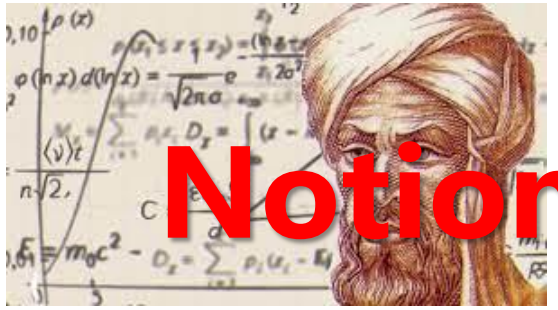
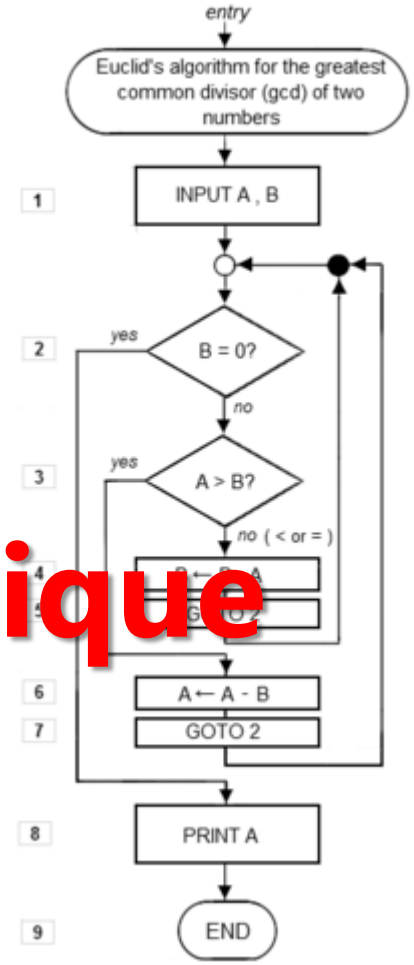
ALGOBOX



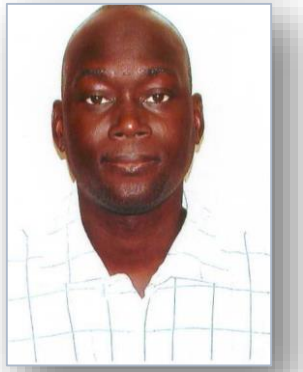
Algorithmique 1

Notions de base en Algorithmique



Prof. Ousmane SALL
Université de THIES - UFR Sciences et Technologies
Département Informatique
osall@univ-thies.sn

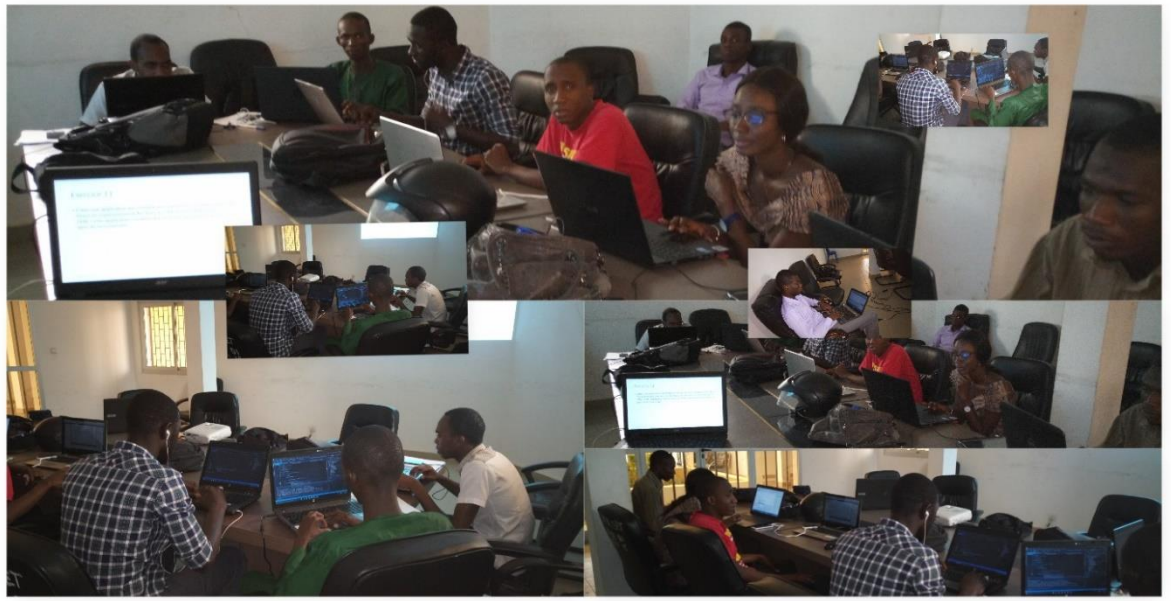
A propos de moi



- Enseignant-Chercheur à l'UFR SET- Université de THIES https://sites.google.com/a/univ-thies.sn/osall751/
- Enseignements:
  - Algorithmique et Programmation(C, Java, PHP)
  - Programmation WEB dynamique(HTML 5 CSS, PHP, MySQL, CMS,...)
  - Programmation Java, Dart, TypeScript
  - Programmation Java, Jakarta EE, JSF, Spring, SpringBoot, Angular
  - Technologies Mobiles Android, Xamarin, Ionic, Flutter
  - Programmation .Net, C#
  - Gestion de Projet Informatique
  - Génie Logiciel, Qualité et Métrique du Logiciel
- Contact:
  - osall@univ-thies.sn
  - UFR SET, Université de THIES -Dpt Informatique, BP 967 THIES.

Une sagesse chinoise…

« J'écoute et j'oublie; je lis et je comprends; je fais et j'apprends »
[Proverbe chinois]



Programme d'Algorithmique 1

- Introduction Générale
- **Notions de base en Algorithmique**
- Saisie et Affichage en algorithmique
- Les Structures de Contrôle
- Les Structures Itératives
- Les tableaux
- Sous-algorithmes/Fonctions

Objectifs/Compétences visé(e)s

- Ce chapitre présente les notions d'algorithmique et de programmation
- Il présente les types de base que l'on peut utiliser en informatique, les notions de variables et de constantes et les actions simples telles que l'affection de valeurs (à des variables) et les instructions d'entrées/sorties.

Qu’est-ce qu’un programme d’ordinateur?

- Allumez un ordinateur, vous n’en tirerez rien!!!
- Pour le faire marcher il faut lui fournir un programme
- Ordinateur = matériel + programme(s)
- Un programme est une suite d’instructions d’ordinateur
- Une instruction est un **ordre** compris par l’ordinateur et qui lui fait exécuter une action, c.-à-d. une modification de son environnement

Les catégories d’ordres

- les ordinateurs, quels qu’ils soient, ne sont fondamentalement capables de comprendre que quatre catégories d'**ordres** (en programmation, on n'emploiera pas le terme d'ordre, mais plutôt celui d'**instructions**). Ces quatre familles d'instructions sont :
  - l’**affectation** de variables
  - la **saisie** / **affichage**
  - les **tests**
  - les **boucles**

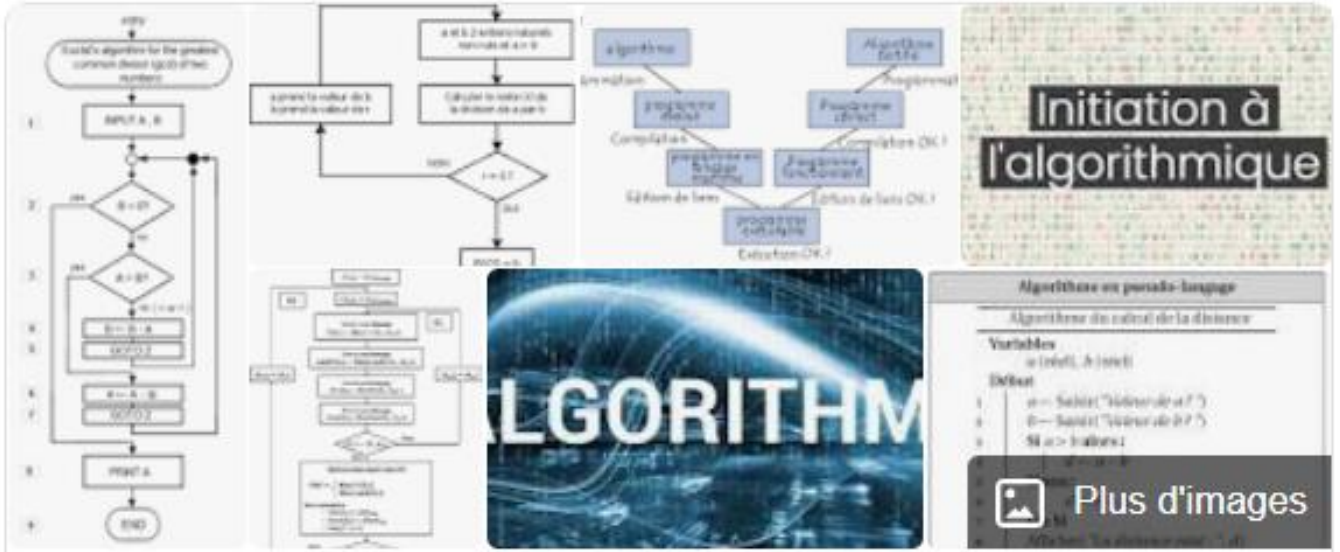
La programmation informatique

Les différentes phases de l'analyse et de la programmation informatique (implémentation sous forme de programme informatique d'un algorithme ou d'un procédé bien formalisé):

1. **Analyse du problème**
2. **Conception d'une solution** : algorithmique, choix de la représentation des données, choix de la méthode utilisée,
3. **Développement** : programmation, choix du langage de programmation, choix de la machine utilisée
4. **Tests**

La programmation informatique: du problème au programme en passant par l'algorithme

1. **Analyse du problème**: L'analyse consiste à bien comprendre l'énoncé du problème: Il est inutile et dangereux de passer à la phase suivante si vous n'avez pas bien discerné le problème. La question à se poser : **QU'EST-CE?**
2. **Conception d'une solution:** C'est la phase **algorithmique** où l'on finalise les choix conceptuels, les structures et outils informatiques à utiliser pour résoudre le problème. Les questions : **AVEC QUOI? COMMENT?**
3. **Le développement:** Plusieurs langages de programmation et architectures informatiques seront disponibles pour implémenter **l'algorithme qui ne doit être lié à aucun d'eux**. Il faut choisir ces éléments en restant cohérent avec la solution proposée.
4. **Tests:** Vérifier l'exactitude du comportement de l'algorithme, son bon déroulement. Si l'algorithme ne répond pas parfaitement à toutes les requêtes exprimées dans l'énoncé du problème, retournez à la phase n°1.



Algorithme

Un algorithme est une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre une classe de problèmes. Le mot algorithme vient du nom d'un mathématicien perse du IXe siècle, Al-Khwârizmî. Le domaine qui étudie les algorithmes est appelé l'algorithmique. Wikipédia

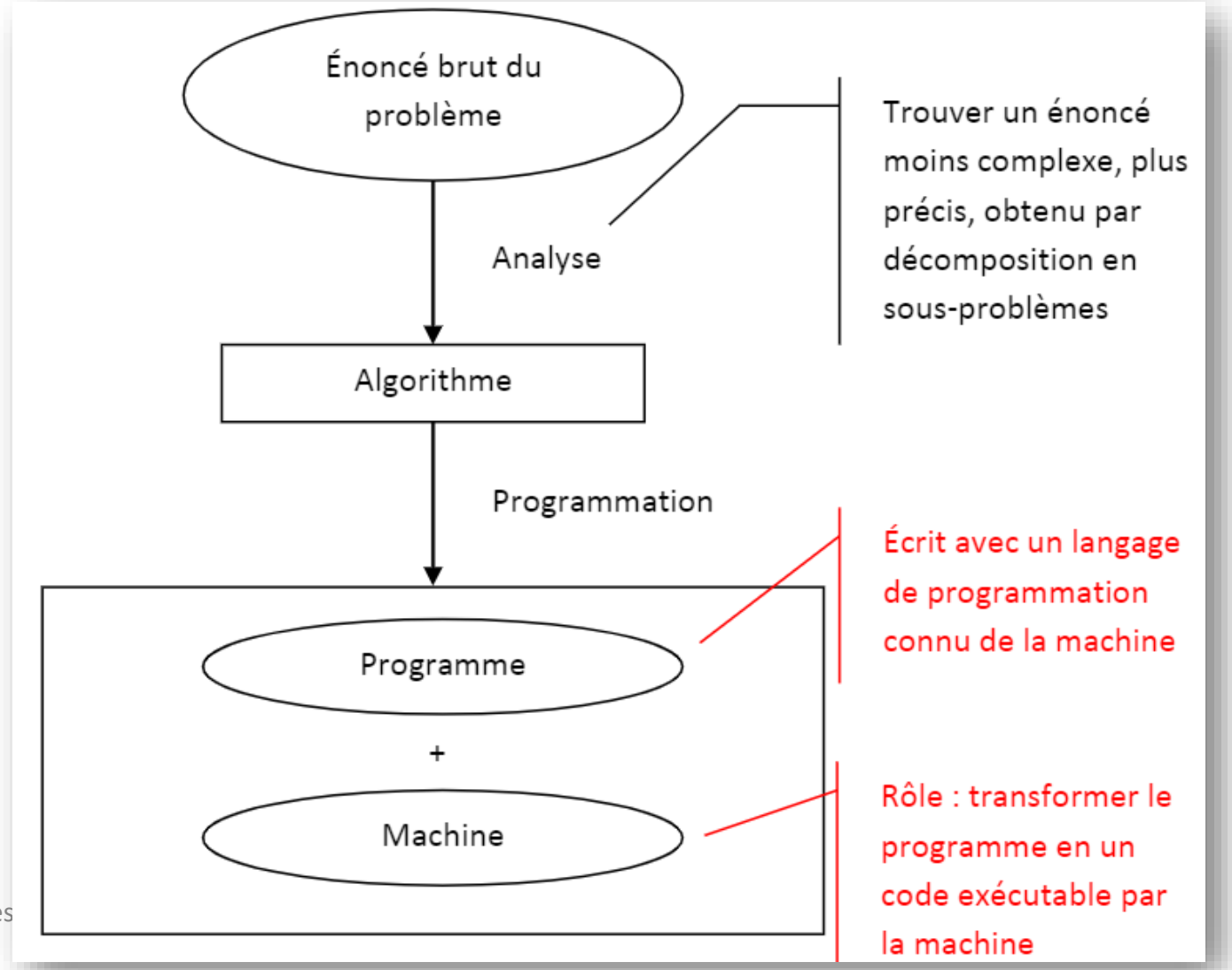
Père : Al-Khwârizmî

Propriétés d'un algorithme

Un algorithme doit être

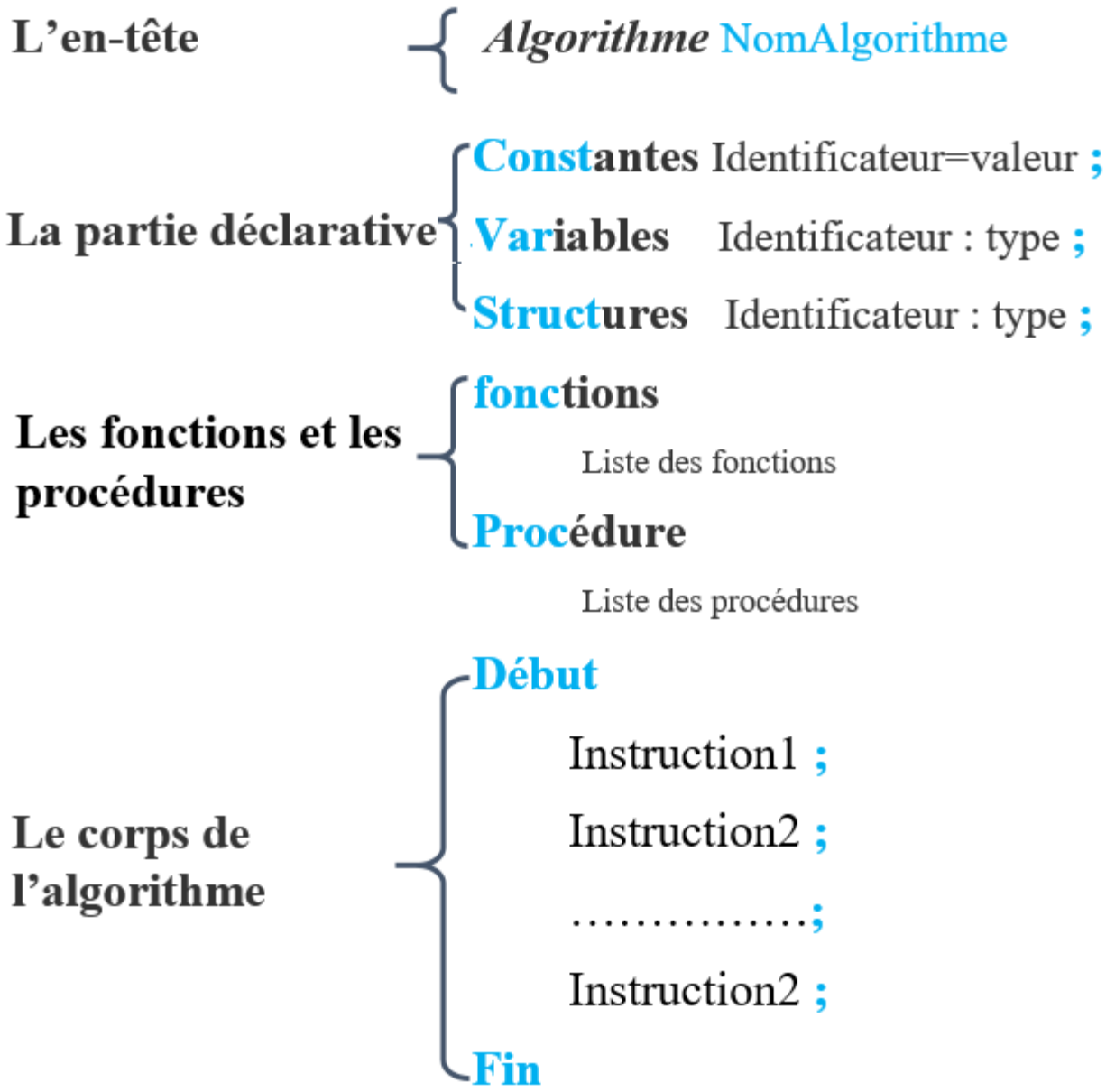
- **PRECIS**: Il doit indiquer:
  - l'ordre des étapes qui le constituent
  - à quel moment il faut cesser une action
  - à quel moment il faut en commencer une autre
  - comment choisir entre différentes possibilités
- **DETERMINISTE**: Une suite d'exécutions à partir des mêmes données doit produire des résultats identiques.
- **FINI DANS LE TEMPS ET PRODUCTIF**: c'est-à-dire s'arrêter au bout d'un temps fini en fournissant le résultat escompté

La programmation informatique: du problème au programme en passant par l'algorithme



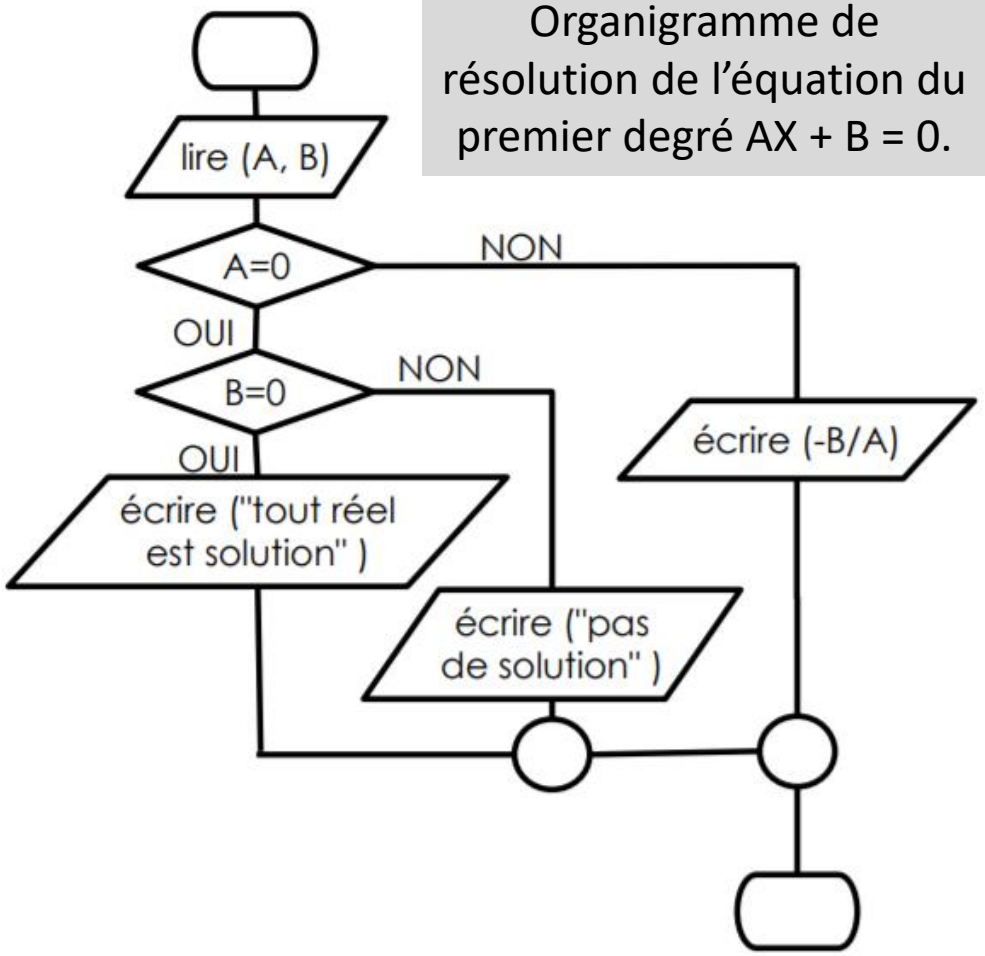
Structure générale d'un algorithme

- **L'en-tête** : permet d'identifier tout simplement l'algorithme (donner un nom en un seul mot ou plusieurs mots composés reliés par – ou _)
- **Les déclarations** : liste exhaustive des objets (constantes et variables principalement), structures et grandeurs utilisés et manipulés dans le corps de l'algorithme. Elle doit être située en début de l'algorithme (tout doit être déclaré avant d'être utilisé)
- **Le corps** : contient les instructions et opérations à exécuter une à une. Elle est délimitée par les mots Début et Fin
- **Les commentaires**: éventuellement présents dans l'algorithme, ils permettent de détailler, de commenter l'algorithme. Ces lignes de commentaires placées entre (* et *) ne sont nullement exécutées, elles sont juste utilisées à titre d'information pour le programmeur (ou pour un autre utilisateur de l'algorithme). (* commentaires *)



Représentation schématique d'un algorithme

- La représentation schématique d'un algorithme est couramment désignée par le terme **organigramme** ou logigramme (norme **NF Z 67-010**).
- L'organigramme permet donc la mise en œuvre de symboles représentant des traitements, des données, des liaisons...
- Il présente l'intérêt d'une visualisation globale mais reste limité aux études peu complexes (s'il ne tient plus sur une page, l'organigramme devient illisible….). En voici quelques symboles:



Organigramme de résolution de l'équation du premier degré AX + B = 0.

Pseudo-code ou LDA (pour Langage de Description d'Algorithmes)

- En programmation, le **pseudo-code**, également appelé **LDA** (pour **Langage de Description d'Algorithmes**) est une façon de décrire un algorithme en langage *presque naturel*, sans référence à un langage de programmation en particulier.
- L'écriture en pseudo-code permet souvent de bien prendre toute la mesure de la difficulté de la mise en œuvre de l'algorithme, et de développer une démarche structurée dans la construction de celui-ci. En effet, son aspect descriptif permet de décrire avec plus ou moins de détail l'algorithme, permettant de ce fait de commencer par une vision très large et de passer outre temporairement certains aspects complexes, ce que n'offre pas la programmation directe.

Source: https://fr.wikipedia.org/wiki/Pseudo-code

Pseudo-code ou LDA (pour Langage de Description d'Algorithmes)

Algorithme Solution_Equation_Seconde _Degre

Variables A, B, C, Delta, X1, X2 : réels;
Début
Lire(A, B, C);
Delta ← $B^2 – 4$ AC;
Si (Delta < 0**) Alors** afficher (" le trinome n'a pas de racine réelle ");
Sinon Si (Delta > 0) **Alors**
X1 ← (-B + racine(delta)) / 2A;
X2 ← (-B - racine(delta)) / 2A;
afficher (" le trinome possède deux racines réelles : ", X1, X2);
Sinon
X1 ← (-B) / 2A;
afficher (" le trinome possède une racine réelle : ", X1);
Fsi
Fsi

Fin

Un exemple d'algorithme

Algorithme ElèveAuCarré

{Cet algorithme calcule le carré du nombre que lui fournit l'utilisateur}

variables unNombre, sonCarre: **entiers**; **{déclarations: réservation d'espace-mémoire}**

début

{préparation du traitement}

afficher("Quel nombre voulez-vous élever au carré?");

Lire(unNombre);

{traitement : calcul du carré}

sonCarre ← unNombre * unNombre;

{présentation du résultat}

afficher("Le carré de ", unNombre);

afficher("c'est ", sonCarre);

fin

Trois étapes d'un algorithme(Corps)

- **Préparation du traitement**
  - données nécessaires à la résolution du problème
- **Traitement**
  - résolution pas à pas, après décomposition en sous-problèmes si nécessaire
- **Edition des résultats**
  - impression à l'écran, dans un fichier, etc.

Notion de variable

- Dans les langages de programmation une **variable** sert à stocker la valeur d'une donnée
- Une variable désigne en fait un **emplacement mémoire** dont le contenu peut changer au cours d'un programme (d'où le nom variable)
- **Règles** : Les variables doivent être déclarées avant d'être utilisées, elle doivent être caractérisées par
  - un **nom** (Identificateur)
  - un **type** (entier, réel, caractère, chaîne de caractères, …)

Choix des identificateurs (1)

Le choix des noms de variables est soumis à quelques règles qui varient selon le langage, mais en général:

- Un nom doit commencer par une lettre alphabétique

  Exemple valide: **A1** Exemple invalide: **1A**

- Doit être constitué uniquement de lettres, de chiffres et du soulignement (Eviter les caractères de ponctuation et les espaces)

  Valides: **VAR2019_2020**, **VAR_2020** Invalides: **VAR 2019**, **VAR-2019**, **VAR;2019**

- Doit être différent des mots réservés du langage (par exemple en C : **int, float, else, switch, case, default, for, main, return, …**)
- La longueur du nom doit être inférieure à la taille maximale spécifiée par le langage utilisé

Choix des identificateurs (2)

- **Conseil**: pour la lisibilité du code choisir des noms significatifs qui décrivent les données manipulées

  exemples: TotalVentes2021, Prix_TTC, Prix_HT

- **Remarque**: en pseudo-code algorithmique, on va respecter les règles citées, même si on est libre dans la syntaxe

Types des variables

- Le type d'une variable détermine l'ensemble des valeurs qu'elle peut prendre, les types offerts par la plus part des langages sont:
- **Type numérique (Entier** ou **Réel)**
  - **Byte** (codé sur 1octet): de 0 à 255
  - **Entier court** (codé sur 2 octets) : -32 768 à 32 767
  - **Entier long** (codé sur 4 ou 8 octets)
  - **Réel simple précision** (codé sur 4 octets)
  - **Réel double précision** (codé sur 8 octets)
- **Type logique ou booléen**: deux valeurs **VRAI** ou **FAUX**
- **Type caractère**: lettres majuscules, minuscules, chiffres, symboles, …

  Exemples: 'A', 'a', '1', '?', …
- **Type chaîne de caractères**: toute suite de caractères,

  Exemples: "SALL, Ousmane", "code postale: 1000", …

Déclaration des variables

- **Rappel**: toute variable utilisée dans un programme doit avoir fait l'objet d'une déclaration préalable
- En pseudo-code, on va adopter la forme suivante pour la déclaration de variables

Variables liste d'identificateurs : **type**;

- **Exemple:**

```
Variables  i, j, k : entier;
           x, y : réel;
           OK: booléen;
           ch1, ch2 : chaîne de caractères;
```

- **Remarque**: pour le type numérique on va se limiter aux entiers et réels sans considérer les sous types

Déclaration de constantes

- Représentent des nombres, des chiffres, des caractères, des chaines de caractères. Une fois initialisées, leurs valeurs ne peuvent pas être modifiées tout au long de l'exécution de l'algorithme.

Mot clé: **constante**

- Exemple:

Constante pi ← **3.14 ;**

L’instruction d’affectation

- **L’affectation** consiste à attribuer une valeur à une variable: consiste en fait à remplir où à modifier le contenu d'une zone mémoire
- En pseudo-code, l'affectation se note avec le signe ←

Var← e : attribue la valeur de e à la variable Var

  - **e** peut être une valeur, une autre variable ou une expression
  - **Var** et e doivent être de même type ou de types compatibles
  - l’affectation ne modifie que ce qui est à gauche de la flèche

- **Exemples valides:**

i ←1;	**j ←i;**	**k ←i+j;**
x ←10.3;	**OK ←FAUX;**	**ch1 ←"LMI";**
ch2 ←ch1;	**x ←4;**	**x ←j;**

(voir la déclaration des variables dans le transparent précédent)

- **Exemples non valides: i ←10.3; OK ←"LMI"; j ←x;**

Quelques remarques

- Beaucoup de langages de programmation (C/C++, Java, C, …) utilisent le signe égal = pour l'affectation ←.
- Attention aux confusions:
  - l'affectation n'est pas commutative : A ← B est différente de B ← A
  - l'affectation est différente d'une équation mathématique :
    - A ← A+1 a un sens en langages de programmation
    - A+1 ← 2 n'est pas possible en langages de programmation et n'est pas équivalente à A ← 1
- Certains langages donnent des valeurs par défaut aux variables déclarées. Pour éviter tout problème il est préférable d'**initialiser les variables** déclarées

Exercices simples sur l'affectation (1)

Exercice 2: Donnez les valeurs des variables A, B et C après exécution des instructions suivantes ?

```
Algorithme Affectation(1)
   Variables A, B, C: Entier;
Début
     A ← 13;
     B ← 17;
     A ← B;
     B ← A+5;
     C ← A + B;
     C ← B – A;
Fin
```

Exercices simples sur l'affectation (2)

Exercice 3: Donnez les valeurs des variables A et B après exécution des instructions suivantes ?

```
Algorithme Affectation(2)
Variables A, B : Entier;
Début
    A ← 18;
    B ← 25;
    A ← B;
    B ← A;
Fin
```

Les deux dernières instructions permettent-elles d'échanger les valeurs de A et B ?

Exercices simples sur l'affectation (3)

Exercice 4:

Ecrivez un algorithme permettant d’échanger les valeurs de deux variables A et B

Expressions et opérateurs

- Une **expression** peut être une valeur, une variable ou une opération constituée de variables reliées par des **opérateurs**

  exemples: 1, b, a*2, a+ 3*b-c, …
- L'évaluation de l'expression fournit une valeur unique qui est le résultat de l'opération
- Les **opérateurs** dépendent du type de l'opération, ils peuvent être :
  - des **opérateurs arithmétiques**: **+, -, *, /, % (modulo), ^ (puissance)**
  - des **opérateurs logiques**: **NON, OU, ET**
  - des **opérateurs relationnels**: **=, <, >, <=, >= , ≠**
  - des **opérateurs sur les chaînes**: **&** (concaténation)
- Une expression est évaluée de gauche à droite mais en tenant compte de **priorités**

Priorité des opérateurs

- Pour les opérateurs arithmétiques donnés ci-dessus, l'ordre de priorité est le suivant (du plus prioritaire au moins prioritaire)
  - **^** : (élévation à la puissance)
  - ***** , **/** (multiplication, division)
  - **%** (modulo)
  - **+** , **-** (addition, soustraction)

exemple: 2 + 3 * 7 vaut 23

- En cas de besoin (ou de doute), on utilise les parenthèses pour indiquer les opérations à effectuer en priorité

exemple: (2 + 3) * 7 vaut 35

Questions de cours

- Expliquez ce qu'est un algorithme
- Donnez la différence entre l'algorithmique et la programmation
- Quelles sont les trois grandes questions à se poser lors de la conception d'un algorithme?
- Quelles sont les propriétés d'un algorithme?
- Donner la représentation graphique de l'algorithme de recherche du maximum entre deux nombres a et b
- Donner les grandes étapes de la programmation informatique
- Détailler la structure générale d'un algorithme

Questions de cours

- Expliquez ce qu'est une variable et à quoi elle sert ?
- Expliquez la différence entre une variable au sens mathématique et le sens algorithmique
- Quels sont les types classiques de variables ?
- A quoi correspond le types alphanumérique ?
- Quelle est la particularité du type booléen ?
- A quoi correspond l'instruction d'affectation ?
- Listez les différentes catégories d'opérateurs

Travaux Dirigés n°1

Préparer la fichier de TD n°1

Bibliographie et Webographie

- Tapez "cours langage c" sur GOOGLE http://www.google.sn/
- https://openclassrooms.com/fr/courses/4366701-decouvrez-le-fonctionnement-des-algorithmes
- Cours : Algorithmique et Programmation 1 http://foad.ugb.sn/course/view.php?id=267
- https://algo.developpez.com/cours/
- Algorithmique,…
  - Tapez "cours Algorithmique" sur GOOGLE http://www.google.sn/
- …