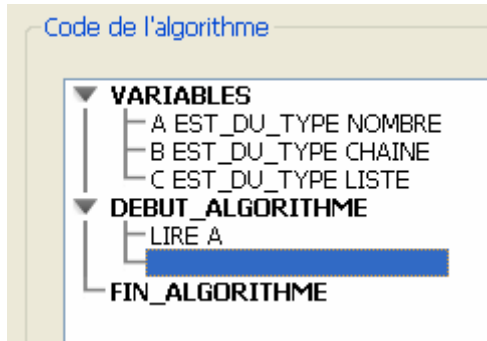
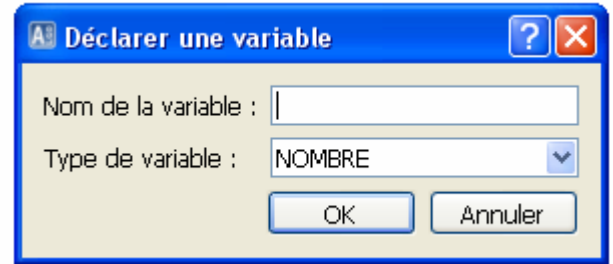
Les variables avec ALGOBOX

Les variables doivent être déclarées en début de programme à l'aide de la touche :



Il faut donner un nom à cette variable et choisir son type parmi les trois types proposés:

- ✓ **Nombre**
- ✓ **Chaîne**
- ✓ **Liste**

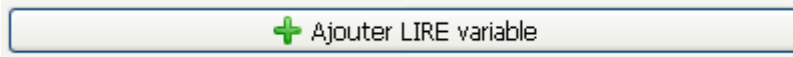


Instructions d'entrée :

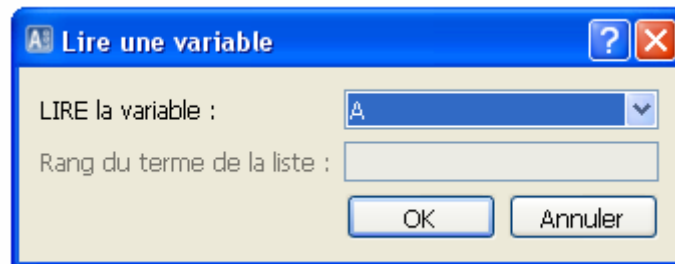
Pour entrer une instruction il faut d'abord créer une nouvelle ligne



On se place sur la ligne vide et on actionne la touche :



La boîte de dialogue suivante apparaît :



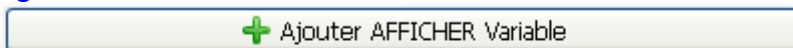
Et on choisit alors la variable que l'on veut utiliser.

Instructions de sortie :

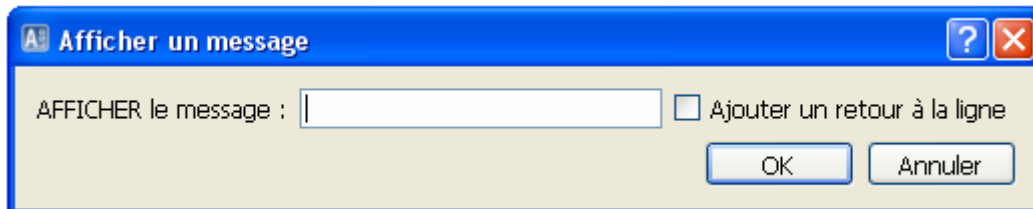
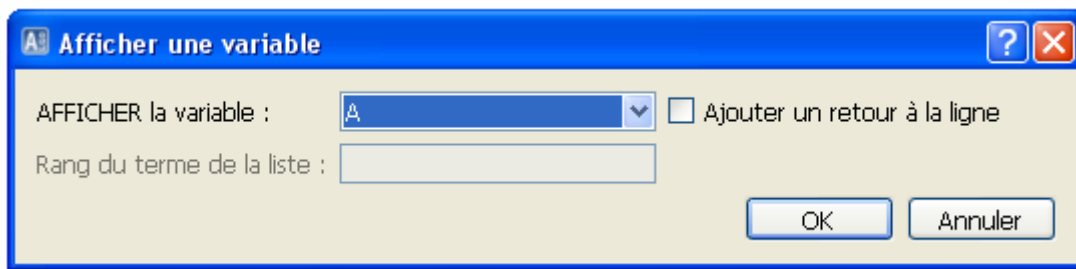
Pour entrer une instruction il faut d'abord créer une nouvelle ligne



On se place ensuite sur la ligne vide et on actionne la touche :



Les boîtes de messages suivantes apparaissent alors.

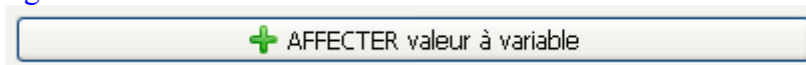


Instructions d'affectation :

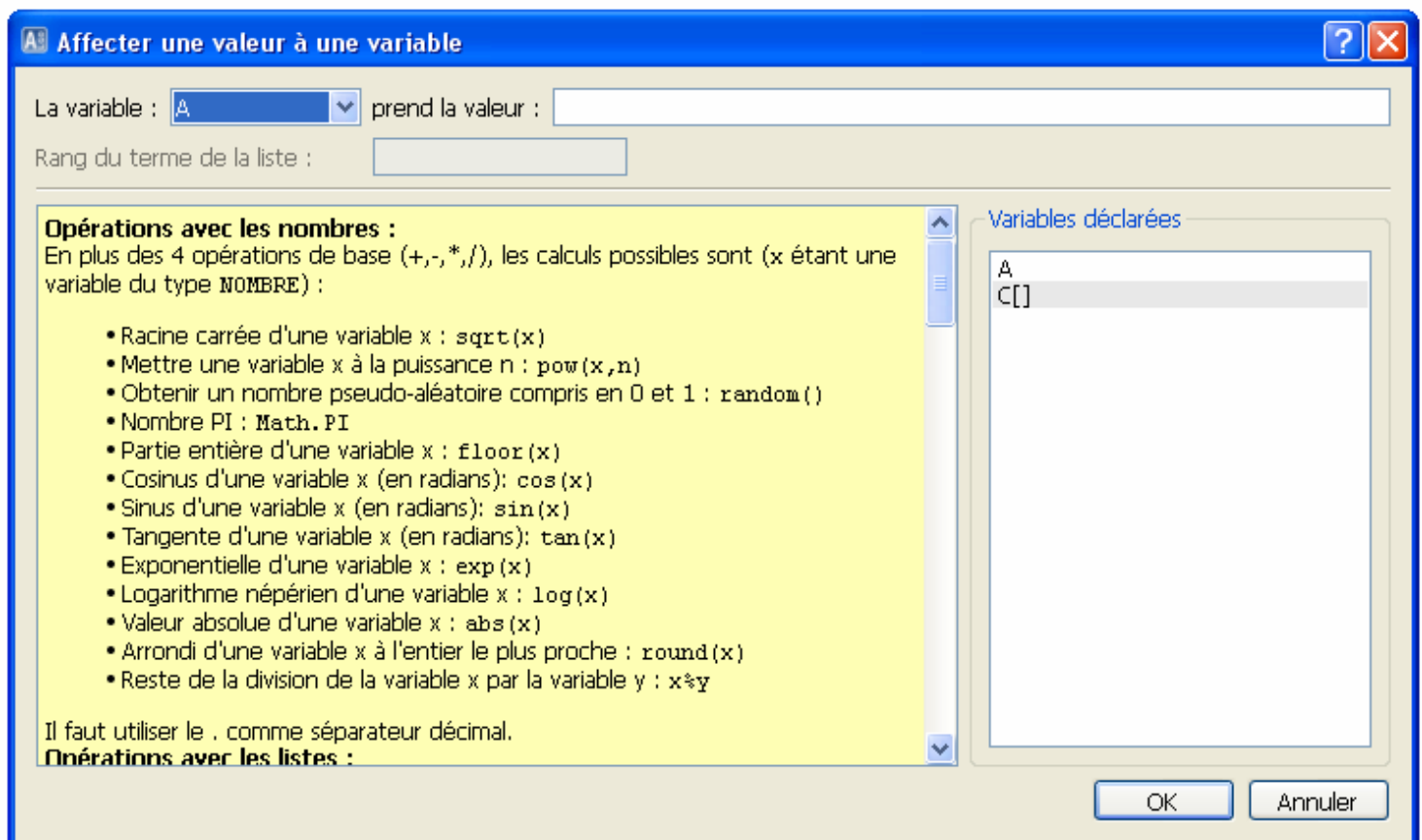
Pour entrer une instruction il faut d'abord créer une nouvelle ligne



On se place ensuite sur la ligne vide et on actionne la touche :



La boîte de dialogue suivante apparaît :



L'algorithme est alors le suivant :

Présentation de l'algorithme

Algorithme qui échange le contenu de deux variables A et B entrées par l'utilisateur.

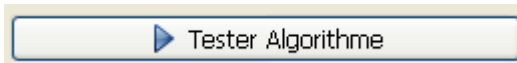
Code de l'algorithme

```

VARIABLES
  A EST_DU_TYPE NOMBRE
  B EST_DU_TYPE NOMBRE
  C EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE A
  LIRE B
  AFFICHER "La valeur de A est:"
  AFFICHER A
  AFFICHER "La valeur de B est:"
  AFFICHER B
  C PREND_LA_VALEUR A
  A PREND_LA_VALEUR B
  B PREND_LA_VALEUR C
  AFFICHER "La nouvelle valeur de A est:"
  AFFICHER A
  AFFICHER "La nouvelle valeur de B est:"
  AFFICHER B
FIN_ALGORITHME

```

Et quand on lance l'algorithme, avec la touche



ALGOBOX : ECHANGE

PRÉSENTATION DE L'ALGORITHME :

Algorithme qui échange le contenu de deux variables A et B entrées par l'utilisateur

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  A EST_DU_TYPE NOMBRE
3  B EST_DU_TYPE NOMBRE
4  C EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6  LIRE A
7  LIRE B
8  AFFICHER "La valeur de A est : "
9  AFFICHER A
10 AFFICHER "La valeur de B est : "
11 AFFICHER B
12 C PREND_LA_VALEUR A
13 A PREND_LA_VALEUR B
14 B PREND_LA_VALEUR C
15 AFFICHER "la nouvelle valeur de A est: "
16 AFFICHER A
17 AFFICHER "la nouvelle valeur de B est : "
18 AFFICHER B
19 FIN_ALGORITHME

```

Résultats

```

***Algorithme lancé***
La valeur de A est : 124
La valeur de B est : 256
la nouvelle valeur de A est: 256
la nouvelle valeur de B est : 124
***Algorithme terminé***

```

Remarque : si les messages "Algorithme lancé" et "Algorithme terminé" n'apparaissent pas au bout d'un moment dans la zone ci-dessous, c'est que l'algorithme contient une erreur.

Les opérations sur les variables numériques avec ALGOBOX

Le séparateur décimal est le point.

Les quatre opérations: +, -, *, /

Le signe * est obligatoire : Il faut écrire 2*x et non 2x

Les autres opérations

Racine carrée d'une variable x	sqrt(x)
Mettre une variable x à la puissance n	pow(x,n)
Obtenir un nombre pseudo-aléatoire compris en 0 et 1	random()
Nombre PI	Math.PI
Partie entière d'une variable x	floor(x)
Cosinus d'une variable x (en radians)	cos(x)
Sinus d'une variable x (en radians)	sin(x)
Tangente d'une variable x (en radians)	tan(x)
Exponentielle d'une variable x	exp(x)
Logarithme népérien d'une variable x	log(x)
Valeur absolue d'une variable x	abs(x)
Arrondi d'une variable x à l'entier le plus proche	round(x)
Reste de la division de la variable x par la variable y	x%y

Il faut respecter les majuscules et minuscules

Exercice 2 :

Dans les sujets du brevet des collèges, on trouve souvent des programmes de calculs du style:

- Choisir un nombre
- Lui ajouter 4
- Multiplier la somme obtenue par le nombre choisi
- Ajouter 4
- Ecrire le résultat obtenu

Ecrire un algorithme que l'on puisse faire tourner sur une machine

Corrections de l'exercice 2 :

Commentaires

L'exercice précédent permet de comprendre la différence entre un algorithme pour un être humain ou pour une machine. Un algorithme pour ordinateurs ne doit contenir que des instructions compréhensibles par la machine. Un ordinateur ne sait faire que:

- Mettre des valeurs dans des zones mémoires
- Faire des opérations sur des variables
- Faire des tests et des branchements

Il permet aussi de travailler sur les expressions algébriques et les fonctions

Variables:

Choisir le nombre a.
Puis b, c et d

Entrée

Demander a.

Ajouter 4, afficher la somme

Multiplier la somme b par a.

Afficher le produit c.

Ajouter 4 à c qui donne d.

Sortie

Afficher le résultat obtenu, d

Présentation de l'algorithme

Sujet du brevet|

Code de l'algorithme

```

VARIABLES
├── a EST_DU_TYPE NOMBRE
├── b EST_DU_TYPE NOMBRE
├── c EST_DU_TYPE NOMBRE
└── d EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
├── LIRE a
├── AFFICHER a
├── b PREND_LA_VALEUR a+4
├── AFFICHER b
├── c PREND_LA_VALEUR b*a
├── AFFICHER c
├── d PREND_LA_VALEUR c+4
├── AFFICHER "le résultat obtenu est "
├── AFFICHER d
└── FIN_ALGORITHME

```

b.

PRÉSENTATION DE L'ALGORITHME :

Sujet du brevet

CODE DE L'ALGORITHME :

```

1  VARIABLES
2    a EST_DU_TYPE NOMBRE
3    b EST_DU_TYPE NOMBRE
4    c EST_DU_TYPE NOMBRE
5    d EST_DU_TYPE NOMBRE
6  DEBUT_ALGORITHME
7    LIRE a
8    AFFICHER a

```

Résultats

```

***Algorithme lancé***
7
11
77
le résultat obtenu est
81
***Algorithme terminé***

```

Exercice 3 :

Ecrire un algorithme qui calcule le périmètre d'un cercle et l'aire d'un disque dont on donne le rayon.

Corrections de l'exercice 3 :**Variables:**

- rayon : un nombre
- périmètre : un nombre
- aire : un nombre

Entrée

- demander la valeur de rayon

Traitement

- donner à périmètre la valeur $2 \cdot \pi \cdot \text{rayon}$
- donner à aire la valeur $\pi \cdot \text{rayon}^2$

Sortie

- afficher « le périmètre est: »
- afficher périmètre
- afficher « aire du disque est »
- afficher aire

Avec la TI 89, on obtient le programme suivant :

```

aireperi ()
Prgm
ClrIO
Disp "Rayon ? : "
Prompt r
when (r≥0, 2*π^2, when (r<0, "Erreur"), π*r^2)→f
Disp "Aire :"
Disp f
when (r≥0, 2*π*r, when (r<0, "Erreur"), 2*π*r)→p
Disp "Perimetre :"
Disp p
EndPrgm

```

[Avec AlgoBox, on obtient le programme suivant :](#)

On traduit l'algorithme dans ALGOBOX. **Attention à respecter les majuscules et les minuscules des différentes instructions.**

Présentation de l'algorithme

Algorithme qui calcule l'aire et le perimetre d'un cercle de rayon donné.

Code de l'algorithme

```

VARIABLES
  Rayon EST_DU_TYPE NOMBRE
  Perimetre EST_DU_TYPE NOMBRE
  Aire EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  AFFICHER "Donner la valeur du rayon"
  LIRE Rayon
  Perimetre PREND_LA_VALEUR 2*Math.PI*Rayon
  Aire PREND_LA_VALEUR Math.PI*pow(Rayon,2)
  AFFICHER "Le perimetre est"
  AFFICHER Perimetre
  AFFICHER "L'aire du disque est"
  AFFICHER Aire
FIN_ALGORITHME

```

Pour lancer cet algorithme, on procède de la manière suivante :

The screenshot shows the AlgoBox Test application window. The title bar reads "AlgoBox Test". The main window content is as follows:

ALGOBOX : AIRE ET PERIMETRE D'UN DISQUE

PRÉSENTATION DE L'ALGORITHME :

Algorithme qui calcule l'aire et le perimetre d'un cercle de rayon donné.

CODE DE L'ALGORITHME :

```

1  VARIABLES
2    Rayon EST_DU_TYPE NOMBRE
3    Perimetre EST_DU_TYPE NOMBRE
4    Aire EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6    AFFICHER "Donner la valeur du rayon"
7    LIRE Rayon
8    AFFICHER Rayon

```

Résultats

```

***Algorithme lancé***
Donner la valeur du rayon
45
Le perimetre est
282.74334
L'aire du disque est
6361.7251
***Algorithme terminé***

```

On the right side of the window, there are several control buttons:

- Lancer Algorithme (with a play icon)
- Mode pas à pas
- Continuer (with a play icon)
- Arrêter (with a stop icon)
- Fermer

Exercice 4 :

Ecrire un algorithme qui calcule les coordonnées du milieu d'un segment

Corrections de l'exercice 4 :**Variables:**

- x_A , y_A , x_B , y_B des nombres.
- X milieu et y milieu des nombres.

Entrée

- Demander x_A
- Demander y_A
- Demander x_B
- Demander y_B

Traitement

- Donner à x milieu la valeur de $(x_A+x_B)/2$
- Donner à y milieu la valeur de $(y_A+y_B)/2$

Sortie

- Afficher «L'abscisse du milieu est : », x milieu
- Afficher «L'ordonnée du milieu est : », y milieu

Avec la TI 89, on obtient le programme suivant :

```
milieu()
Prgm
ClrIO
Input "L'abscisse du premier point est ", xa
Input "L'abscisse du second point est ", xb
Input "L'ordonnée du premier point est ", ya
Input "L'ordonnée du second point est ", yb
Disp "Les coordonnées du milieu I de [AB] sont", {(xa+xb)/2}, {(ya+yb)/2}
EndPrgm
```

Présentation de l'algorithme

Algorithme calculant les coordonnées du milieu I d'un segment [AB].

Code de l'algorithme

VARIABLES

```
xA EST_DU_TYPE NOMBRE
yA EST_DU_TYPE NOMBRE
xB EST_DU_TYPE NOMBRE
yB EST_DU_TYPE NOMBRE
xI EST_DU_TYPE NOMBRE
yI EST_DU_TYPE NOMBRE
```

DEBUT_ALGORITHME

```
AFFICHER "Donner la valeur de xA"
LIRE xA
AFFICHER xA
AFFICHER "Donner la valeur de yA"
LIRE yA
AFFICHER yA
AFFICHER "Donner la valeur de xB"
LIRE xB
AFFICHER xB
AFFICHER "Donner la valeur de yB"
LIRE yB
AFFICHER yB
xI PREND_LA_VALEUR (xA+xB)/2
AFFICHER "L'abscisse du milieu est"
AFFICHER xI
yI PREND_LA_VALEUR (yA+yB)/2
AFFICHER "L'ordonnée du milieu est"
AFFICHER yI
```

FIN_ALGORITHME

Avec Algotbox, on obtient le programme suivant :

On traduit l'algorithme dans ALGOBOX.

Résultats

```
***Algorithme lancé***
Donner la valeur de xA
2
Donner la valeur de yA
4
Donner la valeur de xB
6
```

```
Donner la valeur de yB
8
L'abscisse du milieu est
4
L'ordonnée du milieu est
6
***Algorithme terminé***
```

Exercice 5 : Les fonctions dans Algobox.

Calcul de l'image d'un nombre par une fonction donnée/

Ecrire un algorithme qui calcule et affiche l'image d'un nombre réel par une fonction f donnée.

Corrections de l'exercice 5 :

Pour utiliser une fonction notée par exemple F1, il faut la définir en dehors de l'algorithme. On procède de la manière suivante :

☞ Activer l'option **Utiliser une fonction** dans l'onglet « **Utiliser une fonction numérique** ».

☞ Entrer l'expression de $F1(x)$ en fonction de x dans la boîte de dialogue définie pour cela.

Pour utiliser l'image d'un nombre x par la fonction $F1$ dans l'algorithme, il suffit d'utiliser le code $F1(x)$

ce qui donne le résultat suivant :

PRÉSENTATION DE L'ALGORITHME :
.....

Image par la fonction carré

CODE DE L'ALGORITHME :
.....

```

1  VARIABLES
2    a EST_DU_TYPE NOMBRE
3    b EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5    LIRE b
6    AFFICHER b
7    a PREND_LA_VALEUR F1(b)
8    AFFICHER a

```

Résultats

```

***Algorithme lancé***
4
16
***Algorithme terminé***

```

AIDE

INSTRUCTION CONDITIONNELLE OU ALTERNATIVE.

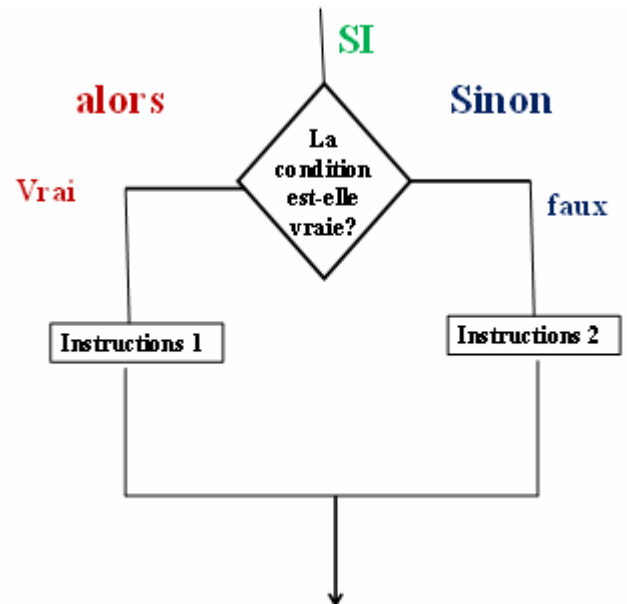
(1) La forme simple :

Si	<i>condition</i>	Alors
	<i>Instructions</i>	
FinSi		

(2) La forme la plus courante:

Si	<i>condition</i>	Alors
	<i>Instructions 1</i>	
Sinon		
	<i>Instructions 2</i>	
FinSi		

On peut schématiser ce fonctionnement de la façon suivante :



(3) Conditions et tests:

Une condition est basée sur une comparaison

Opérateurs : =, <, >, ≠, <=, >=

Conditions composées : On peut mettre plusieurs conditions liées par **ET** ou **OU**

Par exemple $5 < x < 8$ se traduit par ($5 < x$ ET $x < 8$)

Les expressions conditionnelles avec Algobox sont les suivantes.

Etre égal à	==
Différent de	!=
Inférieur	<
Supérieur	>
Inférieur ou égal	<=
Supérieur ou égal	>=

On peut combiner des conditions avec **ET** ou **OU**.

Exercice 6 : Un classique important.

Ecrire un algorithme qui donne le plus grand et le plus petit de deux nombres A et B.

Corrections de l'exercice 6 :

Variables:

- A un nombre.
- B un nombre.
- max un nombre.
- min un nombre.

Entrée

- Demander A
- Demander B

Traitement

- Si $A > B$ alors
 - ♦ Donner à max la valeur A
 - ♦ Donner à min la valeur B.

Sinon

- ♦ Donner à max la valeur B
- ♦ Donner à min la valeur A

FinSi

Sortie

- Afficher «Le maximum est : », max
- Afficher «Le minimum », min

Ce qui donne :

Présentation de l'algorithme	PRÉSENTATION DE L'ALGORITHME :	PRÉSENTATION DE L'ALGORITHME :
Plus grand de deux nombres A et B	Plus grand de deux nombres A et B	Plus grand de deux nombres A et B
Code de l'algorithme	CODE DE L'ALGORITHME :	CODE DE L'ALGORITHME :
<pre> VARIABLES - A EST_DU_TYPE NOMBRE - B EST_DU_TYPE NOMBRE - max EST_DU_TYPE NOMBRE - min EST_DU_TYPE NOMBRE DEBUT_ALGORITHME - LIRE A - AFFICHER A - LIRE B - AFFICHER B - SI (A>B) ALORS - DEBUT_SI - max PREND_LA_VALEUR A - AFFICHER "le maximum est" - AFFICHER max - min PREND_LA_VALEUR B - AFFICHER "le minimum est" - AFFICHER min - FIN_SI - SINON - DEBUT_SINON - max PREND_LA_VALEUR B - AFFICHER "le maximum est" - AFFICHER max - min PREND_LA_VALEUR A - AFFICHER "le minimum est" - AFFICHER min - FIN_SINON FIN_ALGORITHME </pre>	<pre> 1 VARIABLES 2 A EST_DU_TYPE NOMBRE 3 B EST_DU_TYPE NOMBRE 4 max EST_DU_TYPE NOMBRE 5 min EST_DU_TYPE NOMBRE 6 DEBUT_ALGORITHME 7 LIRE A 8 AFFICHER A </pre>	<pre> 1 VARIABLES 2 A EST_DU_TYPE NOMBRE 3 B EST_DU_TYPE NOMBRE 4 max EST_DU_TYPE NOMBRE 5 min EST_DU_TYPE NOMBRE 6 DEBUT_ALGORITHME 7 LIRE A 8 AFFICHER A </pre>
	Résultats	Résultats
	<pre> ****Algorithme lancé**** 4 3 le maximum est 4 le minimum est 3 ***Algorithme terminé*** </pre>	<pre> 4 5 le maximum est 5 le minimum est 4 ***Algorithme terminé*** </pre>

Prolongement:

Ecrire un algorithme qui détermine le plus grand de trois nombres A, B et C

Variables:

- A, B, C max des nombres.

Entrée

- Demander A
- Demander B
- Demander C

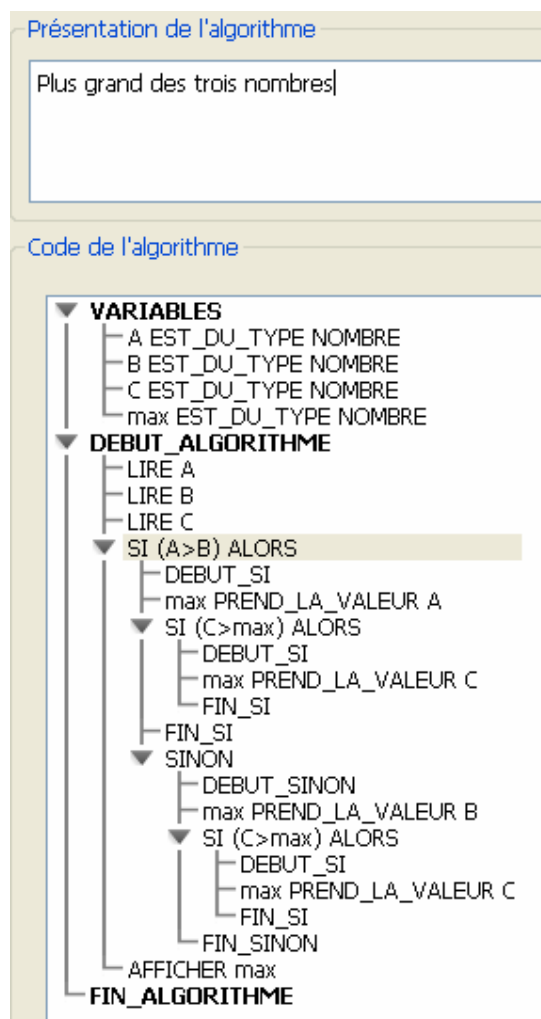
Traitement

- Si $A > B$ alors
 - ♦ Donner à max la valeur A
 - ♦ Si $C > \text{max}$ alors
 - Donner à max la valeur C.
 FinSi
- Sinon
 - ♦ Donner à max la valeur B
 - ♦ Si $C > \text{max}$ alors
 - Donner à max la valeur C
 FinSi
- FinSi

Sortie

- Afficher «Le maximum est : », max

Ce qui donne :

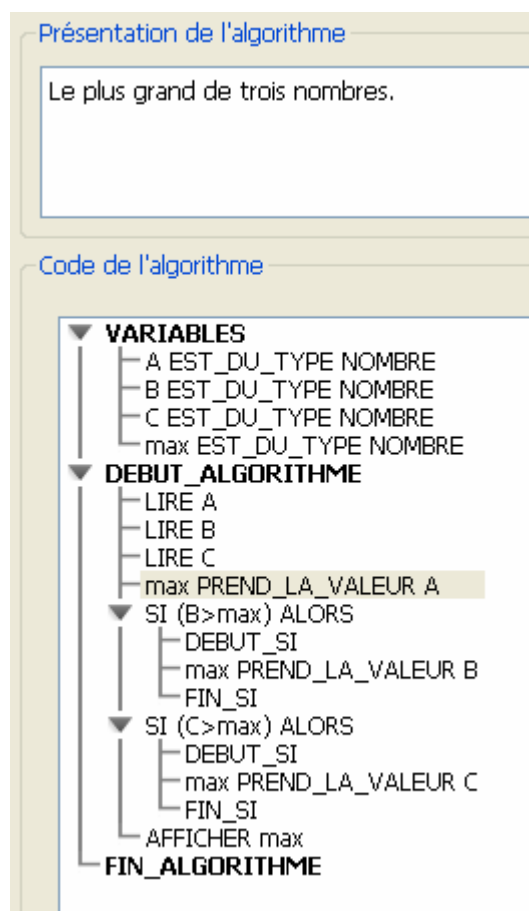


Une autre méthode consiste à supposer que le maximum est A au départ et ensuite comparer le max successivement à B puis à C.

Traitement :

- Donner à max la valeur de A.
- Si $B > \text{max}$ alors
 - Donner à max la valeur de B
- Sinon
 - Si $C > \text{max}$ alors
 - Donner à max la valeur de C
 - FinSi
- FinSi

Cet algorithme peut se généraliser à la recherche du maximum d'une liste de valeurs. Avec AlgoBox, on ne coche pas l'instruction sinon.



Exercice 7 :

On donne trois points A, B et C par leurs coordonnées dans un repère.

Ecrire un algorithme qui permet de savoir si le triangle ABC est rectangle ? Isocèle ? Equilatéral ?

Problème d'arrondis avec la racine carrée pour comparer des distances : problème habituel de l'utilisation des valeurs approchées.

Corrections de l'exercice 7 :

Variables:

- $x_A, y_A, x_B, y_B, x_C, y_C$ des nombres.
- AB, AC, BC des nombres.

Entrée

- Demander x_A, y_A
- Demander x_B, y_B
- Demander x_C, y_C

Traitement

- Donner à AB la valeur de $\text{racine}\left(\left(x_B - x_A\right)^2 + \left(y_B - y_A\right)^2\right)$.
- Donner à AC la valeur de $\text{racine}\left(\left(x_C - x_A\right)^2 + \left(y_C - y_A\right)^2\right)$.
- Donner à BC la valeur de $\text{racine}\left(\left(x_C - x_B\right)^2 + \left(y_C - y_B\right)^2\right)$.
- Si $AB^2 + AC^2 = BC^2$ alors afficher « ABC est rectangle en A ».
- Si $AB^2 + BC^2 = AC^2$ alors afficher « ABC est rectangle en B ».
- Si $BC^2 + AC^2 = AB^2$ alors afficher « ABC est rectangle en C ».

Remarque :

On a fait les trois tests. Donc à chaque exécution les trois tests seront exécutés. On peut améliorer ce programme en utilisant des si-imbriquées.

- Si $AB^2 + AC^2 = BC^2$ alors
 - afficher « rectangle en A »
- Sinon
 - Si $AB^2 + BC^2 = AC^2$ alors
 - afficher « rectangle en B »
 - Sinon
 - Si $BC^2 + AC^2 = AB^2$ alors
 - Afficher « rectangle en C »
 - FinSi
 - FinSi
- FinSi

Exercice 8 : Forfait SMS.

On compare trois forfaits mensuels pour des SMS.

- FORFAIT A : fixe de 20 € quelque soit le nombre de SMS envoyés.
- FORFAIT B : 0,15 € par SMS envoyés.
- FORFAIT C : fixe de 12 € et 0,05 € par SMS envoyés.

Elaborer une démarche permettant d'afficher le forfait le plus avantageux et le montant mensuel à régler, en euros, en fonction du nombre de SMS envoyés dans le mois.

Corrections de l'exercice 8 :

TRAITEMENT ITERATIF ET BOUCLES.

Il s'agit d'instructions qui permettent de **répéter** des instructions: on parle d'**itérations** ou de **boucles**.

Il existe deux types de boucles:

- Les boucles dont on connaît le nombre de répétitions
- Les boucles **conditionnelles** : on ne connaît pas à l'avance le nombre de répétitions

On connaît le nombre N d'itérations

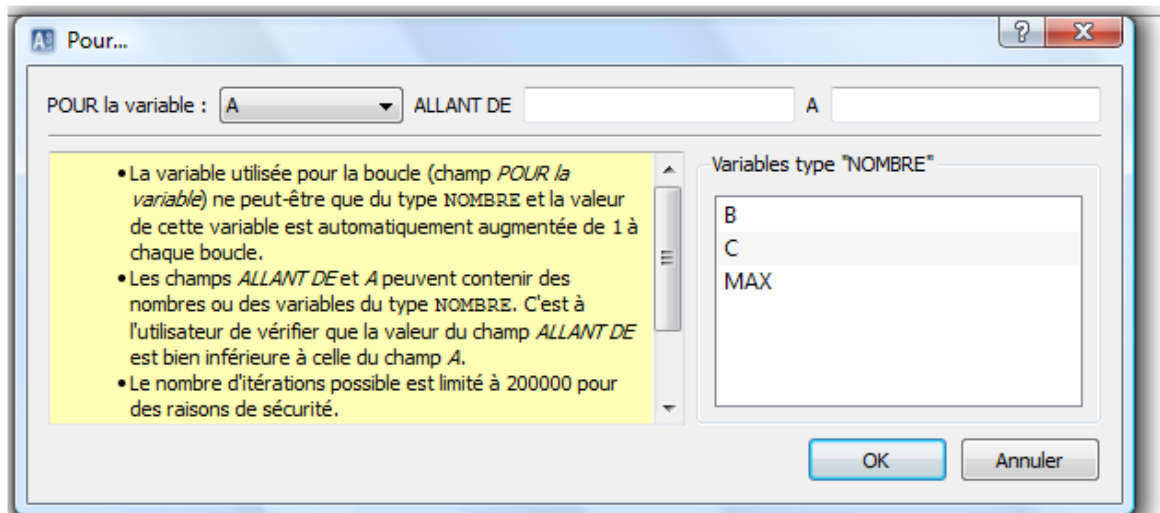
Répéter pour I allant de 1 à N
Instructions
 Fin Re peter (*Valeur suivante de I*)

- N doit être initialisé avant le début de la boucle
- I est une variable (appelée compteur) ; elle est initialisée à l'entrée de la boucle, incrémentée de 1 à chaque passage à Fin répéter et vaut N+1 en sortie de boucle

Dans certains langages de programmation, on peut modifier l'incrément. On a alors par exemple:

Répéter pour I allant de 1 à N pas 3
Instructions
 Fin Re peter (*Valeur suivante de I*)

Avec Algobox , on a :



Exercice 9 :

Ecrire un algorithme qui permet d'afficher les carrés des entiers compris entre 1 et N
Prolongement : Modifier l'algorithme pour qu'il calcule la somme des carrés.

Corrections de l'exercice 9 :

Variables:

- N est un nombre.
- Carre est un nombre.
- S est un nombre
- I est un nombre.

Entrée

- Demander N.

Traitement

- Donner à S la valeur 0.
- Répéter pour I allant de 1 à N.
 - Donner à carre la valeur de I^2 .
 - Afficher carré.
 - Donner à S la valeur de S + carre.
- Fin Répéter.

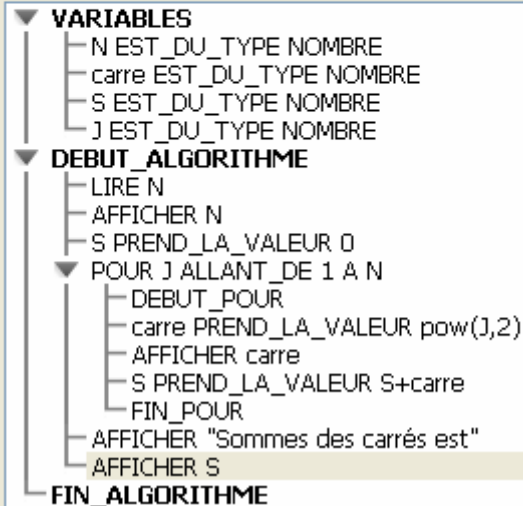
Sortie

- Afficher «La somme des carrés est », S.

Avec Algobox, on obtient :

Présentation de l'algorithme

Affichage du carré des nombres entiers de 1 à N puis le calcul de leur somme.

Code de l'algorithmePRÉSENTATION DE L'ALGORITHME :

Affichage du carré des nombres entiers de 1 à N puis le calcul de leur somme.

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  N EST_DU_TYPE NOMBRE
3  carre EST_DU_TYPE NOMBRE
4  S EST_DU_TYPE NOMBRE
5  J EST_DU_TYPE NOMBRE
6  DEBUT_ALGORITHME
7  LIRE N
8  AFFICHER N
9  S PREND_LA_VALEUR 0
10 POUR J ALLANT_DE 1 A N
11   DEBUT_POUR
12   carre PREND_LA_VALEUR pow(J,2)
13   AFFICHER carre
14   S PREND_LA_VALEUR S+carre
15   FIN_POUR
16 AFFICHER "Sommes des carrés est"
17 AFFICHER S
  
```

Résultats

```

1
4
9
16
25
Sommes des carrés est
55
  
```

▶ Lancer Algorithme

 Mode pas à pas

▶ Continuer

⏹ Arrêter

Fermer

Exercice 10 :

Ecrire un algorithme qui permet de calculer la somme des N premiers entiers naturels.

Sans utiliser les formules sur les suites

Corrections de l'exercice 10 :**Variabes:**

- N est un nombre.
- S est un nombre
- I est un nombre.

Entrée

- Demander N.

Traitement

- Donner à S la valeur 0.
- Répéter pour I allant de 1 à N.
 - Donner à S la valeur de S+I.
- Fin Répéter.

Sortie

- Afficher «La somme des », N, »premiers entiers est :»
- Afficher S

Avec AlgoBox, on obtient :

Présentation de l'algorithme

Algorithme qui permet de calculer la somme des N premiers entiers naturels.

Code de l'algorithme

```

VARIABLES
├── N EST_DU_TYPE NOMBRE
├── S EST_DU_TYPE NOMBRE
├── I EST_DU_TYPE NOMBRE
└── DEBUT_ALGORITHME
    ├── LIRE N
    ├── AFFICHER N
    ├── S PREND_LA_VALEUR 0
    └── POUR I ALLANT_DE 1 A N
        ├── DEBUT_POUR
        ├── S PREND_LA_VALEUR S+I
        ├── FIN_POUR
        ├── AFFICHER "Sommes des , N, premiers entiers est"
        ├── AFFICHER S
        └── FIN_ALGORITHME
  
```

Algorithme qui permet de calculer la somme des N premiers entiers naturels.

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  N EST_DU_TYPE NOMBRE
3  S EST_DU_TYPE NOMBRE
4  I EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6  LIRE N
7  AFFICHER N
8  S PREND_LA_VALEUR 0
9  POUR I ALLANT_DE 1 A N
10 DEBUT_POUR
11   S PREND_LA_VALEUR S+I
12 FIN_POUR
13 AFFICHER "Sommes des , N, premiers entiers est"
14 AFFICHER S
15 FIN_ALGORITHME
  
```

RÉSULTATS :

Algorithme lancé

Résultats

```

***Algorithme lancé***
5
Sommes des , N, premiers entiers est
15
***Algorithme terminé***
  
```

Lancer Algorithme

Mode pas à pas

Continuer

Arrêter

Fermer

Exercice 11 :

Ecrire un algorithme qui permet de calculer la somme (ou la moyenne) de N nombres donnés par l'utilisateur.

Cela peut être l'occasion d'utiliser des listes si on veut garder les nombres en mémoire.

Prolongement possible : Moyenne pondérée avec création de deux listes ou d'un tableau à deux dimensions.

Corrections de l'exercice 11 :

Variables:

- N est un nombre.
- S est un nombre
- NB est un nombre.
- I est un nombre.

Entrée

- Demander N.

Traitement

- Donner à S la valeur 0.
- Répéter pour I allant de 1 à N.
 - Demander NB.
 - Donner à S la valeur de S + NB
- Fin Répéter.

Sortie

- Afficher «La somme est : »
- Afficher S

Avec AlgoBox, on obtient :

Présentation de l'algorithme

Algorithme qui permet de calculer la somme (ou la moyenne) de N nombres donnés par l'utilisateur.

Code de l'algorithme

```

▼ VARIABLES
  N EST_DU_TYPE NOMBRE
  S EST_DU_TYPE NOMBRE
  I EST_DU_TYPE NOMBRE
  NB EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  LIRE N
  S PREND_LA_VALEUR 0
  ▼ POUR I ALLANT_DE 1 A N
    DEBUT_POUR
    LIRE NB
    S PREND_LA_VALEUR S+NB
    FIN_POUR
  AFFICHER "La somme est"
  AFFICHER S
FIN_ALGORITHME

```

OPERATIONS AVEC LES LISTES.

Les listes AlgoBox sont des listes numérotées de nombres (il n'y a pas besoin de préciser le nombre d'éléments de la liste).

Si vous sélectionnez pour *la variable* une variable du type LISTE, vous devez indiquer dans le champ *rang du terme de la liste* le numéro du terme de la liste auquel vous voulez affecter une valeur.

Pour utiliser un terme d'une liste dans un calcul, il faut utiliser la syntaxe suivante :

nomliste[rang]

Exemple : moy prend la valeur $(\text{maliste}[1] + \text{maliste}[2] + \text{maliste}[3]) / 3$ (la variable du type NOMBRE moy contiendra la moyenne des trois premiers termes de la liste maliste)

Initialisation d'une liste. Prolongement de l'exercice 11.

Ecrire un algorithme qui demande N nombres et les met dans un tableau.

(Modification de l'exercice 11: on conserve les nombres entrés) ;

Ecrire un algorithme qui permet de déterminer le maximum et le minimum d'une liste de N nombres.

Le prolongement possible est par exemple un algorithme de tri.

Exercice 12 :

Ecrire un algorithme qui simule N lancers d'un dé et calcule la fréquence d'apparition du 6.

On ne connaît pas le nombre d'itérations.

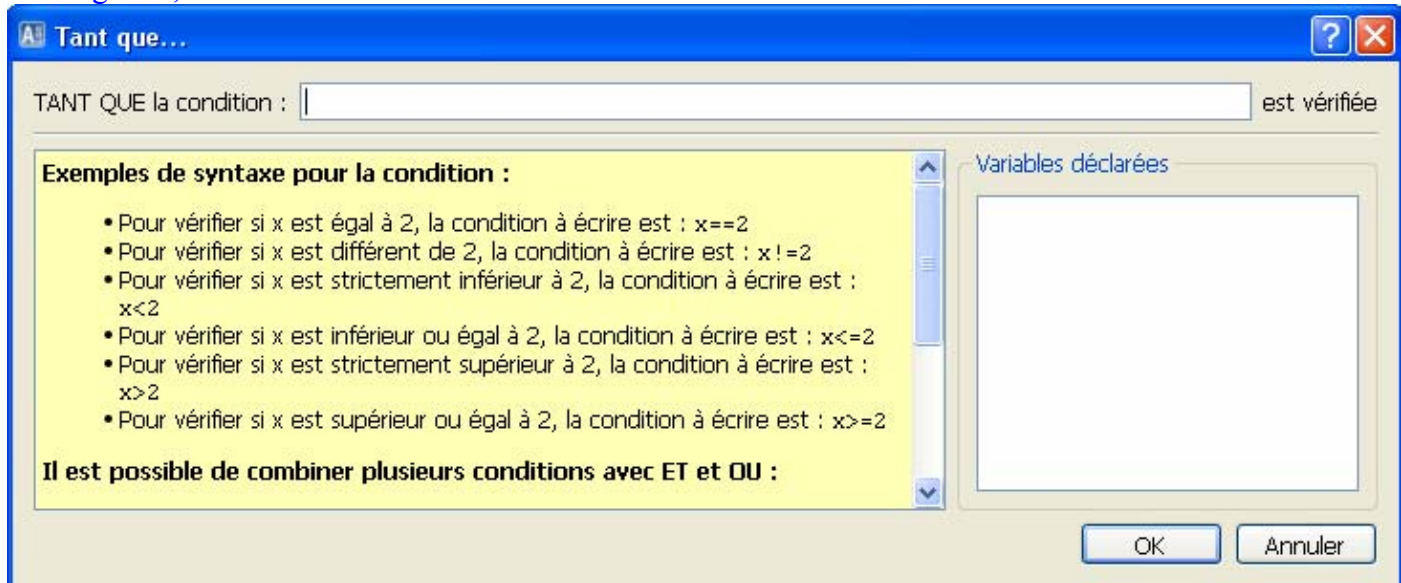
<i>TANT QUE</i>	<i>condition</i>
	<i>instructions.</i>
<i>FIN TANT QUE</i>	

Le bloc instructions est répété tant que la condition est vraie.

Remarques importantes sur la boucle « TANT QUE ».

- La condition doit être initialisée avant le début de la boucle.
- La condition doit être modifiée dans la boucle et doit pouvoir prendre à un moment donné la valeur FAUX car sinon la boucle est infinie (on ne sort jamais de la boucle) et le programme ne s'arrête jamais.
- La boucle peut ne jamais être exécutée, **car condition fausse dès le départ.**

Il peut être intéressant de faire vérifier des algorithmes avec boucles et comprenant des erreurs. Sur Algobox, il faut activer la touche :



Exercice 13 :

Ecrire un algorithme utilisant la boucle « TANT QUE3 qui simule une boucle dont on connaît le nombre de répétitions.

Cet exercice montre qu'on peut se contenter de la boucle conditionnelle dans les algorithmes.

Exercice 14 :

Ecrire un algorithme qui redemande un nombre tant que le nombre donné n'est pas compris dans un intervalle fixé.

Application :

Cet algorithme peut être utilisée pour faire tourner un algorithme tant que l'on répond « OUI » à la question « Voulez vous recommencer ? ».

Exercice 15 : Donné en terminale L.

Soit N un entier naturel non nul. On considère l'algorithme suivant :

<u>Entrée :</u>	Un entier naturel N
<u>Initialisation :</u>	Donner à A la valeur de N Donner à K la valeur de N .
<u>Traitement :</u>	Tant que $K > 2$, répéter la procédure suivante Donner à K la valeur $K - 1$ Donner à A la valeur $A \times K$ Donner à A la valeur $A - 1$ Fin Tant que
<u>Sortie :</u>	Afficher A

1. Pour tout entier naturel N , on note A_N le nombre affiché en sortie de cet algorithme.

a) Calculer A_1, A_2, A_3 et A_4 .

b) Vérifier que $A_5 = 119$.

c) Calculer A_{10} .

2. Pour chacune des propositions suivantes, dire si elle est vraie ou fausse en justifiant la réponse donnée :

a) « A_{11} est un nombre premier pour certaines valeurs de N ».

b) « A_{11} est un nombre premier pour n'importe quelle valeur de N ».

c) « Quel que soit l'entier naturel non nul N , si N est premier, alors A_{11} est premier. »

d) « Il existe A_{11} premier tel que N n'est pas premier ».

3. Etudier la parité des nombres A_N

Exercice 16 : Autre exercice donné en TL.

On considère l'algorithme suivant:

<u>Entrée :</u>	un entier naturel N
<u>Initialisation :</u>	Donner à U la valeur de N
<u>Traitement :</u>	Tant que $U \geq 10$ Donner à U la valeur de $U-10$ fin tant que Donner à A la valeur $(N-U)/10$ Donner à C la valeur $A \times (A+1) \times 100 + 25$
<u>Sortie :</u>	Afficher C .

1. Qu'obtient-on en sortie si on donne en entrée à N la valeur 27 ? Et pour 129 ?

Que remarque-t-on?

2. Soit N un entier dont le chiffre des unités est 5.

a) Faire fonctionner cet algorithme avec plusieurs entiers N et vérifier que la sortie de l'algorithme appliqué à N donne alors le carré de N .

b) Justifier ce résultat.

3. Calculer sans calculatrice le carré de 45 et de 75.

Exercice 17 :

Division euclidienne

Ecrire un algorithme qui donne le quotient et le reste de la division euclidienne d'un entier naturel a par un entier naturel b .

Prolongement : recherche du PGCD de deux entiers naturels

Corrections de l'exercice 17 :

Avec la TI 89, on a :

```
diveucl ()
Prgm
ClrIO
Local a,b,q,r
Disp "Soit a le dividende"
Disp "Soit b le diviseur"
Prompt a,b
If b=0 Then
Disp "Impossible"
Else
intDiv(a,b)→q
mod(a,b)→r
Disp "Quotient",q
Disp "Reste",r:Pause
EndIf
DispHome|
EndPrgm
```

Exercice 18 :

Ecrire un algorithme qui permet d'afficher la liste et la somme des diviseurs d'un entier naturel N.

Exercice 19 :

Les grands parents de Julien déposent à sa naissance en 2010 une somme de 1500 € sur un livret d'épargne. Ce livret rapporte 2,5% par an.

En quelle année, la somme aura-t-elle doublée?

Prolongements possibles:

- ✓ Modifier la somme de départ ou la mettre en donnée à saisir
- ✓ Mettre le taux d'intérêt en donnée à saisir pour pouvoir comparer différents placements possibles

Exercice 20 :

Jeu du « trop grand-trop petit »

Ecrire un algorithme qui choisit un nombre entier aléatoire et vous propose de deviner ce nombre.

L'algorithme affichera le nombre de coups utilisés pour trouver le résultat.

LES FONCTIONS AVEC ALGOBOX.

Tableau de valeurs et représentation graphique d'une fonction.

Etant donnée une fonction f, écrire un algorithme qui permet d'obtenir un tableau de valeurs de f sur un segment [a;b] et d'afficher les points de la courbe correspondants.

Cet énoncé est trop généraliste: il faut préciser par exemple la fonction

Les fonctions dans AlgoBox

Pour utiliser une fonction notée **F1** ; il faut la définir en dehors de l'algorithme. Pour cela :

- Activer l'option **Utiliser une fonction** dans l'onglet « **Utiliser une fonction numérique** » :

The screenshot shows the AlgoBox interface with three tabs: 'Opérations standards', 'Utiliser une fonction numérique', and 'Dessiner dans un repère'. The 'Utiliser une fonction numérique' tab is active. Underneath, there is a checkbox labeled 'Utiliser une fonction' which is checked. Below the checkbox is a section titled 'Définir la fonction' containing a text input field with the label 'F1(x)= '.

Entrer l'expression de $F1(x)$ en fonction de x dans le champ prévue pour cela. (On peut aussi utiliser l'aide, en bas, à droite).

Pour utiliser l'image d'un nombre nb par la fonction F1 dans l'algorithme, il suffit d'utiliser le code : $F1(nb)$

Analyse du problème:

Dans un premier temps, on va demander la valeur de l'abscisse entière du premier point et on va faire la table des valeurs F1 des nombres obtenus avec un pas de 1

Il faut définir la fonction F1

Variables :

- A, N, I des nombres.
- x, y des nombres.

Entrée

- ⋮ Demander A
- ⋮ Demander N

Traitement :

- Répéter pour I allant de 1 à N
 - Donner à x la valeur $x + (I - 1)$
 - Donner à y la valeur $F1(x)$
 - Afficher x
 - Afficher y
 - Placer le point de coordonnées (x ;y).
- Fin Répéter.

Les sorties sont faites lors du traitement.

Ce qui donne avec AlgoBox :

Table des valeurs d'une fonction avec un pas de 1
le nombre de valeurs de départ et la valeur de départ
sont demandés à l'utilisateur.
fenetre graphique , w>-10 et la fonction est :
 $F1(x)=2x^2+3x-4$

Code de l'algorithme

```

VARIABLES
├── A EST_DU_TYPE NOMBRE
├── N EST_DU_TYPE NOMBRE
├── I EST_DU_TYPE NOMBRE
├── x EST_DU_TYPE NOMBRE
└── y EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
├── LIRE A
├── LIRE N
└── POUR I ALLANT_DE 1 A N
    ├── DEBUT_POUR
    ├── x PREND_LA_VALEUR A+I-1
    ├── y PREND_LA_VALEUR F1(x)
    ├── AFFICHER x
    ├── AFFICHER " "
    ├── AFFICHER y
    ├── TRACER_POINT (x,y)
    └── FIN_POUR
FIN_ALGORITHME

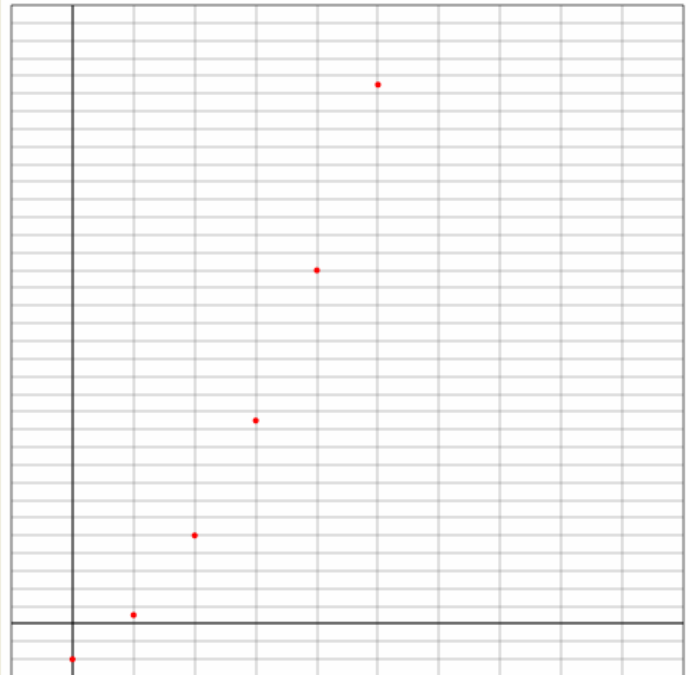
```

PRÉSENTATION DE L'ALGORITHME :

Table des valeurs d'une fonction avec un pas de 1
le nombre de valeurs de départ et la valeur de départ
sont demandés à l'utilisateur.

fenetre graphique , w>-10 et la fonction est :
 $F1(x)=2x^2+3x-4$

GRAPHIQUE :



Xmin: -1 ; Xmax: 10 ; Ymin: -6 ; Ymax: 70 ; GradX: 1 ; GradY: 2

Résultats

```

40
5
61
***Algorithme terminé***

```

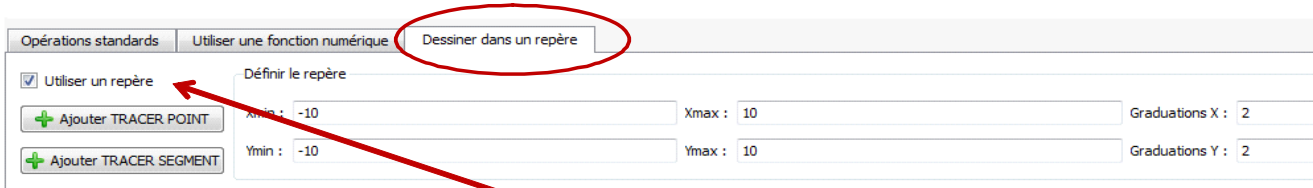
Exercice 21 :Tableau de valeurs et représentation graphique d'une fonction.

Etant donnée une fonction f , écrire un algorithme qui permet d'obtenir un tableau de valeurs de f sur un segment $[a;b]$ avec un pas de 0,1 et d'afficher les points de la courbe correspondants.

Tracer des points et des segments dans un repère

Il faut :

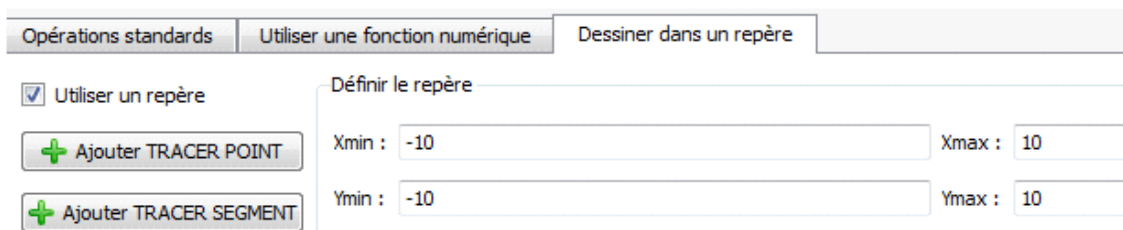
Aller dans l'onglet "Dessiner dans un repère "



- activer 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.

Les boutons **Ajouter TRACER POINT** et **Ajouter TRACER SEGMENT** sont alors activés et peuvent être inclus dans le code de l'algorithme des instructions pour tracer des points et des segments dans ce repère



Ajouter TRACER POINT

Opérations standards Utiliser une fonction numérique Dessiner dans un repère

Utiliser un repère

+ Ajouter TRACER POINT

+ Ajouter TRACER SEGMENT

Définir le repère

Xmin : -10 Xmax : 10

Ymin : -10 Ymax : 10

Si on clique sur ce bouton, la fenêtre suivante apparaît

A Tracer un point

Coordonnées du point :

X : Y :

Couleur du point : Rouge

OK Annuler

Ajouter TRACER SEGMENT

Opérations standards Utiliser une fonction numérique Dessiner dans un repère

Utiliser un repère

+ Ajouter TRACER POINT

+ Ajouter TRACER SEGMENT

Définir le repère

Xmin : -10 Xmax : 10

Ymin : -10 Ymax : 10

Si on clique sur ce bouton, la fenêtre suivante apparaît

A Tracer un segment

Coordonnées du point de départ :

X : Y :

Coordonnées du point d'arrivée :

X : Y :

Couleur du segment : Rouge

OK Annuler

Prolongements:***Tableau de valeurs et représentation graphique d'une fonction.***

a) Le pas est choisi par l'utilisateur

b) Ecrire un algorithme permettant d'encadrer, avec une précision donnée, la valeur qui annule une fonction continue et strictement monotone sur un segment $[a;b]$: méthode du balayage.**Exercice 22 :****On jette un dé autant de fois qu'il faut pour obtenir un 6****Ecrire un algorithme qui compte le nombre de lancers nécessaires pour obtenir un 6 pour la première fois.****Prolongement:***Adapter l'algorithme précédent afin de calculer le nombre moyen de lancers pour obtenir un premier 6*Autre exercice du même type :**Etude du lancer de deux dés et de la somme obtenue****On veut étudier le nombre de lancers nécessaires pour obtenir une somme donnée****Exercice 23 :****Marche aléatoire sur une droite**

Une puce se déplace sur un axe gradué : elle part de l'origine et, à chaque saut, elle se déplace d'une unité vers la droite ou vers la gauche, de manière aléatoire.

Pour choisir le sens (gauche, droite) de chaque saut, on lance une pièce de monnaie équilibrée : si l'on obtient PILE (codé P) la puce se déplace à gauche, si l'on obtient FACE (codé F) la puce part à droite.

Ecrire un algorithme qui permette de simuler N promenades de 4 sauts et de dénombrer le nombre de promenades qui se terminent à la distance 0, 2 ou 4 de l'origine.

Exercice 24 : Un exemple de TP**Passage en caisse****PREMIERE PARTIE****Lors du passage à la caisse, on veut calculer le total à payer, les prix des articles sont rentrés un par un.**

On veut faire l'addition des nombres rentrés dans la machine.

Chaque prix saisi est suivi de la touche entrée (on ne se sert pas de la touche "+").

Pour signaler que la saisie est terminée, on rentre le nombre zéro.

La machine doit fournir alors le total à payer.

Rédiger un algorithme qui permettra ce travail**DEUXIEME PARTIE****Nombre d'articles, prix moyen**

Compléter l'algorithme précédent pour qu'il donne en plus le nombre d'articles achetés et le prix moyen d'un article.

TROISIEME PARTIE**Le magasin fête son anniversaire: offre promotionnelle! Pendant la durée de la journée anniversaire, le magasin offre une remise de 15% sur l'article le plus cher.**

Compléter l'algorithme précédent de façon à ce qu'il calcule et affiche la réduction ainsi que le nouveau total à payer.

Exercice complémentaire*Pour manipuler les chaînes de caractères***Ecrire un algorithme qui teste si un mot ou une phrase est un palindrome****Exercice 25 :**

Parité d'un nombre entier positif :

Structure de l'algorithme :

Si n a pour reste 0 dans la division par 2 alors

Afficher « nombre pair »

Sinon

Afficher « nombre impair »

FinSi

Ce qui donne avec Albox :

Présentation de l'algorithme

Parité d'un entier positif

Code de l'algorithme

```

VARIABLES
  n EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
  LIRE n
  AFFICHER n
  SI (n%2==0) ALORS
    DEBUT_SI
    AFFICHER "L'entier est pair"
    FIN_SI
  SINON
    DEBUT_SINON
    AFFICHER "L'entier est impair"
    FIN_SINON
FIN_ALGORITHME

```

Parité d'un entier positif

CODE DE L'ALGORITHME :

```

1  VARIABLES
2    n EST_DU_TYPE NOMBRE
3  DEBUT_ALGORITHME
4    LIRE n
5    AFFICHER n
6    SI (n%2==0) ALORS
7      DEBUT_SI
8      AFFICHER "L'entier est pair"

```

Résultats

```

***Algorithme lancé***
124587
L'entier est impair
***Algorithme terminé***

```

Exercice 26 :Algorithme donnant le résultat en augmentant de 4 % un certain montant M en euros, pour rappel le résultat estalors $M \times \left(1 + \frac{4}{100}\right) = M \times 1,04$.Par exemple, on dépose sur un livret un montant M et il augment chaque année de 4 %. On veut connaître le montant obtenu au bout de 10 ans.

On répète pour cela 10 fois de suite l'augmentation de 4 %, et on obtient l'algorithme suivant :

Variables :• M, I des nombres.**Entrée**; Demander M **Traitement :**Pour I allant de 1 à 10 faire• M prend la valeur $M \times 1,04$

• Pour.

Sortie :Afficher M .

Avec AlgoBox , on a :

Présentation de l'algorithme

Augmentation de 4 %

Code de l'algorithme

```

VARIABLES
├── M EST_DU_TYPE NOMBRE
├── I EST_DU_TYPE NOMBRE
└── DEBUT_ALGORITHME
    ├── LIRE M
    ├── AFFICHER M
    ├── POUR I ALLANT_DE 1 A 10
    │   ├── DEBUT_POUR
    │   ├── M PREND_LA_VALEUR M*1.04
    │   ├── FIN_POUR
    │   └── AFFICHER M
    └── FIN_ALGORITHME
  
```

PRÉSENTATION DE L'ALGORITHME :

Augmentation de 4 %

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  M EST_DU_TYPE NOMBRE
3  I EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5  LIRE M
6  AFFICHER M
7  POUR I ALLANT_DE 1 A 10
8  DEBUT_POUR
  
```

Résultats

```

***algorithme lancé***
1000
1480.2443
***algorithme terminé***
  
```

Exercice 27 :

Programmer les puissances de 2 successives, sans utiliser la fonction puissance.

Présentation de l'algorithme

Puissances de 2

Code de l'algorithme

```

VARIABLES
├── a EST_DU_TYPE NOMBRE
├── n EST_DU_TYPE NOMBRE
├── a_puissance_n EST_DU_TYPE NOMBRE
└── DEBUT_ALGORITHME
    ├── a PREND_LA_VALEUR 2
    ├── n PREND_LA_VALEUR 0
    ├── a_puissance_n PREND_LA_VALEUR 1
    ├── POUR n ALLANT_DE 1 A 100
    │   ├── DEBUT_POUR
    │   ├── n PREND_LA_VALEUR n+1
    │   ├── a_puissance_n PREND_LA_VALEUR a_puissance_n*a
    │   ├── AFFICHER a_puissance_n
    │   └── FIN_POUR
    └── FIN_ALGORITHME
  
```

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  a EST_DU_TYPE NOMBRE
3  n EST_DU_TYPE NOMBRE
4  a_puissance_n EST_DU_TYPE NOMBRE
5  DEBUT_ALGORITHME
6  a PREND_LA_VALEUR 2
7  n PREND_LA_VALEUR 0
8  a_puissance_n PREND_LA_VALEUR 1
  
```

Résultats

```

32
64
128
256
512
1024
2048
4096
  
```

Exercice 28 :

Valeur absolue d'un nombre

Présentation de l'algorithme

Valeur absolue d'un nombre

Code de l'algorithme

```

VARIABLES
├── a EST_DU_TYPE NOMBRE
├── Valeur_absolue EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
├── a PREND_LA_VALEUR -84.2
├── SI (a>0 ou a==0) ALORS
│   ├── DEBUT_SI
│   ├── Valeur_absolue PREND_LA_VALEUR a
│   └── FIN_SI
├── SINON
│   ├── DEBUT_SINON
│   ├── Valeur_absolue PREND_LA_VALEUR a*(-1)
│   └── FIN_SINON
├── AFFICHER "La valeur absolue de a est"
├── AFFICHER Valeur_absolue
FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

Valeur absolue d'un nombre

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      a EST_DU_TYPE NOMBRE
3      Valeur_absolue EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5      a PREND_LA_VALEUR -84.2
6      SI (a>0 ou a==0) ALORS
7          DEBUT_SI
8          Valeur_absolue PREND_LA_VALEUR a

```

Résultats

```

***Algorithme lancé***
La valeur absolue de a est
84.2

***Algorithme terminé***

```

Exercice 29 :

Equations du second degré dans l'ensemble des nombres réels, c'est-à-dire sans les solutions complexes.

Présentation de l'algorithme

Equation du second degré sans les solutions complexes.

Code de l'algorithme

```

VARIABLES
├── a EST_DU_TYPE NOMBRE
├── b EST_DU_TYPE NOMBRE
├── c EST_DU_TYPE NOMBRE
├── d EST_DU_TYPE NOMBRE
├── x1 EST_DU_TYPE NOMBRE
├── x2 EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
├── LIRE a
├── LIRE b
├── LIRE c
├── d PREND_LA_VALEUR b*b-4*a*c
├── SI (d<0) ALORS
│   ├── DEBUT_SI
│   ├── AFFICHER "Pas de racine réelle"
│   └── FIN_SI
├── SINON
│   ├── DEBUT_SINON
│   ├── x1 PREND_LA_VALEUR (-b-sqrt(d))/(2*a)
│   ├── AFFICHER "x1="
│   ├── AFFICHER x1
│   ├── x2 PREND_LA_VALEUR (-b+sqrt(d))/(2*a)
│   ├── AFFICHER "x2="
│   ├── AFFICHER x2
│   └── FIN_SINON
FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

Equation du second degré sans les solutions complexes.

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      a EST_DU_TYPE NOMBRE
3      b EST_DU_TYPE NOMBRE
4      c EST_DU_TYPE NOMBRE
5      d EST_DU_TYPE NOMBRE
6      x1 EST_DU_TYPE NOMBRE
7      x2 EST_DU_TYPE NOMBRE
8  DEBUT_ALGORITHME

```

Résultats

```

***Algorithme lancé***
x1=-1.5
x2=-1

***Algorithme terminé***

```

Exercice 30:

Le calcul d'une somme, par exemple le calcul de la somme des 100 premiers inverses des entiers naturels.

Présentation de l'algorithme

Apprendre à calculer une somme.
Ici, on calcule la somme des 100 premiers inverses des entiers naturels.

Code de l'algorithme

```

▼ VARIABLES
  |
  |— n EST_DU_TYPE NOMBRE
  |— somme EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  |
  |— somme PREND_LA_VALEUR 0
  |— POUR n ALLANT_DE 1 A 100
  |   |
  |   |— DEBUT_POUR
  |   |   |
  |   |   |— somme PREND_LA_VALEUR somme+1/n
  |   |   |— FIN_POUR
  |   |— AFFICHER somme
  |— FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

Apprendre à calculer une somme.
Ici, on calcule la somme des 100 premiers inverses des entiers naturels.

CODE DE L'ALGORITHME :

```

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

```

Résultats

```

***Algorithme lancé***
5.1873775
***Algorithme terminé***

```

Exercice 31:

Calcul itératif, le nombre de boucles étant donné.

Présentation de l'algorithme

Calcul itératif, le nombre de boucles étant donné.

Code de l'algorithme

```

▼ VARIABLES
  |
  |— n EST_DU_TYPE NOMBRE
  |— inverse_n EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  |
  |— POUR n ALLANT_DE 1 A 100
  |   |
  |   |— DEBUT_POUR
  |   |   |
  |   |   |— inverse_n PREND_LA_VALEUR 1/n
  |   |   |— AFFICHER inverse_n
  |   |   |— FIN_POUR
  |— FIN_ALGORITHME

```

Exercice 32 :

La suite de Fibonacci, « sans le dire, uniquement pour faire une boucle avec un arrêt conditionnel ».

Mathématicien italien, né et mort à Pise. Connu aussi sous le nom de Léonard de Pise, Leonardo Fibonacci fut éduqué en Afrique du Nord, où son père, marchand de la ville de Pise (l'un des plus grands centres commerciaux d'Italie, à l'époque, au même rang que Venise et Gênes), dirigeait une sorte de comptoir ; c'est ainsi qu'il eut l'occasion d'étudier les travaux algébriques d'al-Khuwarizmi. Par la suite, Fibonacci voyagea dans tout le monde méditerranéen, rencontrant de nombreux scientifiques et prenant connaissance des différents systèmes de calcul en usage chez les marchands de l'époque. De toutes les méthodes de calcul, il jugea celle des Arabes la plus avancée. Aussi, de retour à Pise, il publie en 1202 un ouvrage, Liber abbaci, où, le comparant au système romain, il expose le système de numération indo-arabe. Il est le premier grand mathématicien à l'adopter et à le vulgariser auprès des scientifiques. Son ouvrage contient également la plupart des résultats connus des Arabes en algèbre et en arithmétique (racines carrées, racines cubiques, équations du premier et du second degré). En 1220, il publie Practica geometriae, qui recense toutes les connaissances de l'époque en géométrie et en trigonométrie (écrits d'Euclide et des autres mathématiciens grecs, transmis par des manuscrits arabes ou traduits par des Italiens) ; en particulier, l'ouvrage contient la formule de Héron donnant l'aire du triangle en fonction des longueurs des trois côtés. Mais Fibonacci ne se contenta pas de faire connaître les travaux des Anciens et d'être à l'origine de la renaissance des études mathématiques en Occident, il poursuivit aussi ses propres travaux. Sa réputation scientifique était telle que l'empereur Frédéric II s'arrêta à Pise pour le voir et lui poser des « colles » (cette sorte de compétition entre scientifiques devait se développer au XVIe et au XVIIe siècle). La résolution de ces problèmes (les plus célèbres étant : trouver un nombre x tel que $x^2 + 5$ et $x^2 - 5$ soient tous deux des carrés ; résoudre l'équation du troisième degré $x^3 + 2x^2 + 10x = 20$) ainsi que la résolution d'autres problèmes de même nature sont contenues dans Liber quadratorum (1225). Notons enfin que Fibonacci est à l'origine d'une suite récurrente qui porte son nom, suite dont les deux premiers termes sont 0 et 1 et dont le terme d'ordre $n + 1$ est égal à la somme des deux termes d'ordre n et $n - 1$ pour tout n supérieur ou égal à 2.

Présentation de l'algorithme

Fibonacci sans le dire pour faire
une boucle avec arrêt conditionnel.

Code de l'algorithme

```

▼ VARIABLES
  a EST_DU_TYPE NOMBRE
  b EST_DU_TYPE NOMBRE
  f EST_DU_TYPE NOMBRE
  n EST_DU_TYPE NOMBRE
▼ DEBUT_ALGORITHME
  a PREND_LA_VALEUR 1
  b PREND_LA_VALEUR 1
  n PREND_LA_VALEUR 1
  ▼ TANT_QUE (f<1000) FAIRE
    DEBUT_TANT_QUE
    AFFICHER "a = "
    AFFICHER a
    AFFICHER " ; b = "
    AFFICHER b
    f PREND_LA_VALEUR a+b
    a PREND_LA_VALEUR b
    b PREND_LA_VALEUR f
    AFFICHER "n = "
    AFFICHER n
    AFFICHER " ; f = "
    AFFICHER f
    n PREND_LA_VALEUR n+1
    FIN_TANT_QUE
  FIN_ALGORITHME

```

PRÉSENTATION DE L'ALGORITHME :

Fibonacci sans le dire pour faire
une boucle avec arrêt conditionnel.

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  a EST_DU_TYPE NOMBRE
3  b EST_DU_TYPE NOMBRE
4  f EST_DU_TYPE NOMBRE
5  n EST_DU_TYPE NOMBRE
6  DEBUT_ALGORITHME
7  a PREND_LA_VALEUR 1

```

Résultats

```

a = 233 ; b = 377
n = 13 ; f = 610
a = 377 ; b = 610
n = 14 ; f = 987
a = 610 ; b = 987
n = 15 ; f = 1597

```

Algorithme terminé

Exercice 33 :

Les variables sont des tiroirs dans lesquels on range des nombres. Il faut suivre a, b et c.

Tiroirs dans lequel on range des nombres.

Code de l'algorithme

```

VARIABLES
├── a EST_DU_TYPE NOMBRE
├── b EST_DU_TYPE NOMBRE
├── c EST_DU_TYPE NOMBRE
└── DEBUT_ALGORITHME
    ├── a PREND_LA_VALEUR 3
    ├── b PREND_LA_VALEUR a+2
    ├── a PREND_LA_VALEUR b
    ├── b PREND_LA_VALEUR c+a
    ├── AFFICHER "Que vaut b?"
    ├── AFFICHER b
    └── FIN_ALGORITHME
  
```

Exercice 34:

PRÉSENTATION DE L'ALGORITHME :

Tiroirs dans lequel on range des nombres.

CODE DE L'ALGORITHME :

```

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 3
7  b PREND_LA_VALEUR a+2
  
```

Résultats

```

***Algorithme lancé***
Que vaut b?
5
***Algorithme terminé***
  
```

Suite géométrique de raison $4/3$, en première :

Présentation de l'algorithme

Suite géométrique de raison $4/3$

Code de l'algorithme

```

VARIABLES
├── a EST_DU_TYPE NOMBRE
├── compteur EST_DU_TYPE NOMBRE
└── DEBUT_ALGORITHME
    ├── AFFICHER "Initialisation de a :"
    ├── LIRE a
    ├── AFFICHER a
    ├── compteur PREND_LA_VALEUR 0
    └── TANT_QUE (a <= 100) FAIRE
        ├── DEBUT_TANT_QUE
        ├── a PREND_LA_VALEUR a*4/3
        ├── compteur PREND_LA_VALEUR compteur + 1
        ├── AFFICHER compteur
        ├── AFFICHER a
        └── FIN_TANT_QUE
    └── FIN_ALGORITHME
  
```

PRÉSENTATION DE L'ALGORITHME :

Suite géométrique de raison $4/3$

CODE DE L'ALGORITHME :

```

1  VARIABLES
2  a EST_DU_TYPE NOMBRE
3  compteur EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5  AFFICHER "Initialisation de a :"
6  LIRE a
7  AFFICHER a
8  compteur PREND_LA_VALEUR 0
  
```

Résultats

```

5
63.209877
6
84.279835
7
112.37311
***Algorithme terminé***
  
```

Exercice 35 :

Gestion de la valeur interdite de la fonction $x \xrightarrow{f} \frac{1}{x-3}$

Présentation de l'algorithme

gestion de la valeur interdite
de $f(x) = 1 / (x-3)$

Code de l'algorithme

```

VARIABLES
├── x EST_DU_TYPE NOMBRE
└── y EST_DU_TYPE NOMBRE
DEBUT_ALGORITHME
├── LIRE x
├── SI (x==3) ALORS
│   ├── DEBUT_SI
│   ├── AFFICHER "3 n'a pas d'image"
│   └── FIN_SI
└── SINON
    ├── DEBUT_SINON
    ├── y PREND_LA_VALEUR 1/(x-3)
    ├── AFFICHER y
    └── FIN_SINON
FIN_ALGORITHME
  
```

PRÉSENTATION DE L'ALGORITHME :

gestion de la valeur interdite
de $f(x) = 1 / (x-3)$

CODE DE L'ALGORITHME :

```

1  VARIABLES
2      x EST_DU_TYPE NOMBRE
3      y EST_DU_TYPE NOMBRE
4  DEBUT_ALGORITHME
5      LIRE x
6      SI (x==3) ALORS
7          DEBUT_SI
  
```

Résultats

```

***Algorithme lancé***
3 n'a pas d'image

***Algorithme terminé***
  
```