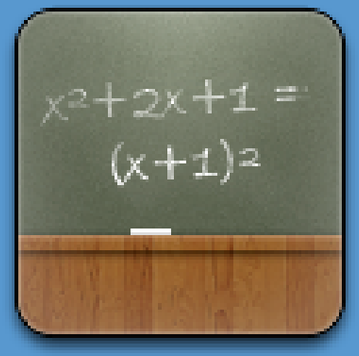
Programmation



Cursus

1

Level

1

Algorithmie

Module 1 Partie 1

Objectifs

- Aborder la programmation. Présentation d'un langage universel.
- Organiser un programme.
- Utiliser des données élémentaires.
- Structurer les étapes de résolution d'un problème.

Sommaire

- Présentation
- Structure de données.
- Structure algorithmique.
- Structure de programme.

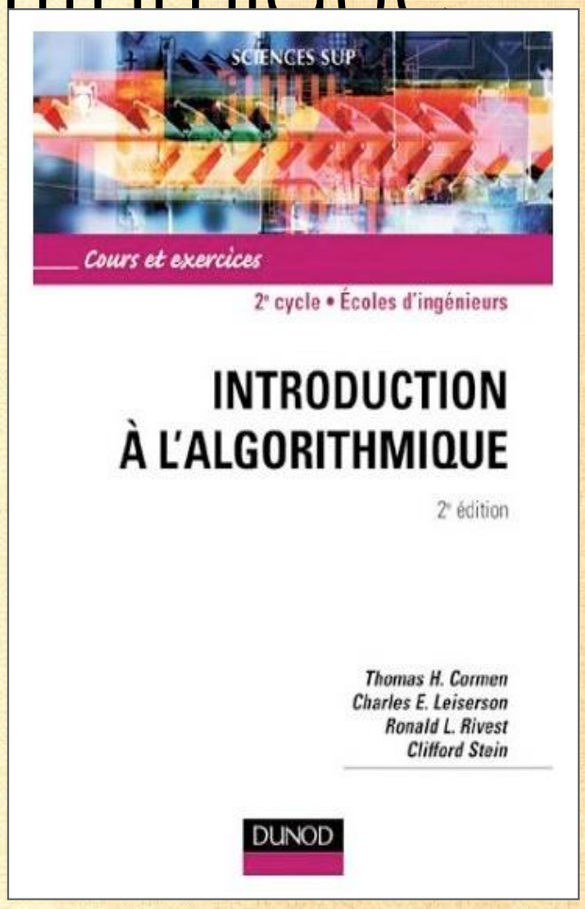
Références Internet

Références Bibliographiques

Introduction à l'algorithmique :

Cours et exercices

de Cormen, Leiserson, Rivest

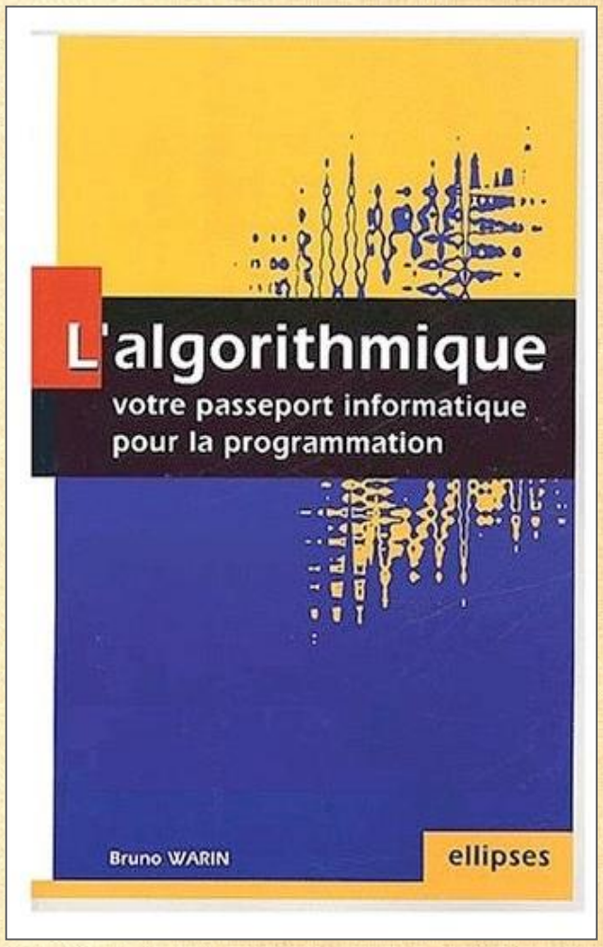


Références Bibliographiques

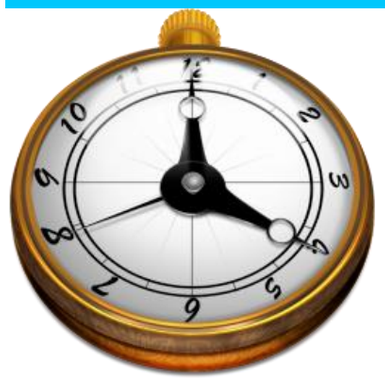
L'algorithmique :

Votre passeport informatique pour la programmation

de Bruno Warin



Liens pédagogiques



Pré-requis:

Historique et modèle de Von Neumann

Nécessaire pour:

Réaliser tout programme.

Algorithmie
Partie N°1
Notions de base

Algorithmie
Notions de base

- Définition
- Méthodologie
- Variables
- Séquence
- Algorithme

Citation

« A l'origine de toute erreur attribuée à l'ordinateur, vous trouverez au moins deux erreurs humaines. Dont celle consistant à attribuer l'erreur à l'ordinateur. »

Définition

Algorithmique:

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

(sources : E. Universalis).

Définition

Algorithmie:

Suite d'opérations nécessaires et suffisantes à l'accomplissement d'une tâche.

★Algorithme d'assemblage d'un produit fini

★Algorithme d'Euclide (calcul du PGCD)

★Algorithme de résolution d'une équation …

Algorithmie> Présentation
Objectif

- Faire réaliser une tâche particulière par une machine
- Disposer d'une écriture simplifiée proche d'un langage informatique (bas niveau, haut niveau)
- Disposer d'un langage intermédiaire entre le langage naturel et un langage informatique évolué
- Disposer d'un langage universel, commun à toutes les publications dans le domaine
- Modéliser un problème sous une forme écrite combinant des séquences d'opérations à réaliser, ainsi que des ruptures de séquences, voire des imbrications complexes de ces opérations

Algorithmie> Présentation
Nécessité

- Les langages de haut niveau évoluent très vite, et nous ne connaissons pas la syntaxe et les particularités des futurs langages, l'algorithmique a traversé 40 ans de langages de haut niveau tout en s'adaptant aux nouveautés

- L'écriture algorithmique permet de prendre de la distance face aux spécificités des langages évolués pour lesquels la compréhension ainsi que la construction de nouveaux programmes se sont considérablement alourdis

- La validation des structures algorithmiques est bien plus aisée que la validation de langages évolués

- Le travail en équipe nécessite un langage commun, puis une répartition suivant le langage évolué approprié aux spécificités du moment

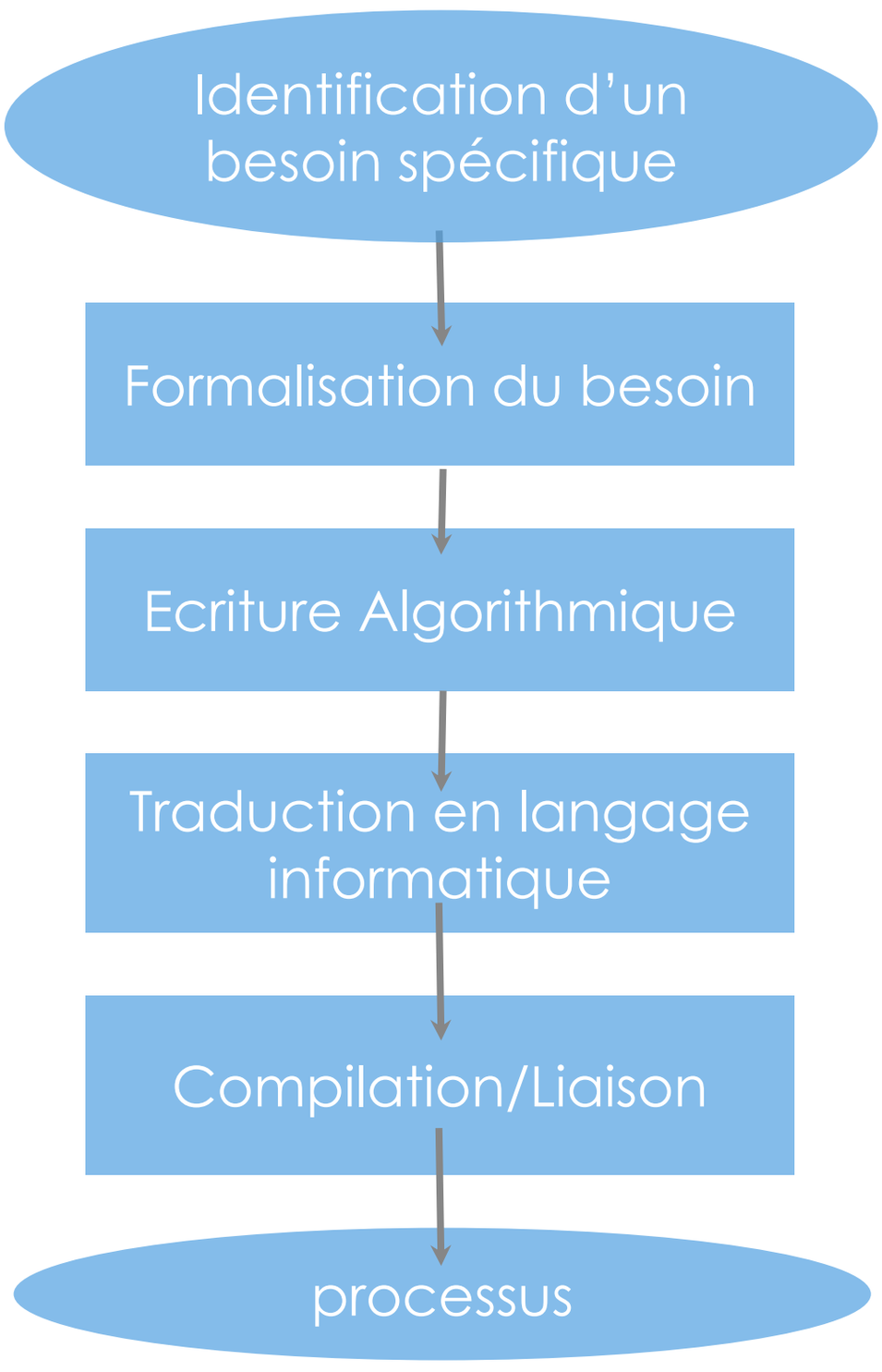
Algorithmie> Présentation
Historique

- Science très ancienne visant à transmettre des moyens efficaces pour obtenir des résultats en partant d'éléments donnés

- Désigne, par la suite, tout procédé de calcul systématique

- En informatique moderne : procédé automatique et effectif, comportant une description (finie) des entrées, des sorties, et des tâches élémentaires à effectuer

Organigramme



PROCEDURE de REALISATION d'un PROGRAMME INFORMATIQUE :

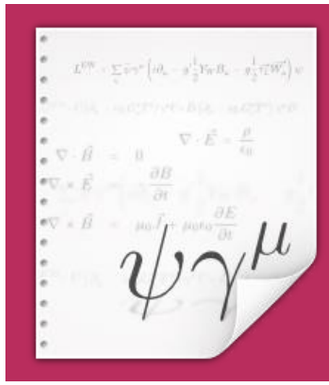
L'écriture algorithmique est une phase intermédiaire et indispensable pour réaliser un programme.

La qualité du développement final dépend aussi de cette phase cruciale.

La suppression de cette phase, contrairement à une idée courante de "gain de temps" qui en découlerait, est généralement la cause de retard de livraison finale du produit logiciel, et parfois à l'origine de l'accroissement de la difficulté de résolution d'un programme complexe.

Algorithmie> Présentation
Notion de variable

Vous connaissez déjà la notion de variables en mathématiques.
Elle diffère cependant de la notion informatique.
Reprenons une équation simple:



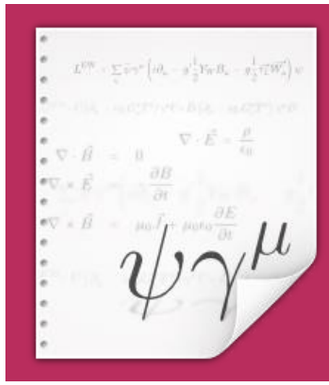
Formule

$$\left(\frac{1}{49}x^2\sqrt{\frac{||x|-3|}{|x|-3}}+\frac{1}{9}y^2\sqrt{\frac{\left|y+\frac{3}{7}\sqrt{33}\right|}{y+\frac{3}{7}\sqrt{33}}}-1\right)\left(\frac{1}{2}|x|-\frac{1}{112}\left(3\sqrt{33}-7\right)x^2-3\right.$$

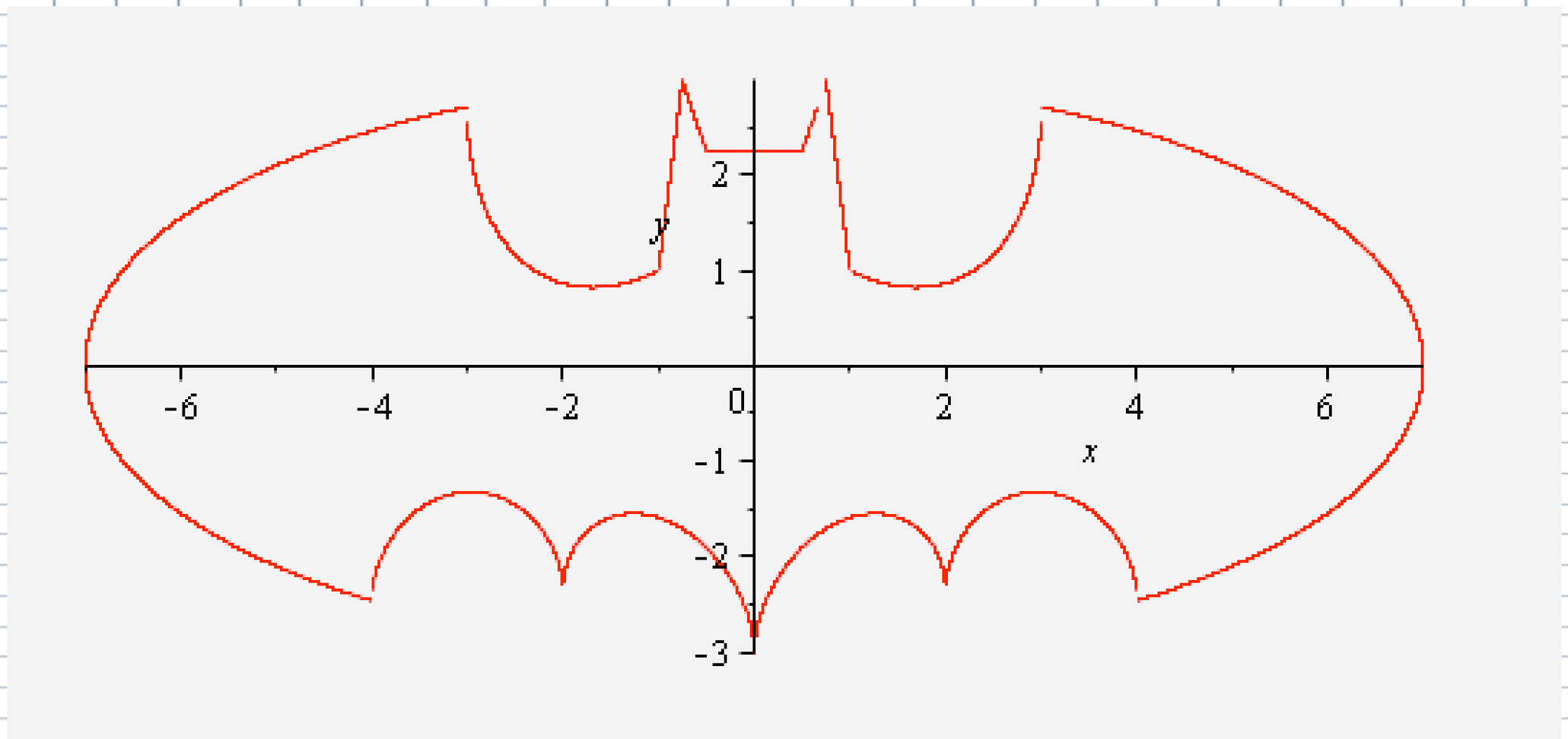
$$\left.+\sqrt{1-(||x|-2|-1)^2}-y\right)\left(9\sqrt{\frac{|(|x|-1)\,(|x|-0.75)|}{(1-|x|)\,(|x|-0.75)}}-8|x|-y\right)\left(3|x|\right.$$

$$\left.+0.75\sqrt{\frac{|(|x|-0.75)\,(|x|-0.5)|}{(0.75-|x|)\,(|x|-0.5)}}-y\right)\left(2.25\sqrt{\frac{|(x-0.5)\,(x+0.5)|}{(0.5-x)\,(x+0.5)}}-y\right)\left(\frac{6}{7}\sqrt{10}\right.$$

$$\left.+(1.5-0.5|x|)\sqrt{\frac{||x|-1|}{|x|-1}}-\frac{3}{7}\sqrt{10}\sqrt{4-(|x|-1)^2}-y\right)$$

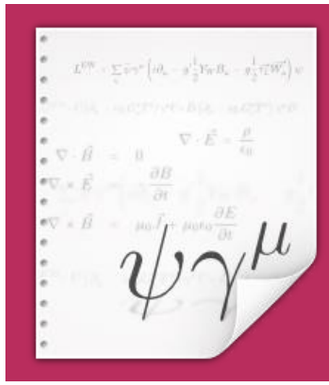


Formule



Algorithmie> Présentation
Notion de variable

Dans le modèle précédent, x peut prendre toutes les valeurs appartenant à $\mathbb{R}$.
Une variable peut aussi justifier ou non une équation.



Formule

$X+2=3$

$X=Y+2$

$Y=3$

$2x=3x$

$X^{2=}-2$

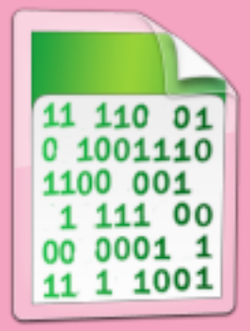
$X/X=0$

Algorithmie> Présentation
Notion de variables

Dans un programme informatique, on va avoir en permanence besoin de stocker provisoirement des valeurs. Il peut s'agir de données issues du disque dur, fournies par l'utilisateur (frappées au clavier), une manette, une balance. Il peut aussi s'agir de résultats obtenus par le programme, intermédiaires ou définitifs. Ces données peuvent être de plusieurs types : des nombres, du texte, une image, etc. Toujours est-il que dès que l'on a besoin de stocker une information au cours d'un programme, on utilise une variable.

Pour employer une image, une variable est une boîte, que le programme (l'ordinateur) va repérer par une étiquette. Pour avoir accès au contenu de la boîte, il suffit de la désigner par son étiquette.

Les langages informatiques plus évolués (ce sont ceux que presque tout le monde emploie) se chargent précisément, entre autres rôles, d'épargner au programmeur la gestion fastidieuse des emplacements mémoire et de leurs adresses.



Code


```
Data Segment
    Message DB "Bonjour, le monde !$"
Data EndS
Code Segment
    Assume Cs : Code, Ds : Data
Main Proc
    Mov Ah, 09h
    Mov Dx, Offset Message
    Int 21h
    Mov Ah, 0C00h
    Int 21h
Main EndP
Code EndS
End Main
```

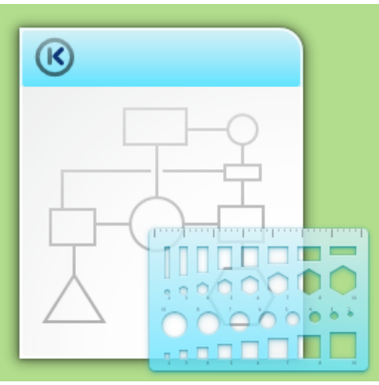
Algorithmie> Présentation
Notion de variables

La place étant limitée dans l'ordinateur, il convient de «dimensionner» la taille des «boîtes». En prenant des grandes «boîtes», nous sommes sûr de pouvoir y stocker ce que l'on veut mais le nombre de boîte que l'on peut stocker diminue. Si on prend des «boîtes» plus petites on augmente la capacité de stockage mais une «variable» sera peut-être trop grande pour y être rangée.

Il est donc évident que pour manipuler une variable elle doit d'abord être définie.

Les opérations élémentaires possibles sur les variables sont:

- Déclaration
- Affectation
- Utilisation
- Ré-affectation



Déclaration

Nom attribué à la variable déclarée.

nom de la variable

nom_de_la_variable :<TYPE>

type

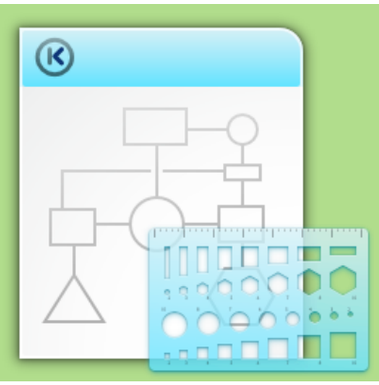
Le type est toujours précédé du signe :

Algorithmie> Présentation
Notion de variables

Nous verrons par la suite le type de variable dans le détail. Vous avez déjà vu en mathématiques les ensembles $\mathbb{R}$, $\mathbb{R}^+$,$\mathbb{N}$, $\mathbb{N}^+$

Les types élémentaires possibles sont:

- Les entiers
- Les réels
- Les caractères
- Les chaînes de caractères
- Les booléens



Déclaration

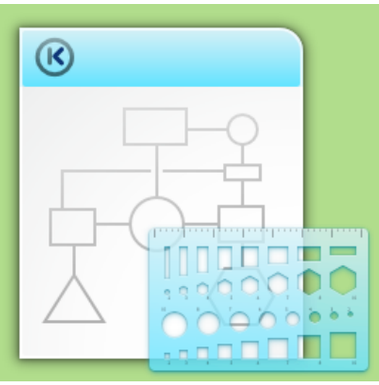
Nom attribué à la variable déclarée.

nom de la variable

note_sur_20 : ENTIER

type

Le type est toujours précédé du signe :



Déclaration

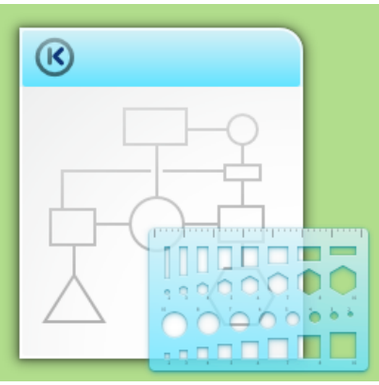
Nom attribué aux variables déclarées.

nom des variables

nom_var1, var2, var3 :<TYPE>

type

Le type est toujours précédé du signe :



Affectation

Nom de la variable affectée

Valeur affectée à la donnée

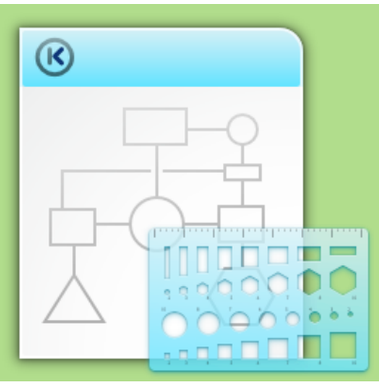
nom de la variable

valeur

nom_de_la_variable ← valeur

affectation

Symbole d'affectation



Affectation

Nom de la variable affectée

Valeur affectée à la donnée

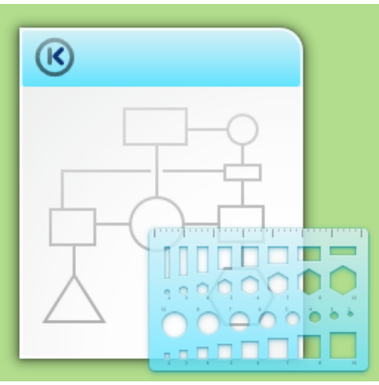
nom de la variable

valeur

note_sur_20 ← 15

affectation

Symbole d'affectation



Affectation

Nom de la variable affectée

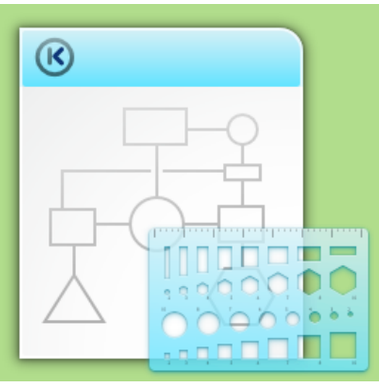
Nom de la variable copiée

nom de la variable **nom d'une autre variable**

nom_de_la_variable ← nom_variable2

affectation

Symbole d'affectation



Affectation

Nom de la variable affectée

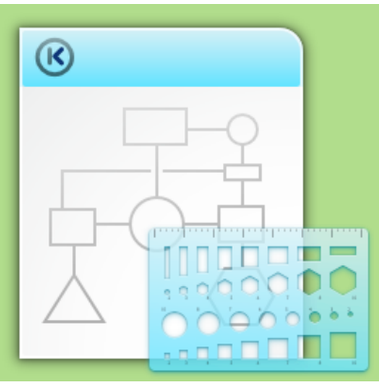
Nom de la variable copiée

nom de la variable — **nom d'une autre variable**

note_sur_100 ← **note_sur_20 * 5**

affectation

Symbole d'affectation



Réaffectation

Nom de la variable affectée

Nom de la variable copiée

nom de la variable — **nom de la même variable**

note_sur_100 ← **note_sur_100 + 5**

affectation

Symbole d'affectation

Algorithmie> Présentation
Notion de variables

Soit x, un entier
x : entier

```
x ← 3
x ← x + 4
```

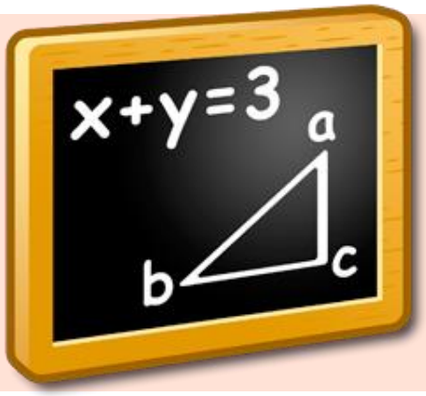
mais en mathématiques:
`soit x ∈ ℕ`
`x = x + 4` n'est pas possible.

En mathématiques, une « variable » est généralement une inconnue, qui recouvre un nombre non précisé de valeurs. Lorsque j'écris :
$y = 3x + 2$
les « variables » x et y satisfaisant à l'équation existent en nombre infini (graphiquement, l'ensemble des solutions à cette équation dessine une droite). Lorsque j'écris :
$ax^2 + bx + c = 0$
la « variable » x désigne les solutions à cette équation, c'est-à-dire zéro, une ou deux valeurs à la fois…

Algorithmie> Présentation
Notion de variables

En informatique, une variable possède à un moment donné une valeur et une seule. A la rigueur, elle peut ne pas avoir de valeur du tout (une fois qu'elle a été déclarée, et tant qu'on ne l'a pas affectée. A signaler que dans certains langages, les variables non encore affectées sont considérées comme valant automatiquement zéro). Mais ce qui est important, c'est que cette valeur justement, ne « varie » pas à proprement parler. Du moins ne varie-t-elle que lorsqu'elle est l'objet d'une instruction d'affectation.

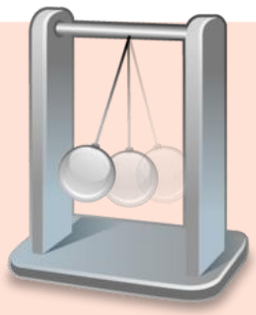
La deuxième remarque concerne le signe de l'affectation. En algorithmique, comme on l'a vu, c'est le signe ←. Mais en pratique, la quasi totalité des langages emploient le signe égal. Et là, pour les débutants, la confusion avec les maths est également facile. En maths, A = B et B = A sont deux propositions strictement équivalentes. En informatique, absolument pas, puisque cela revient à écrire A ← B et B ← A, deux choses bien différentes.



Exercice

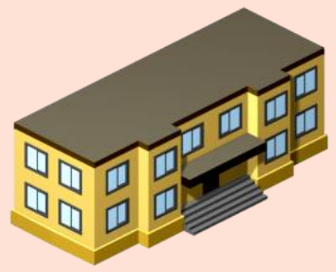
Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
VAR: a, b: ENTIER
DEBUT
   a ← 1
   b ← a + 3
   a ← 3
FIN
```

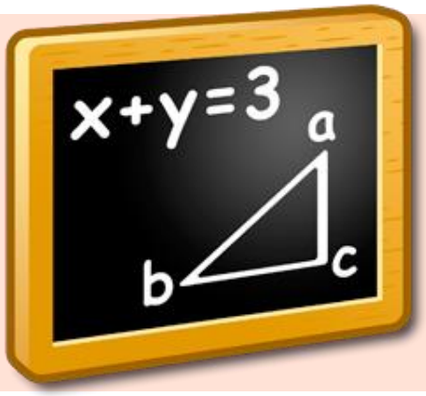


1 min

solo



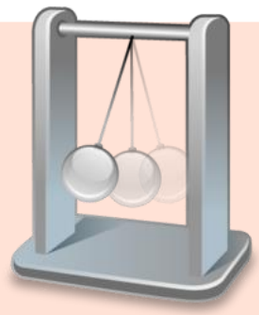
Live



Exercice

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

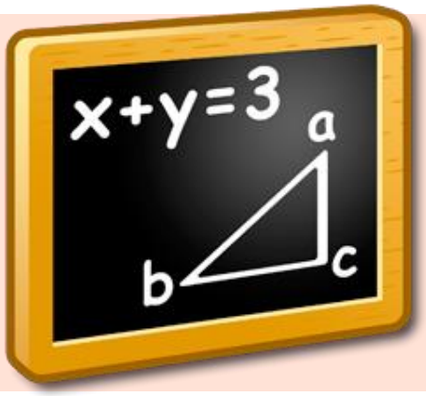
```
VAR: a, b, c: ENTIER
DEBUT
   a ← 5
   b ← 3
   c ← a + b
   a ← 2
   c ← b – a
FIN
```



1 min

solo

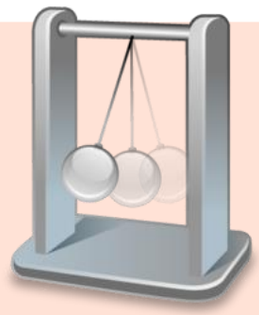
Live



Exercice

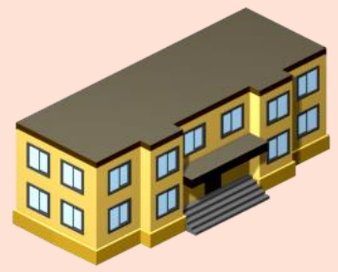
Quelles seront les valeurs des variables A et B après exécution des instructions suivantes ?

```
VAR: a, b: ENTIER
DEBUT
   a ← 5
   b ← a + 4
   a ← a + 1
   b ← a – 4
FIN
```

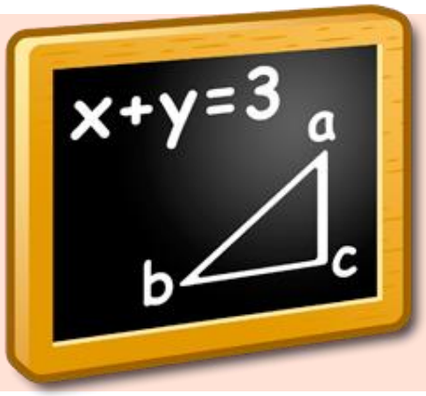


1 min

solo



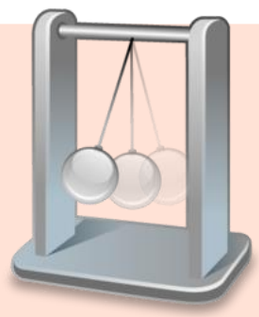
Live



Exercice

Quelles seront les valeurs des variables A, B et C après exécution des instructions suivantes ?

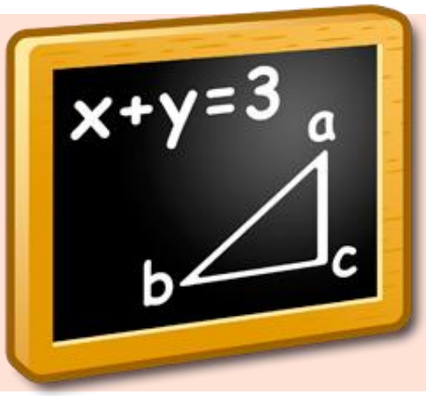
```
VAR: a, b, c: ENTIER
DEBUT
   a ← 3
   b ← 10
   c ← a + b
   b ← a + b
   a ← c
FIN
```



1 min

solo

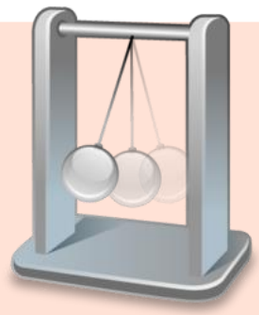
Live



Exercice

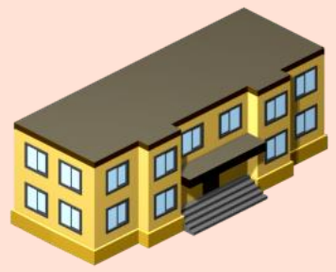
Quelles seront les valeurs des variables A, et B après exécution des instructions suivantes ?

```
VAR: a, b: ENTIER
DEBUT
   a ← 5
   b ← 2
   a ← b
   b ← a
FIN
```

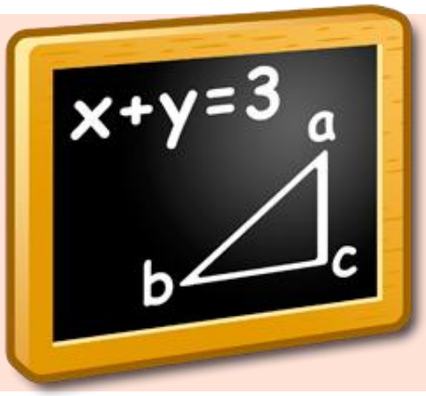


1 min

solo



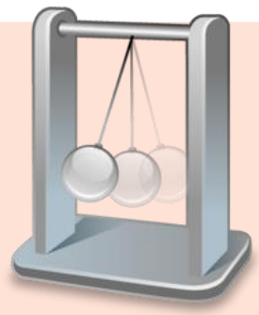
Live



Exercice

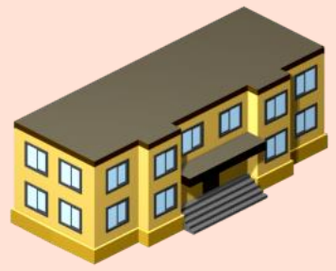
Quelles seront les valeurs des variables A, et B après exécution des instructions suivantes ?

```
VAR: a, b: ENTIER
DEBUT
   a ← 5
   b ← 2
   b ← a
   a ← b
FIN
```

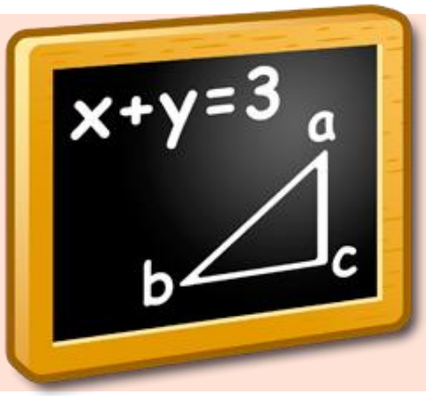


1 min

solo

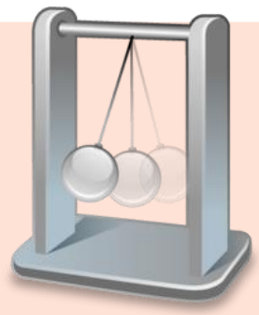


Live



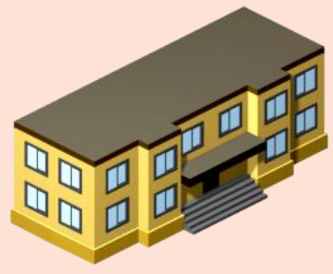
Exercice

Plus difficile, mais c'est un classique absolu, qu'il faut absolument maîtriser : écrire un algorithme permettant d'échanger les valeurs de deux variables a et b, et ce quel que soit leur contenu préalable.



3 min

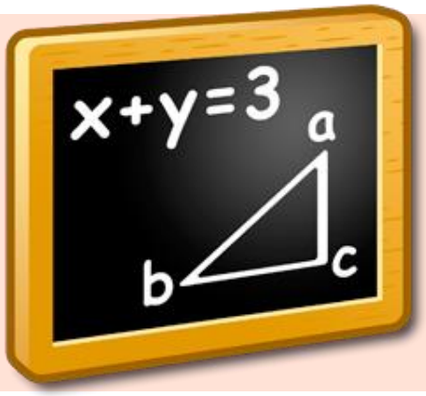
solo



Live

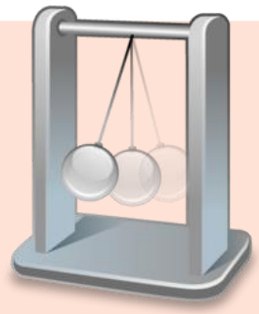
Solution

VAR: a, b, c: ENTIER
DEBUT
    a ← ...
    b ← ...
    c ← a
    a ← b
    b ← c
FIN



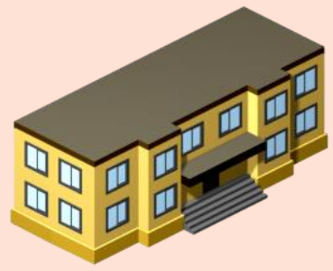
Exercice

Une variante du précédent : on dispose de trois variables a, b et c. Ecrivez un algorithme transférant à b la valeur de a, à c la valeur de b et à a la valeur de c (toujours quels que soient les contenus préalables de ces variables).



5 min

solo



Live

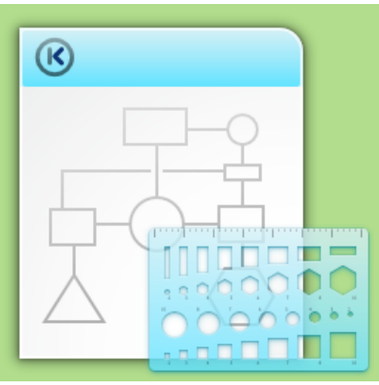
Solution

```
VAR: a, b, c, x: ENTIER
DEBUT
   a ← ...
   b ← ...
   c ← ...
   x ← c
   c ← b
   b ← a
   a ← x
FIN
```

Algorithmie> Présentation
Notion de constante

- Les Variables : VAR
  - Déclaration
  - Affectation
  - Utilisation
  - Ré-affectation

- Les Constantes : CONST
- Initialisation (avec déclaration)
- Utilisation



Déclaration

Nom de la constante déclarée

valeur définitive

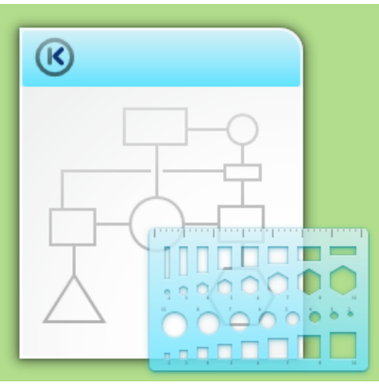
nom de la constante

valeur

nom_de_la_constante ← valeur : <Type>

affectation

Symbole d'affectation



Déclaration

Nom de la constante déclarée

valeur définitive

nom de la constante

valeur

Pi ← 3,14 : REEL

affectation

Symbole d'affectation

Algorithmie> Présentation
Notion de constante

Pourquoi utiliser une constante plutôt qu'une variable ?

- Optimisation
- Fiabilité
- Souplesse
- facilité de relecture

Algorithmie> Présentation
Notion de données

- Les données manipulées par les instructions sont :
- des variables proprement dites
- des constantes
- des valeurs littérales ("Zombie U", 45, VRAI)
- des expressions complexes (combinaisons de variables, constantes et valeurs littérales avec des opérateurs)

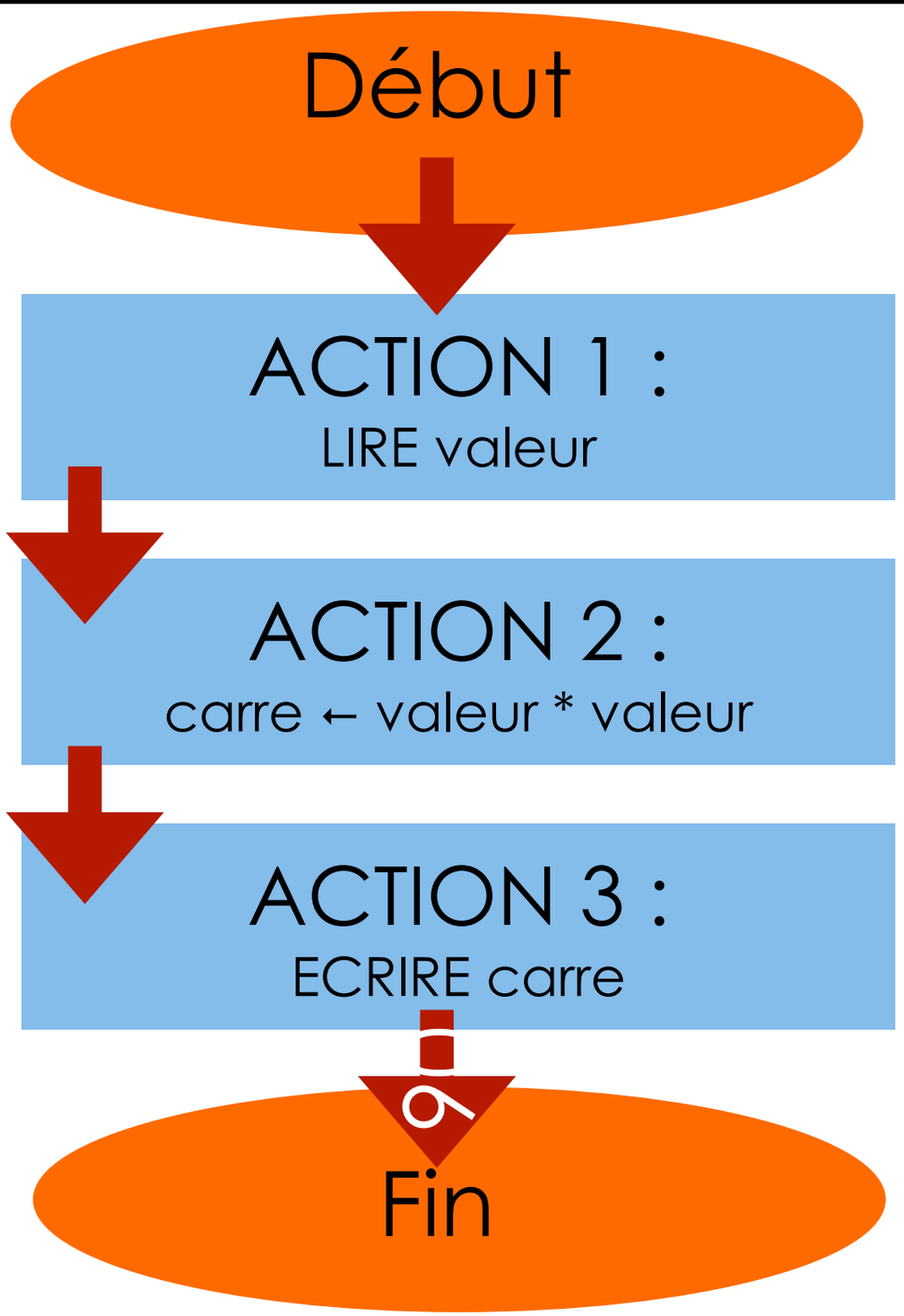
Algorithmie> Présentation
Primitives

- Une instruction est un ordre élémentaire - au sens algorithmique, que peut exécuter un programme.

Deux instructions d'entrée-sortie :

- LIRE : lecture de la frappe (au clavier)
- ECRIRE : affichage (à l'écran)

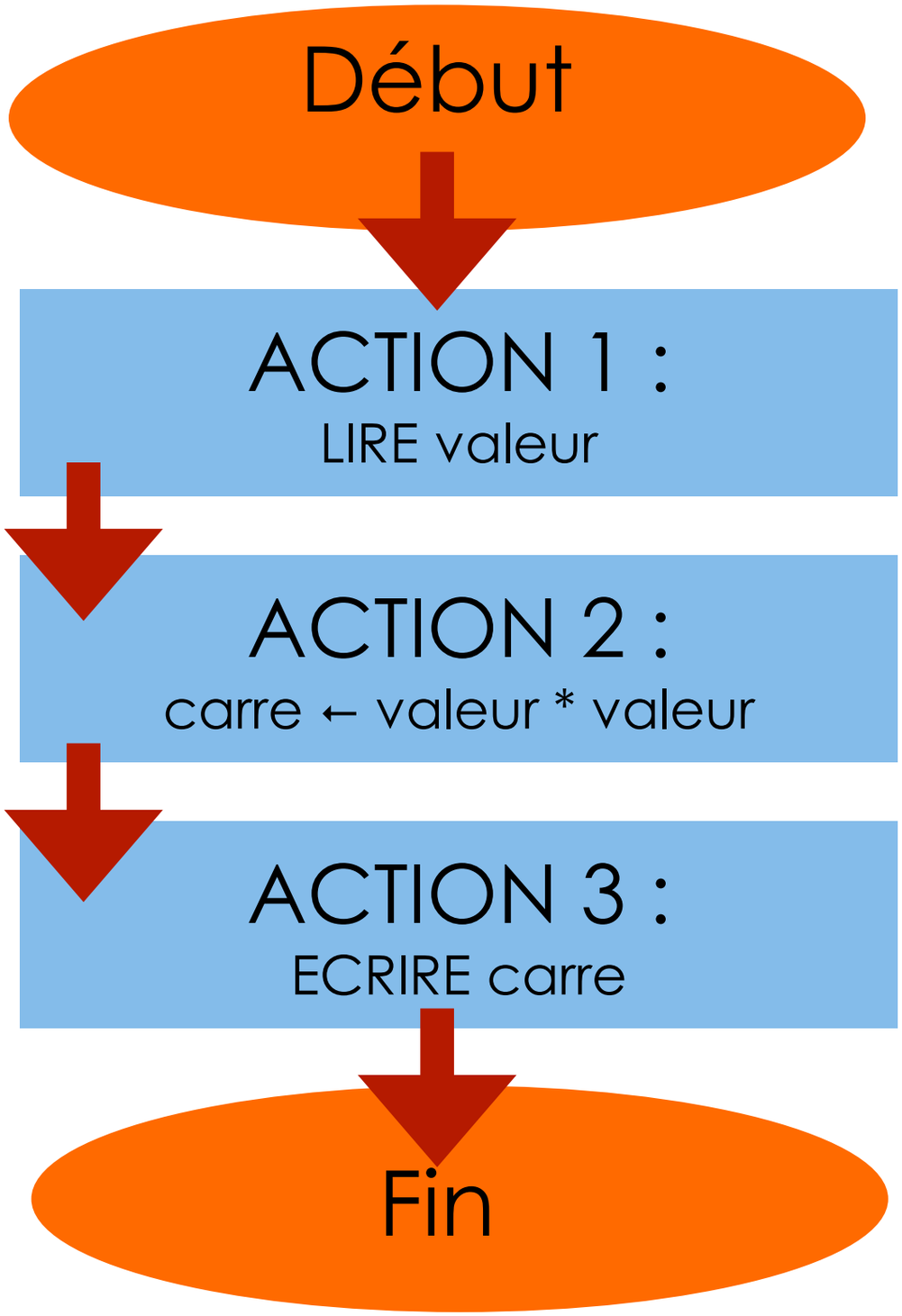
Algorithmie> Présentation
Notion de séquences



Suite d'opérations élémentaires (ou ACTIONS) dont l'exécution est séquentielle.

La lecture d'un algorithme s'effectue de haut en bas.

Algorithmie> Présentation
Notion de séquences




```
ALGORITHME: Algo_carre
//BUT :     calcul le carré d'un nombre
//ENTREE : entier saisi par l'utilisateur
//SORTIE : valeur de l'entier au carré

VAR :
          valeur, carre : ENTIER

DEBUT
   LIRE valeur                        // ACTION 1
   carre ← valeur * valeur            // ACTION 2
   ECRIRE  carre                      // ACTION 3
FIN
```

Algorithmie> Présentation
Structure d'un algorithme

```
ALGORITHME : nom_algorithme
```

Nom de l'algorithme

```
//BUT : cet algorithme effectue ...
//ENTREE : une liste de M noms
//SORTIE : une liste triée par ordre alphabétique
```

Spécifications de l'algorithme

```
// CONSTANTES : les constantes
// l'initialisation est obligatoire
// au moment de leur déclaration
CONST :  pi  ⟵ 3,1416      : REEL
         nom ⟵ "Bonjour"   : CHAINE
// VARIABLES : les variables (au sens strict)
VAR :    prix, quantité    : REEL
         prenom            : CHAINE
```

Déclarations

```
DEBUT
// instructions
FIN
```

Corps de l'algorithme

Algorithmie> Présentation
Conventions d'écriture

Les commentaires :

- sont précédés de : //

Les mots clés et Primitives :

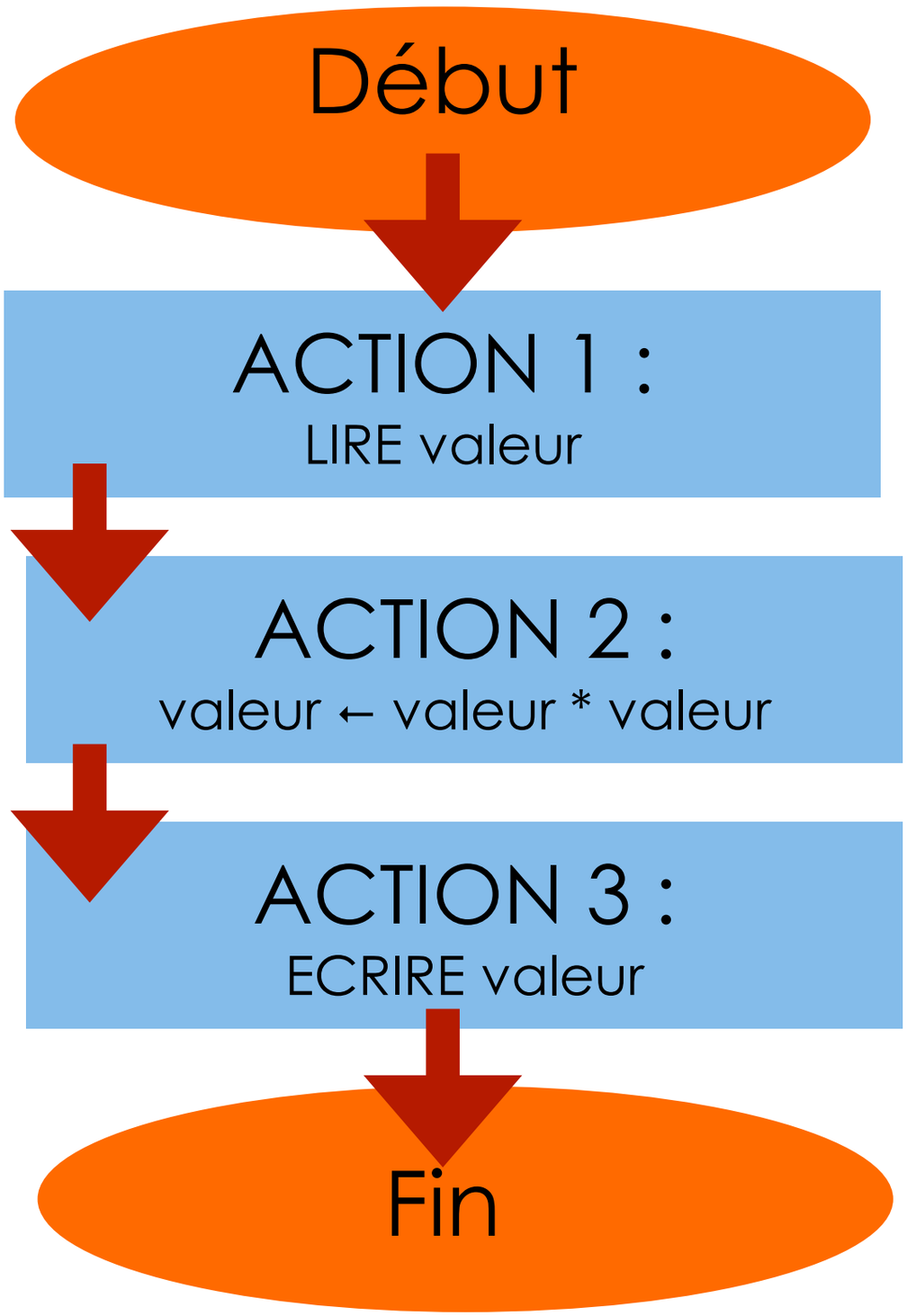
- sont écrits en majuscule : VAR:

Les chaînes de caractères sont encadrés par : "

Les caractères uniques sont encadrés par : '

Les accents sont seulement autorisés :

- dans les lignes de commentaire
- dans les chaînes de caractères : "élève"
- pour les caractères uniques : 'à '




```
ALGORITHME: carre
//BUT :     calcul le carré d'un nombre
//ENTREE : entier saisi par l'utilisateur
//SORTIE : valeur de l'entier au carré

VAR :
          valeur, produit : ENTIER

DEBUT
   LIRE valeur                          // ACTION 1
   valeur ← valeur * valeur             // ACTION 2
   ECRIRE valeur                        // ACTION 3
FIN
```

Algorithmie> Présentation Valeurs intermédiaires

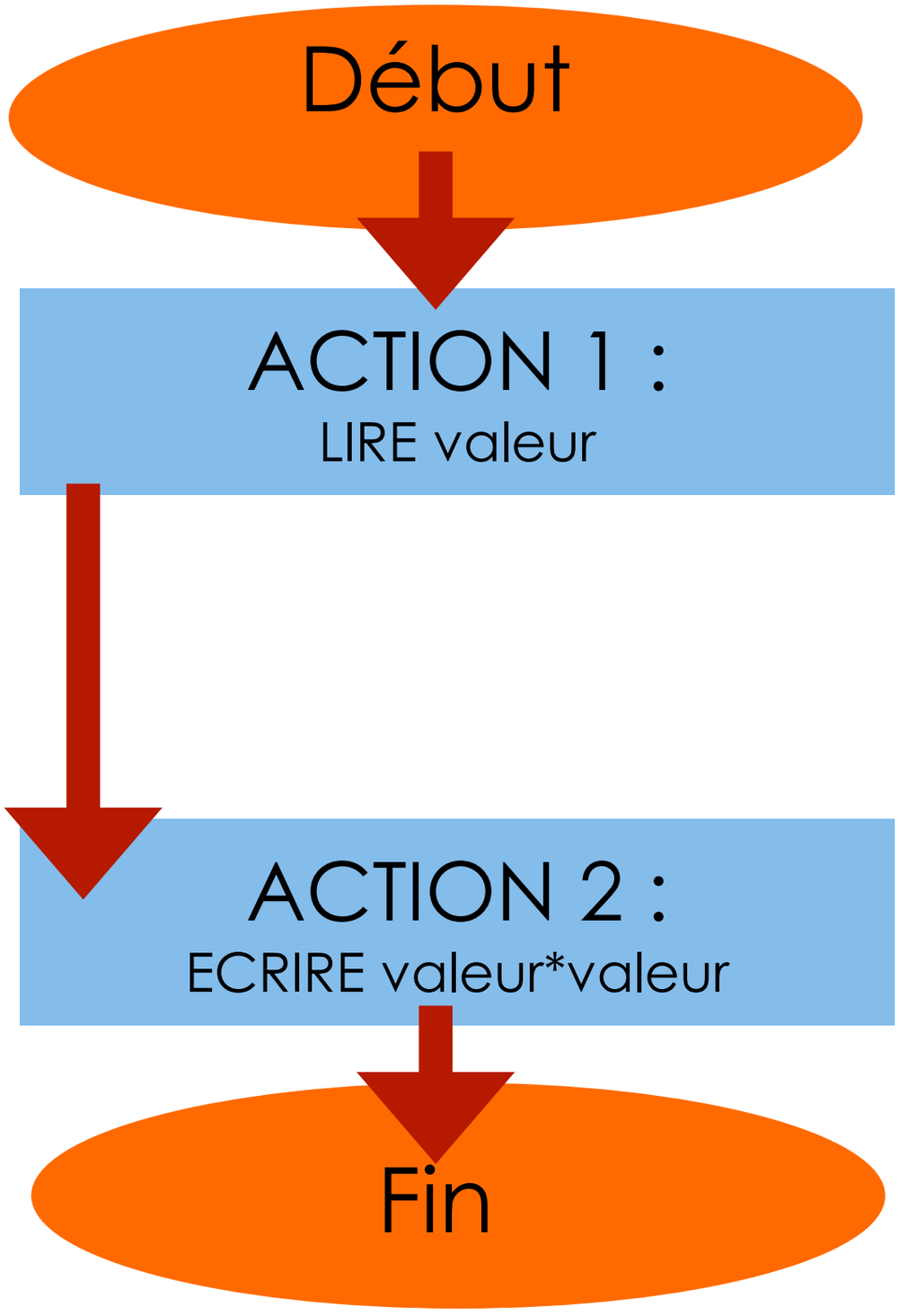
- On ne connait plus la valeur initiale entrée par l'utilisateur

- Il serait plus «user friendly» d'utiliser l'instruction

```
ECRIRE "le carré de"
ECRIRE valeur
ECRIRE "   est"
ECRIRE produit
```

- Est-on vraiment à un entier prêt ?

Algorithmie> Présentation
Notion de séquences



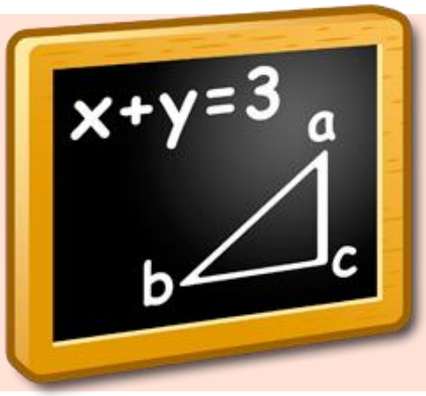

```
ALGORITHME: carre
//BUT :     calcul le carré d'un nombre
//ENTREE : entier saisi par l'utilisateur
//SORTIE : valeur de l'entier au carré

VAR :
          valeur: ENTIER

DEBUT
   LIRE valeur                      // ACTION 1
   ECRIRE valeur * valeur           // ACTION 2
FIN
```

Algorithmie> Présentation Valeurs intermédiaires

- On peut penser que la troisième solution est la plus optimisée mais l'ordinateur fera peut-être le calcul dans une variable avant de l'afficher.
- Plus important on perd en lisibilité en relisant le code
- Avec l'expérience, ne pas pouvoir «tracer» les valeurs intermédiaires lors du «deboggage» est une perte de temps.

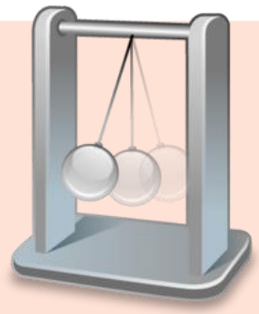
Algorithmie> Présentation
Notion de séquences

```
ALGORITHME: Algo_carre
//BUT :    calcul le carré d'un nombre
//ENTREE : entier saisi par l'utilisateur
//SORTIE : valeur de l'entier au carré

VAR :
         valeur, carre : ENTIER

DEBUT
   ECRIRE ''Donnez un nombre pour calculer son carré''
   LIRE valeur                    // ACTION 1
   carre ← valeur * valeur        // ACTION 2
   ECRIRE "Le carré de"
   ECRIRE  valeur
   ECRIRE ''est''
   ECRIRE  carre                  // ACTION 3
FIN
```



Exercice

Ecrire un algorithme complet qui demande trois nombres entiers à un utilisateur et qui affiche leur somme.



10 min

solo

Live

Solution

```
ALGORITHME : Calcul1
//BUT : cet algorithme effectue la somme de trois entiers
//ENTREE : Trois entiers saisis par l'utilisateur
//SORTIE : la somme
VAR :    nb1, nb2,nb3, somme          : ENTIER
```

DEBUT

ECRIRE " Veuillez entrer 3 entiers "

LIRE nb1

LIRE nb2

LIRE nb3

somme ← nb1 + nb2 + nb3

ECRIRE " La somme est: "

ECRIRE somme

FIN

Conseil

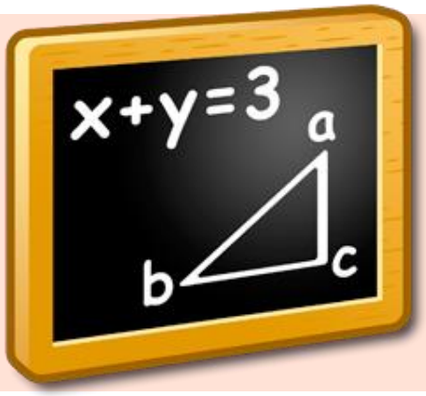
Les primitives LIRE et ECRIRE peuvent prendre plusieurs paramètres en entrée.

Solution

```
ALGORITHME : Calcul1
//BUT : cet algorithme effectue la somme de trois entiers
//ENTREE : Trois entiers saisis par l'utilisateur
//SORTIE : la somme
VAR :     nb1, nb2,nb3, somme          : ENTIER
```

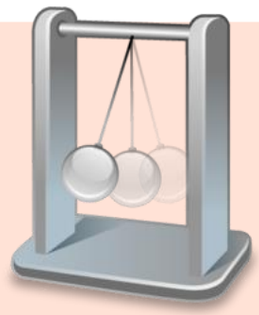
DEBUT
ECRIRE " Veuillez entrer 3 entiers "
LIRE nb1,nb2,nb3
somme ← nb1 + nb2 + nb3
ECRIRE " La somme est: " + somme
FIN

```
ALGORITHME : Calcul1
//BUT : cet algorithme effectue la somme de
trois entiers
//ENTREE : Trois entiers saisis par
l'utilisateur
//SORTIE : la somme
VAR :     nb1, nb2,nb3, somme : ENTIER
DEBUT
   ECRIRE " Veuillez entrer 1 entier "
   LIRE nb1
   ECRIRE " Veuillez entrer un deuxième entier "
   LIRE nb2
   ECRIRE " Veuillez entrer un dernier entier "
   LIRE nb3
   somme ← nb1 + nb2 + nb3
   ECRIRE " La somme est : " + somme
FIN
```



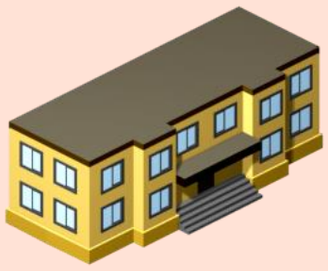
Exercice

Ecrire un algorithme complet qui demande deux nombres entiers à un utilisateur et qui affiche la somme, le produit et le quotient de ces valeurs.



10 min

solo



Live

Solution

```
ALGORITHME : Calcul2
//BUT : cet algorithme effectue divers calcul
//ENTREE : Deux entier saisis par l'utilisateur
//SORTIE : la somme, le produit et le quotient
VAR :     nb1, nb2, somme, produit     : ENTIER
          quotient                     : REEL
```

DEBUT

ECRIRE ''Veuillez entrer deux nombres''

LIRE nb1

LIRE nb2

somme ← nb1 + nb2

produit ← nb1 * nb2

quotient← nb1 / nb2

ECRIRE somme

ECRIRE produit

ECRIRE quotient

FIN

Solution

```
ALGORITHME : Calcul2
//BUT : cet algorithme effectue divers calcul
//ENTREE : Deux entier saisis par l'utilisateur
//SORTIE : la somme, le produit et le quotient
VAR :     nb1, nb2, somme, produit     : ENTIER
          quotient                     : REEL
```

DEBUT

LIRE nb1, nb2

somme ← nb1 + nb2

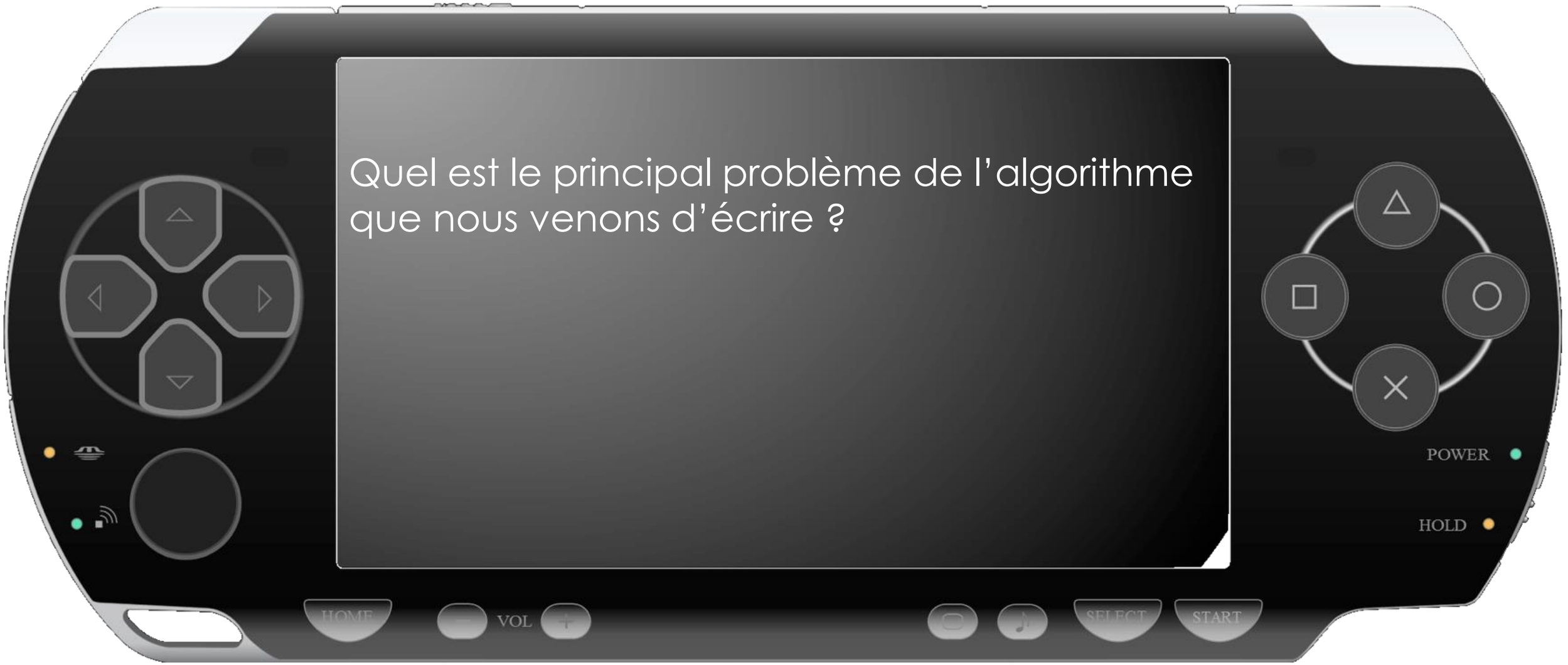
produit ← nb1 * nb2

quotient← nb1 / nb2

ECRIRE somme + produit + quotient

FIN

Question


Quel est le principal problème de l'algorithme que nous venons d'écrire ?
POWER
HOLD
HOME
VOL
SELECT
START

Solution

```
ALGORITHME : Calcul2
//BUT : cet algorithme effectue divers calcul
//ENTREE : Deux entier saisis par l'utilisateur
//SORTIE : la somme, le produit et le quotient
VAR :     nb1, nb2, somme, produit     : ENTIER
          quotient                     : REEL
```

DEBUT

LIRE nb1, nb2

somme ← nb1 + nb2

produit ← nb1 * nb2

quotient← nb1 / nb2 ☚ et si nb2 vaut zéro ?

ECRIRE somme + produit + quotient

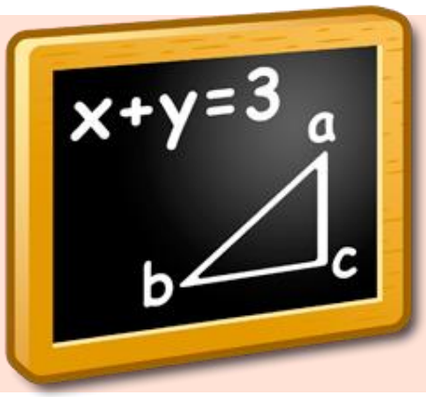
FIN

Interdiction

Le développeur doit s'assurer qu'aucune «aberration» ne peut se produire

Conseil

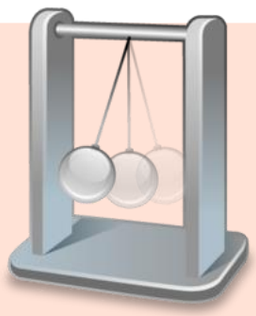
En algorithmie on considère que la lecture ou l'écriture «réussissent». Dans le cas d'un programme, c'est le rôle du développeur de s'assurer que l'utilisateur ne rentre pas «Mario» dans un réel, 3.14 dans un entier, etc...
Cette vérification constante est fastidieuse mais nécessaire.
Pensez aussi lors d'un site web qu'un utilisateur peut saisir des balises dans un champ de formulaire.



Exercice

Indiquer ce qu'affiche l'algorithme suivant:

```
ALGORITHME : exo1
VAR :    x,y        : ENTIER
DEBUT
   LIRE x,y
   x ← x+2-x
   ECRIRE x + 'x' + "x" + 2*x
Fin
```



5 min

solo

Live

Solution

2xx4

Algorithmie> Présentation Structures

Pour pouvoir répondre à toutes les éventualités nous devons structurer nos algorithmes (donc nos programmes et donc nos pensées)

- Structure algorithmique :
  - Séquences
  - Ruptures de séquences
- Structure de données :
  - Structures élémentaires
  - Structures avancées
- Structure de programmes

FIN

de la Partie 1