

# Logiciel EXECALGO

## 1/ Introduction

Le logiciel Execalgo (Version 2) a pour objectif de faciliter l'enseignement de la notion d'algorithme pour les élèves du second cycle (sans qu'il soit exclu de l'utiliser avec des collégiens).

Un algorithme mis en œuvre avec ce logiciel se présente sous la forme d'une suite d'instructions dont les mots clés sont en français, formant un texte lisible, ce qui facilite son analyse.

Le résultat de l'exécution est aussi sous forme de texte.

Ces textes peuvent être copiés, collés, enregistrés, imprimés, transmis par réseau ou Internet comme n'importe quel autre texte.

Le texte du programme et le texte résultant de son exécution sont présentés côte à côte par le logiciel.

L'exécution peut être demandée pas à pas ; dans ce cas l'effet de chaque instruction exécutée est explicité dans le texte résultat (voir exemple plus loin).

L'enseignant peut choisir d'utiliser ce logiciel de façon collective ou individuelle, selon la complexité des algorithmes à traiter et les capacités mobilisables par les élèves (et selon les opportunités offertes par l'équipement de l'établissement).

Le nombre d'instructions et le nombre d'opérations élémentaires exécutées est ajouté à la fin du texte résultat, ce qui fournit un instrument de mesure de l'efficacité de l'algorithme.

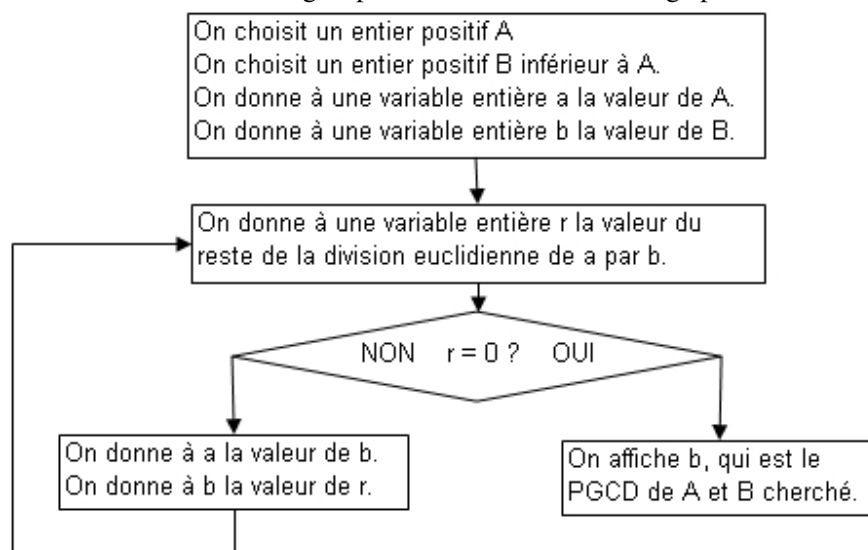
Les compétences recherchées sont :

- Comprendre ce qu'est une variable et le type d'une variable.
- Comprendre comment les affectations ou demandes permettent d'agir sur les variables.
- Comprendre ce que sont les affichages, ou sorties.
- Comprendre la notion de déroulement séquentiel d'un programme.
- Comprendre les exécutions conditionnelles et les boucles.

## 2/ Exemple : l'algorithme d'Euclide avec division et reste

**Étape 1** : On décrit l'algorithme par un schéma ou un texte en français.

L'algorithme d'Euclide peut être décrit par l'organigramme ci-dessous, pour lequel les codages standards ont été utilisés : des rectangles pour les actions et un losange pour le test.



Ce schéma est déjà un premier pas vers la mise en application de l'algorithme : des variables sont nommées, les actions à effectuer, avant le test et selon son résultat, sont décrites avec précision.

**Étape 2** : Le schéma est traduit par une suite d'instructions.

Cette étape oblige à se passer des avantages du schéma : des flèches qui peuvent partir dans toutes les directions pour indiquer quelle sera l'instruction à exécuter à la suite d'une instruction donnée au cours du déroulement de l'algorithme. En effet, par défaut, les instructions sont exécutées dans l'ordre où elles ont été écrites.

Lorsque l'ordre d'écriture devra ne pas être respecté, ces flèches seront remplacées par des instructions appelées ici « branchement conditionnel » lorsqu'elles dépendent du résultat d'un test et « branchement simple » dans le cas contraire.

Il faut d'autre part définir la partie du programme vers laquelle le branchement doit s'effectuer, c'est le rôle d'un « point de branchement ».

- Implémentation possible avec le logiciel Execalgo :

<b>Lignes de programme</b>	<b>Commentaires</b>
# Déclaration des variables	Une ligne commençant par # est une ligne de commentaire non analysée, n'entrant pas dans le décompte des instructions exécutées.
Variable(entier ; A ; B ; a ; b ; r )	Types de variables possibles : entier, réel, complexe, booléen, texte On peut déclarer une liste, par exemple L(1..1000) On peut déclarer un tableau : T(1..10,1..100)
# Entrées	Commentaire
Demander( A )	Affectation choisie par l'utilisateur. Syntaxe : Demander ( variable )
Demander( B )	Affectation choisie par l'utilisateur
# Initialisations	Commentaire
Affecter(a ; A)	Affectation définie par le programmeur Celle-ci a pour objectif de conserver la valeur de A pour l'affichage final. Syntaxe : Affecter ( variable ; valeur ) Voir la syntaxe plus détaillée en fin de document.
Affecter( b ; B)	Affectation définie par le programmeur
FaireTantQue ( r>0 )	Boucle du type « Tant que » Syntaxe : FaireTantQue(condition) Instructions FinFaire
Affecter ( r ; reste(a,b) )	La fonction reste est prédéfinie, on peut ne pas l'utiliser et détailler le calcul en passant par le quotient (voir la syntaxe en fin de document).
Afficher (" a =" ; a ; "        b =" ; b ; "        r =" ; r )	Instruction d'affichage. Voir la syntaxe en fin de document.
Affecter( a ; b)	Affectation
Affecter ( b ; r)	Affectation
FinFaire	Fait revenir à l'instruction FaireTantQue associée
Afficher ("Le PGCD de" ; A ; " et" ; B ; " est " ; a)	Affichage de sortie

### 3/ Exemple d'affichage produit par le logiciel

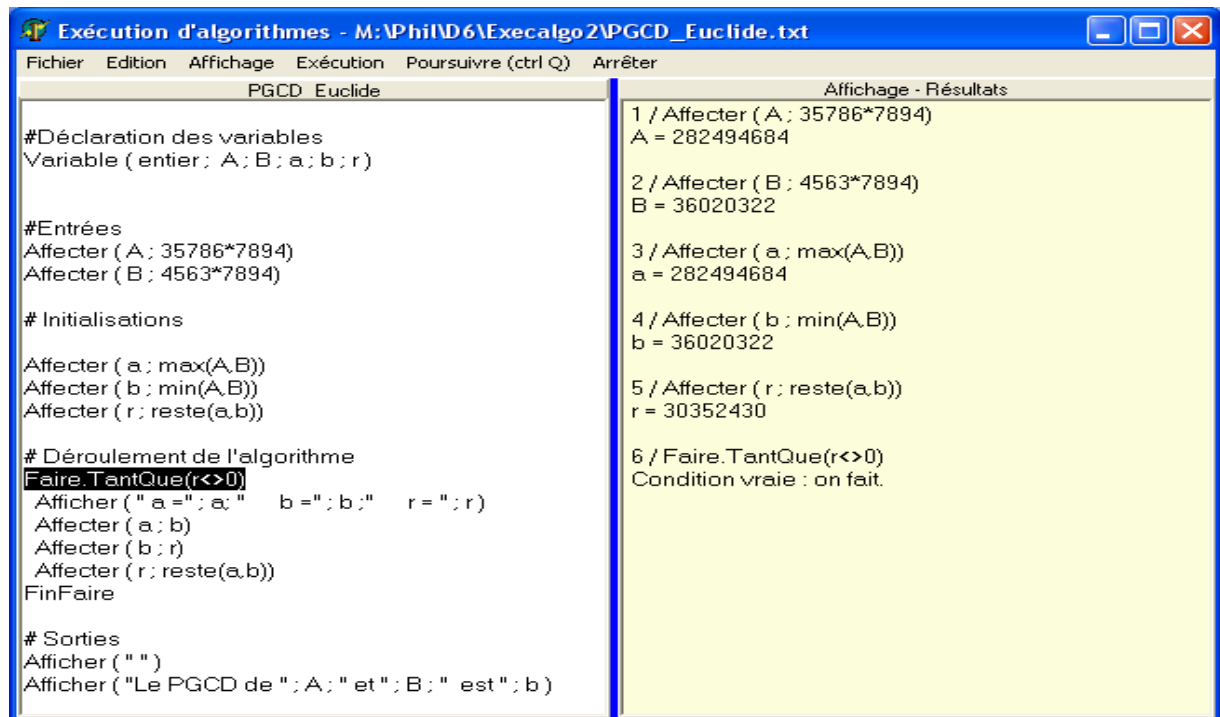
N.B. Tant le programme que son résultat sont présentés sous forme de texte simple et peuvent être enregistrés ou copiés et collés dans tout document.

Programme	Résultat de l'exécution
<pre>#Déclaration des variables Variable ( entier ; A ; B ; a ; b ; r ) #Entrées Affecter ( A ; 35786*7894) Affecter ( B ; 4563*7894)  # Initialisations  Affecter ( a ; max(A,B)) Affecter ( b ; min(A,B)) Affecter ( r ; reste(a,b))  # Déroulement de l'algorithme FaireTantQue(r&lt;&gt;0)   Afficher ( " a =" ; a ; "   b =" ; b ; "   r =" ; r )   Affecter ( a ; b)   Affecter ( b ; r)   Affecter ( r ; reste(a,b)) FinFaire  # Sorties Afficher ( " " ) Afficher ( "Le PGCD de " ; A ; " et " ; B ; " est " ; b )</pre>	<pre>a =282494684   b =36020322   r = 30352430 a =36020322   b =30352430   r = 5667892 a =30352430   b =5667892   r = 2012970 a =5667892    b =2012970   r = 1641952 a =2012970    b =1641952   r = 371018 a =1641952    b =371018   r = 157880 a =371018     b =157880   r = 55258 a =157880     b =55258    r = 47364 a =55258      b =47364    r = 7894  Le PGCD de 282494684 et 36020322 est 7894 Exécution terminée. Instructions exécutées : 62</pre>

#### 4/ Le mode exécution pas à pas

Il permet de suivre le déroulement du programme en mettant en correspondance la ligne de programme en cours d'exécution et le résultat de cette exécution.

Exemple :



#### 5/ Correspondance entre le logiciel et des calculatrices programmables

Correspondance entre des rédactions possibles de cet algorithme avec le logiciel d'une part et les marques les plus courantes de calculatrice utilisées en lycée d'autre part. Cet exemple utilise volontairement un éventail minimum d'instructions pour être compatible avec un maximum de modèles.

Type d'instruction	Avec Execalgo (version 2)	Sur Casio	Sur TI
Affectation (par l'utilisateur)	Demander( A )	Input A	
Affectation (par le programmeur)	Affecter ( B ; 56745 )	56745 → B	
Affectation	Affecter( R ; 0 )	0 → R	
Boucle de type « Tant que »	FaireTantQue(R>0)	While R>0	
Affectation	Affecter ( R ; reste(A,B) )	A - B*Int(A/B) →R	
Affectation	Affecter ( A ; B )	B → A	
Affectation	Affecter ( B ; R )	R → B	
Fin de boucle	FinFaire	WhileEnd	End
Affichage	Afficher B	B ↵	Disp B

## 6/ Interface du logiciel

### 6. 1. Le menu Fichier et Le menu Édition

Les items de ces menus ont le même usage que dans tout traitement de texte.

Les copies après sélection de texte peuvent concerner le texte du programme ou le texte produit par l'exécution de ce programme

Le format d'enregistrement est le format texte élémentaire (suffixe .txt).

### 6. 2. Le menu Affichage

#### 6. 2. 1. L'item Informations et l'item Résultats

L'item Informations permet d'afficher dans la partie droite, à la place des résultats, un texte qui rappelle un certain nombre d'informations sur le logiciel, notamment un résumé de la syntaxe. L'item Résultats permet de revenir, dans la partie droite, à l'affichage des résultats de l'exécution du programme.

#### 6. 2. 2. L'item Taille Affichages

Il permet d'augmenter la taille des affichages, notamment dans le cas d'une présentation collective à l'aide d'un vidéo-projecteur.

#### 6. 2. 3. L'item A propos

Il indique notamment que le logiciel est libre de tous droits.

Les défauts repérés ou les demandes d'améliorations peuvent être envoyées à :

[philippe.seres@ac-paris.fr](mailto:philippe.seres@ac-paris.fr)

### 6. 3. Le menu Exécution

#### 6. 3. 1. L'item Normale (Ctrl-E)

Cet item lance l'analyse du programme, puis, si aucune erreur n'a été décelée, il l'exécute.

Pour éviter un plantage dû à un texte de programme qui tourne en boucle sans se terminer, l'exécution est automatiquement suspendue après 10 millions d'instructions exécutées. L'utilisateur peut alors arrêter l'exécution ou la poursuivre pour encore au maximum une nouvelle tranche de 10 millions d'instructions

#### 6. 3. 2. L'item Pas à pas

Cet item lance l'analyse du programme, puis, si aucune erreur n'a été décelée, il exécute la première instruction.

Pour continuer, utiliser l'item Poursuivre (Ctrl-Q), qui permet d'exécuter l'instruction suivante.

Les items **Poursuivre (Ctrl-Q)** et **Arrêter** sont relatifs à l'exécution. Ils ont été placés au premier niveau pour faciliter leur accès.

## 7/ Syntaxe des instructions comprises par le logiciel

Lorsqu'une instruction reçoit plusieurs paramètres, le séparateur utilisé est le point-virgule.

### 7. 1. Les commentaires

Tout texte écrit sur une ligne après un # est considéré comme un commentaire libre et n'est pas analysé.

Exemples :

# Déclaration des variables :

Variable( Entier ; p) # p représente le nombre des diviseurs de n

## 7. 2. Les variables

Les variables, réelles ou entières, peuvent être des variables simples, des listes (tableaux à une dimension) ou des tableaux (à deux dimensions).

Les caractères autorisés sont les lettres (non accentuées) et les chiffres. Le premier caractère doit être une lettre.

Exemples :

Variable (entier ; j ; k ; L (1..100))

Variable (réel ; x1 ; x2 ; y ; A (1..4 , 1..4))

Les mots clés pour les types de variables reconnus sont : entier, réel, condition (mot plus familier en seconde que booléen), texte, complexe.

## 7. 3. Les affectations

Les valeurs doivent être compatibles avec le type des variables : par exemple une variable entière ne peut recevoir un réel. Voir plus loin la syntaxe des expressions.

Les affectations peuvent être faites par le programmeur (Affecter) ou par l'utilisateur (Demander)

Exemples :

**Affecter** ( n ; n+1 )

**Affecter** ( A ( 1 , 3 ) ; (2\*x + 1)\*(3\*x - 2) )

**Demander** ( A )

La valeur fournie par l'utilisateur doit être une constante et non une expression.

Les demandes sont réservées aux variables de type réel, entier ou texte.

## 7. 4. Les affichages

On peut demander l'affichage d'une suite d'éléments tels que :

du texte ; des expressions ; des plages de variables liste ou tableau.

Pour les réels, le nombre de décimales affichées est limité à 8. La fonction arrondi permet de le réduire : arrondi(pi,4) est égal à 3.1416

Exemples :

**Afficher** ( "k = " ; k ; "x = " ; x ; L (k) )

**Afficher** ( "Liste des diviseurs :" )

**Afficher** ( L (1..n) )

**Afficher** ( A (1,1..4) )

Une liste peut être affichée « à l'envers ». Exemple : Afficher L(100..1)

Si une expression a pour résultat un booléen, l'affichage produit la lettre V ou la lettre F.

Exemple : **Afficher** (x\*y < 2 )

## 7. 5. Les exécutions conditionnelles

**FaireSi**(condition)

instructions exécutées si la condition est vraie

**Sinon** (optionnel)

instructions exécutées si la condition est fausse (optionnel)

**FinFaire**

**FaireTantQue**(condition)

instructions exécutées en boucle tant que la condition est vraie

**FinFaire**

**FaireFois**(nb)

instructions exécutées nb fois en boucle

**FinFaire**

## 7. 6. Les modules et fonctions (sous-programmes)

La déclaration d'une fonction est délimitée par un couple Fonction / Retourne.

Pour un module : Module / FinModule

Entre les deux, il peut y avoir des déclarations de variables locales et d'autres instructions.

Syntaxe :

**Fonction**(nom de la fonction ; type du résultat ; type d'un paramètre ; nom d'un paramètre ; etc..)

**Module**(nom du module ; type d'un paramètre ; nom d'un paramètre ; etc..)

Le nombre de paramètres est limité à 10.

Exemples :

**Fonction** ( f ; réel; réel : x)

**Retourne** (  $x^2 - 10$ )

**Fonction** ( u ; réel ; entier ; n)

**Variable** ( réel ; x)

**Affecter** ( x ; - 3 )

**FaireFois**(n)

**Affecter** ( x ; f(x) )

**FinFaire**

**Retourne** ( x )

Les paramètres peuvent être des variables simples, listes ou tableaux.

Par défaut, une fonction ne modifie pas les paramètres qu'elle reçoit.

Elle peut cependant le faire pour tout paramètre précédé du mot clé : modif

Dans ce cas, à l'appel de la fonction, le paramètre correspondant doit être une variable du même type et de la même taille.

Exemple :

**Module**( prodmat ; entier ; a(1..2,1..2) ; b(1..2,1..2) ; modif p(1..2,1..2))

**Variable**(entier; m ; j ; k)

**Affecter**(m;0)

**FaireFois**(2)

**Affecter**(m ; m+1)

**Affecter**(j;0)

**FaireFois**(2)

**Affecter**(j ; j+1)

**Affecter**(p(m , j) ; 0)

**Affecter**(k ; 0)

**FaireFois**(2)

**Affecter**(k ; k+1)

**Affecter**(p(m , j) ; p(m , j) + a(m , k) \* b(k , j))

**FinFaire**

**FinFaire**

**FinModule**

L'exécution d'un module se fait à l'aide de l'instruction FaireModule :

Exemple : FaireModule ( prodmat ( A ; B ; P ) )

Les modules et fonctions ont un triple objectif :

a/ Permettre de matérialiser la décomposition des tâches à accomplir en tâches plus simples.

b/ Permettre de n'écrire qu'une fois une liste d'instructions exécutées ensuite à des moments différents.

c/ Permettre d'évacuer une partie des instructions du mode d'exécution Pas à Pas :

Si une partie bien maîtrisée de l'algorithme ne ferait qu'alourdir inutilement un déroulement pas à pas, le fait de la placer dans un module aura pour effet de la remplacer dans la partie principale du programme (celle qui est développée pas à pas) par une seule instruction, celle d'appel du module)

## 7.7 L'instruction Pause

Elle permet d'interrompre ou non le programme à un endroit particulier, permettant à l'affichage de s'actualiser. Elle est utilisée pour « déboguer » le programme, c'est à dire pour analyser son mauvais fonctionnement et y remédier.

## 8. Syntaxe des expressions

La virgule étant réservée pour séparer les arguments des fonctions à plusieurs variables, le point décimal doit être utilisé pour les constantes décimales.

Pour certaines opérations ou fonctions, le type du résultat dépend du type des arguments. Par exemple la somme d'un entier et d'un réel sera logiquement un réel.

Lorsqu'un réel est attendu comme argument, on peut fournir un entier, et pour un complexe, un entier ou un réel.

Deux constantes sont prédéfinies : le réel pi ( $\pi$ ) et le nombre complexe  $i$ . Cependant si le programmeur nomme une variable  $i$ , elle pourra être reconnue (ne pas le faire si on utilise les complexes).

### 8.1. Les opérations

Nom	Types des arguments	Type du résultat
+ (somme)	entier, réel, complexe, texte	entier, réel, complexe, texte
- (opposé, soustraction)	entier, réel, complexe	entier, réel, complexe
* (multiplication)	entier, réel, complexe	entier, réel, complexe
/ (division)	réel, complexe	réel, complexe
! (factorielle)	entier	entier
= (égale)	entier, réel, complexe, texte, condition	condition
<> (différent)	entier, réel, complexe, texte, condition	condition
< (inférieur)	entier, réel, texte	condition
<= (inférieur ou égal)	entier, réel, texte	condition
> (supérieur)	entier, réel, texte	condition
>= (supérieur ou égal)	entier, réel, texte	condition
^ (puissance)	entier, réel, complexe	entier, réel, complexe

*Remarques :*

- Exemple pour la somme de deux textes : "bord" + "eaux" donne "bordeaux"
- La division donne un réel même pour deux entiers multiples l'un de l'autre. Pour obtenir un résultat de type entier, utiliser la fonction à deux variables « quot » (voir plus loin) qui donne le quotient euclidien.
- Le signe multiplier ne peut pas être sous-entendu : écrire  $2 + 3 * i$  et non  $2 + 3 i$ .
- pour la puissance (^), si y est un réel,  $x^y$  ne sera défini que pour  $x > 0$ .
- Pour les variables de type texte, les comparaisons utilisent l'ordre alphabétique.
- Deux conditions sont égales si elles sont toutes les deux vraies ou toutes les deux fausses.

## 8.2 Fonctions numériques à une variable numérique

Nom	Type de l'argument	Type du résultat
ent, int, E (partie entière)	réel	entier
rac (racine carrée)	réel	réel
cos (cosinus)	réel, complexe	réel, complexe
sin (sinus)	réel, complexe	réel, complexe
tan (tangente)	réel, complexe	réel, complexe
arccos (arc cosinus)	réel	réel
arcsin (arc sinus)	réel	réel
arctan (arc tangente)	réel	réel
abs (valeur absolue)	entier, réel	entier, réel

**Après la seconde :**

exp (exponentielle)	réel, complexe	réel, complexe
ln (logarithme népérien)	réel	réel
cosh (cosinus hyperbolique)	réel, complexe	réel, complexe
sinh (sinus hyperbolique)	réel, complexe	réel, complexe
tanh (tangente hyperbolique)	réel, complexe	réel, complexe
arccosh	réel	réel
arcsinh	réel	réel
arctanh	réel	réel
conj (conjugué)	complexe	complexe
arg (argument)	complexe	réel
module (module)	complexe	réel
re (partie réelle)	complexe	réel
im (partie imaginaire)	complexe	réel

### 8.3 Autres fonctions

- **quot**(a,b) et **reste**(a,b), avec a et b entiers, donnent respectivement le quotient entier et le reste entier de la division euclidienne de a par b.

- **Ehasard**(n1,n2), avec n1 et n2 entiers, donne un entier au hasard dans l'intervalle [n1 ; n2].

- **Rhasard**(a,b), avec a et b réels, donne un réel au hasard de l'intervalle [a,b[.

- **min**(a,b), **max**(a,b) avec a et b entiers ou réels.

- **arrondi**(x,n) donne l'arrondi du réel x avec n chiffres après la virgule.

- **μ**(condition) donne l'entier 1 si la condition est vraie et l'entier 0 si elle est fausse.

Exemple :  $\mu(2 < 5)$  donne 1

- **divise**(a, b) est vrai si l'entier a est un diviseur de l'entier b.

- **et** (c1 , c2) est vrai si les conditions c1 et c2 sont vraies

- **ou** (c1 , c2) est vrai si l'une au moins des conditions c1 ou c2 est vraie

- **premier** (n) est vrai si l'entier n est un nombre premier.

- **si** (condition, valeur si vrai, valeur si faux)

Le type du résultat dépend de celui des valeurs.

- **ecr** : renvoie le texte de l'écriture décimale d'un nombre.

Exemple :  $\text{ecr}(3/4)$  donne "0.75".

Fonctions pour les textes identiques aux fonctions tableur du même nom :

- **CAR** : Caractère d'après son code ASCII. Exemple :  $\text{CAR}(65) = "A"$

- **CODE** : fonction inverse de CAR :  $\text{CODE}("A") = 65$

- **NBCAR** : longueur du texte :  $\text{NBCAR}("trois") = 5$

- **CHERCHE** : cherche la position d'un sous-texte dans un texte

Exemple:  $\text{CHERCHE}("G", "ALGORITHME") = 3$

La fonction renvoie 0 si le sous-texte ne figure pas dans le texte.

- **STXT** : sous-texte extrait d'un texte avec une position et le nombre de caractères à extraire.

Exemple :  $\text{STXT}("ALGORITHME", 3, 5) = "GORIT"$

- **GAUCHE** : idem en partant de la gauche :  $\text{GAUCHE}("ALGORITHME", 5) = "ALGOR"$

- **DROITE** : idem en partant de la droite :  $\text{DROITE}("ALGORITHME", 4) = "THME"$