Introduction à l'Algorithmique - 2

A.U: 2019 - 2020

Animé par :

Ibrahim GUELZIM

ibr_guelzim@yahoo.fr

Procédure du cours

- Présence
- Prises de notes de cours
- Diffusion de l'information:
  - **Responsable** de la section
  - facebook : **Dept Info FSA**
  - site web: ibrahimguelzim.atspace.co.uk
  - contact: ibr_guelzim@yahoo.fr

REFERENCES

- Introduction à l'algorithmique. Cours et exercices. Cormen et al. 2e édition. (EN 3rd Edition)

- Algorithms, FOURTH EDITION, Robert Sedgewick and Kevin Wayne. Princeton University.

- Algorithmique Raisonner pour concevoir. Christophe HARO.
- Éléments d'algorithmique. D. Beauquier et al.
- *Informatique pour tous. P. Chatel.*
- Algorithmique, M. El Marraki.

- http://pise.info/algo/

SOMMAIRE

- Rappel
- Tableaux (statiques et dynamiques)
- Pointeurs
- Fonctions et procédures
- Fichiers, Enregistrements et Structures
- Notions sur des structures de données élementaires
- Tas et tri par tas
- La récursivité
- La complexité
- Preuves d’algorithmes
- Applications

ALGORITHME: EXEMPLE 1

- Recette Paella (بقيّة)

Pour 6 personne(s)

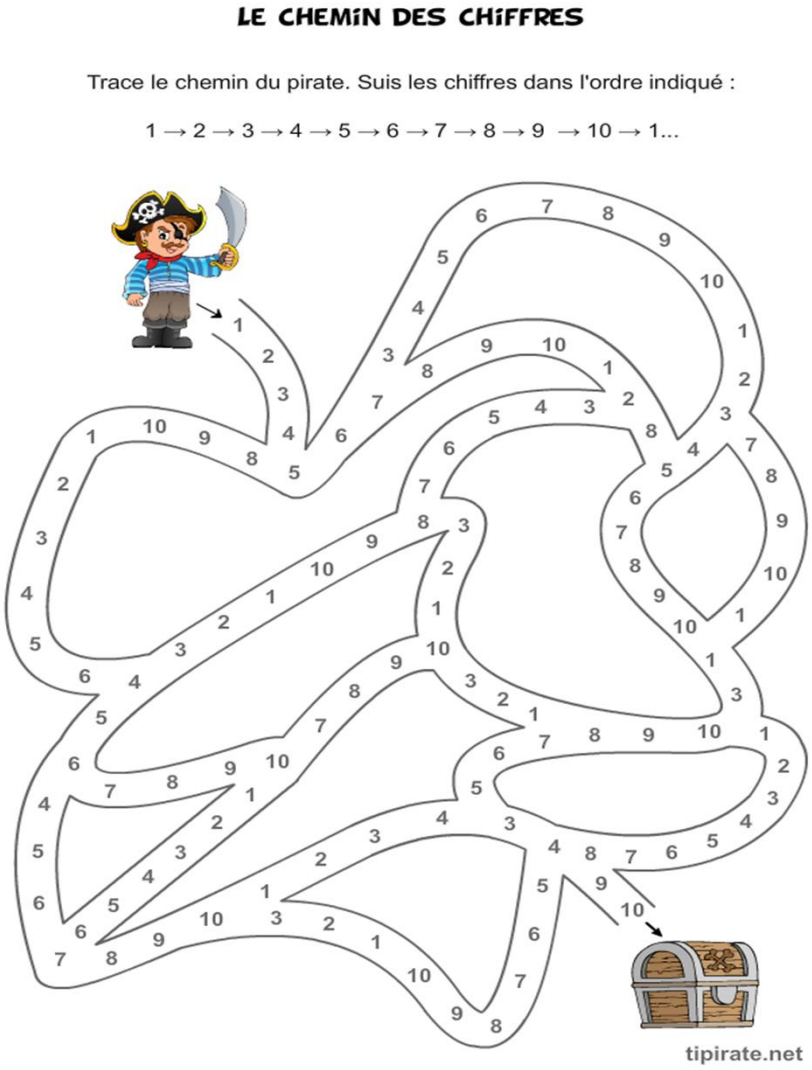
1/2 kg de crevettes
1/2 poulet de taille moyenne
1/2 kg de riz
1/2 kg de petit pois
1/2 kg de tomates concassées
1 poivron vert
1 poivron rouge
1 poivron piquant
2 gros oignons
8 gousses d'ail
2 feuilles de laurier
1 bouquet de persil
1 grand verre d'huile de table
1 citron pour la décoration
10 grosses crevettes pour la décoration
colorant alimentaire
sel, poivre

1. Décortiquez les crevettes, faites cuire les carcasses dans de l'eau salée, filtrez pour obtenir un bouillon de crevettes.
2. Dans un faitout, versez 1/2 verre d'huile, ajoutez 1 oignon émince, le poulet coupé en morceaux, le persil haché, 1 feuille de laurier, du sel, du poivre et le colorant et faites cuire 30 min
3. Dans une poêle, versez 1/2 verre d'huile, faites revenir 1 oignon émince, les poivrons coupés en lamelles, ajoutez les tomates concassées, les petits pois, la feuille de laurier, le persil haché, l'ail haché et les crevettes décortiquées, salez et poivrez et laissez cuire 10 min.
4. Préchauffez le four th.6 (180°C)
5. Ajoutez le riz et le colorant alimentaire, mélangez bien le tout.
6. Ajoutez le poulet cuit avec la sauce à cet appareil et arrosez avec le bouillon de crevettes et de l'eau et laissez cuire à feu doux
7. Versez dans un plat, décorez avec des grosses crevettes et enfournez pendant 20 min.
8. Servez décoré de quartiers de citron.

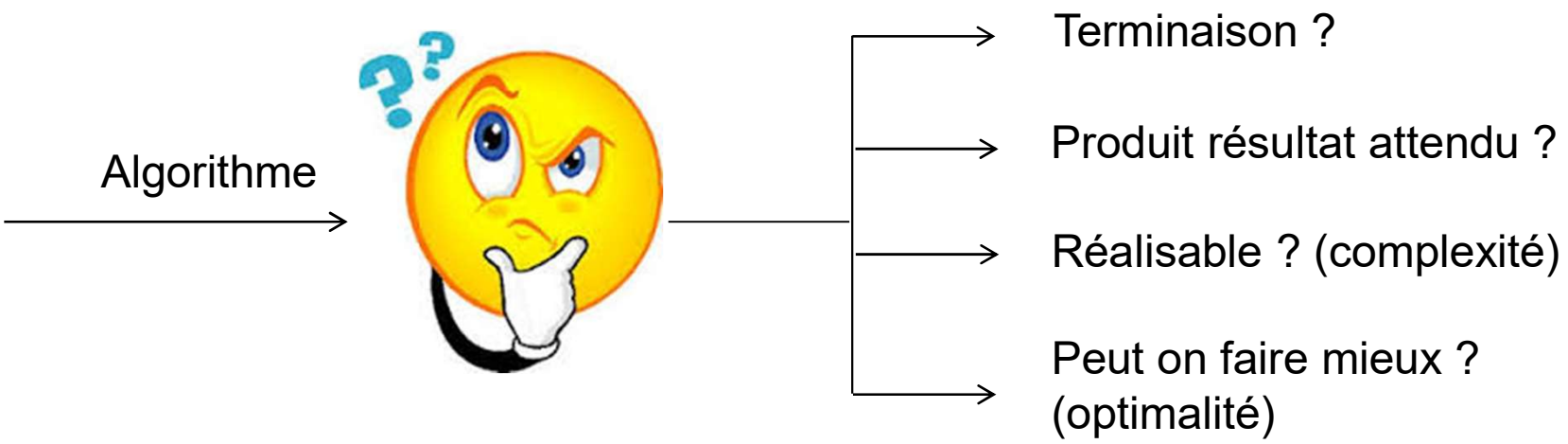
- http://www.une-recette.com/paella-a-la-marocaine.htm

ALGORITHME: EXEMPLE 2

- Labyrinthe



Objectif Ultime du cours



RAPPEL

- Algorithmique :
  - Hommage à الخوارزمي [780 – 850] :

    Savant mathématicien perse (Khawarezm ∈ Khorasan → Baghdad pendant la dynastie abbaside)

  - Alkhawarizmi → Algoritmi →Algorithmique

  - **Art** de décrire une tâche (chemin, recette…)

  - **Art** de mettre les idées en ordre: penser avant d'agir.

  - Mise de **l'ordre** dans la pensée: gain considérable de temps.

RAPPEL : ALGORITHME :: DÉFINITIONS

- **une suite d'étapes** dont le but est de décrire la résolution d'un problème ou l'accomplissement d'une tâche.

- une méthode de résolution de problème énoncée sous la forme d'une série d'opérations à effectuer dans un **ordre bien défini**.

- Procédé permettant de résoudre un problème en un **nombre fini** d'opérations.

- une suite **finie** et **non ambigüe** d'opérations permettant de résoudre un problème.

RAPPEL

- Algorithme sur machine:
  - *Objectif: donner à la machine la possibilité de **substituer** l'homme pour accomplir une tâche dans un **temps relatif bien réduit**.*
  - *Comment ?*
    - Décrire (analyser et concevoir) la tâche d'une manière compréhensible par la machine.
    - Décomposer la tâche en opérations élémentaires.
    - Traduire l'algorithme en un langage de programmation → Programme.

RAPPEL

Entrée → *Algorithme* → Sortie

Programme

- Langage de Programmation
- Ordinateur

Algorithme

- Pseudo code
- Model of computation: mesurer la complexité d'un algo en temps d'execution et en espace mémoire

QUESTION

- Q: Utilité de l'algorithmique ?!
- R: exemples
  - Internet: Recherche de routes optimales pour l'acheminement des données.
  - Commerce électronique: La cryptographie s'appuient sur des algorithmes numériques pour préserver la confidentialité.
  - Une compagnie pétrolière veut savoir où placer ses puits de façon à maximiser les profits.
  - …

ANALYSE ET CONCEPTION DES ALGORITHMES

- Analyser un algorithme :
  - Prévoir les ressources nécessaires à cet algorithme.
    - la mémoire,
    - le processeur,
    - mais, le plus souvent, c'est le temps de calcul qui nous intéresse.

- Plusieurs algorithmes peuvent en résulter

  → éliminer les algorithmes inférieurs et garder la meilleure solution.

- Analyse du meilleur cas, du plus défavorable, ou du cas moyen ?

ANALYSE ET CONCEPTION DES ALGORITHMES

- Conception:
  - **Méthode incrémentale**: utilise un processus **itératif** où chaque itération augmente la quantité d'information.

    Ex: Tri par insertion (cf. plus loin)

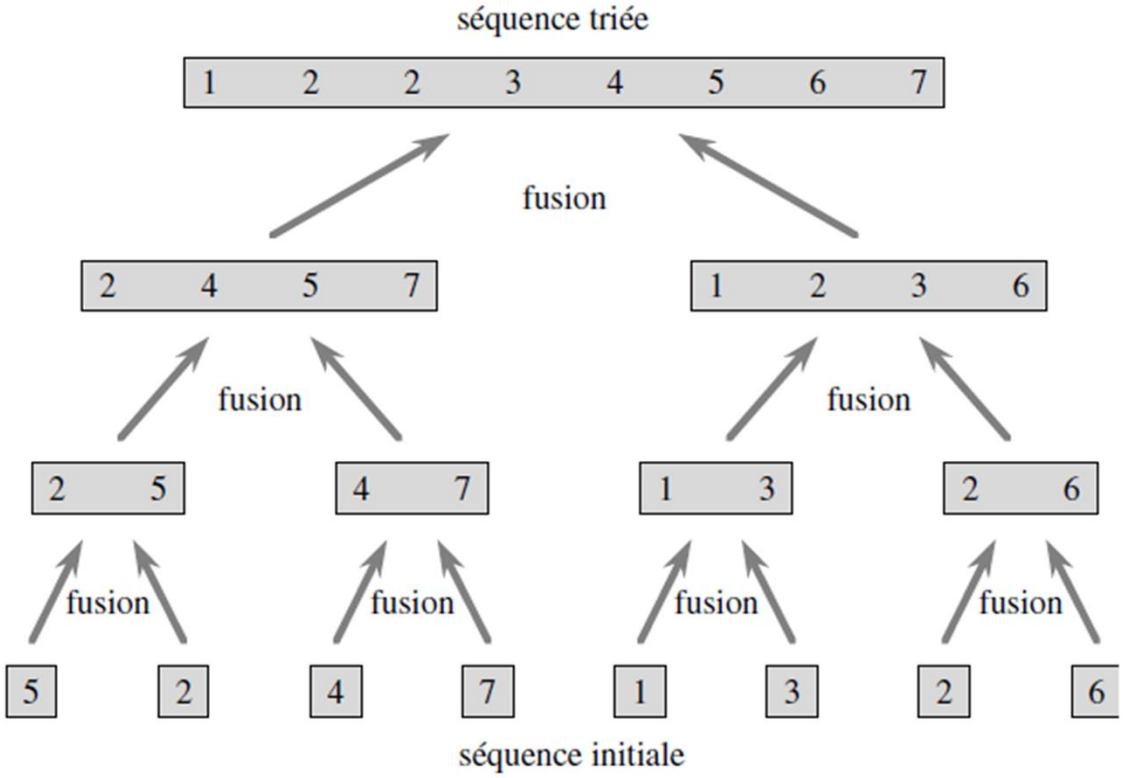
  - **Diviser pour régner**:
    - Analyse Descendante : Décomposer chaque Problème "Composé" en sous Pb (Décomposable ou pas), jusqu'à l'arrivé à des sous Pb élémentaires. (feuille de l'arbre)

    - Analyse Ascendante: Reconstitution de la solution de chaque Pb en regroupant les solutions de ses sous Pb. (idem pour le temps d'execution)

    Ex: Tri par fusion (cf. plus loin)

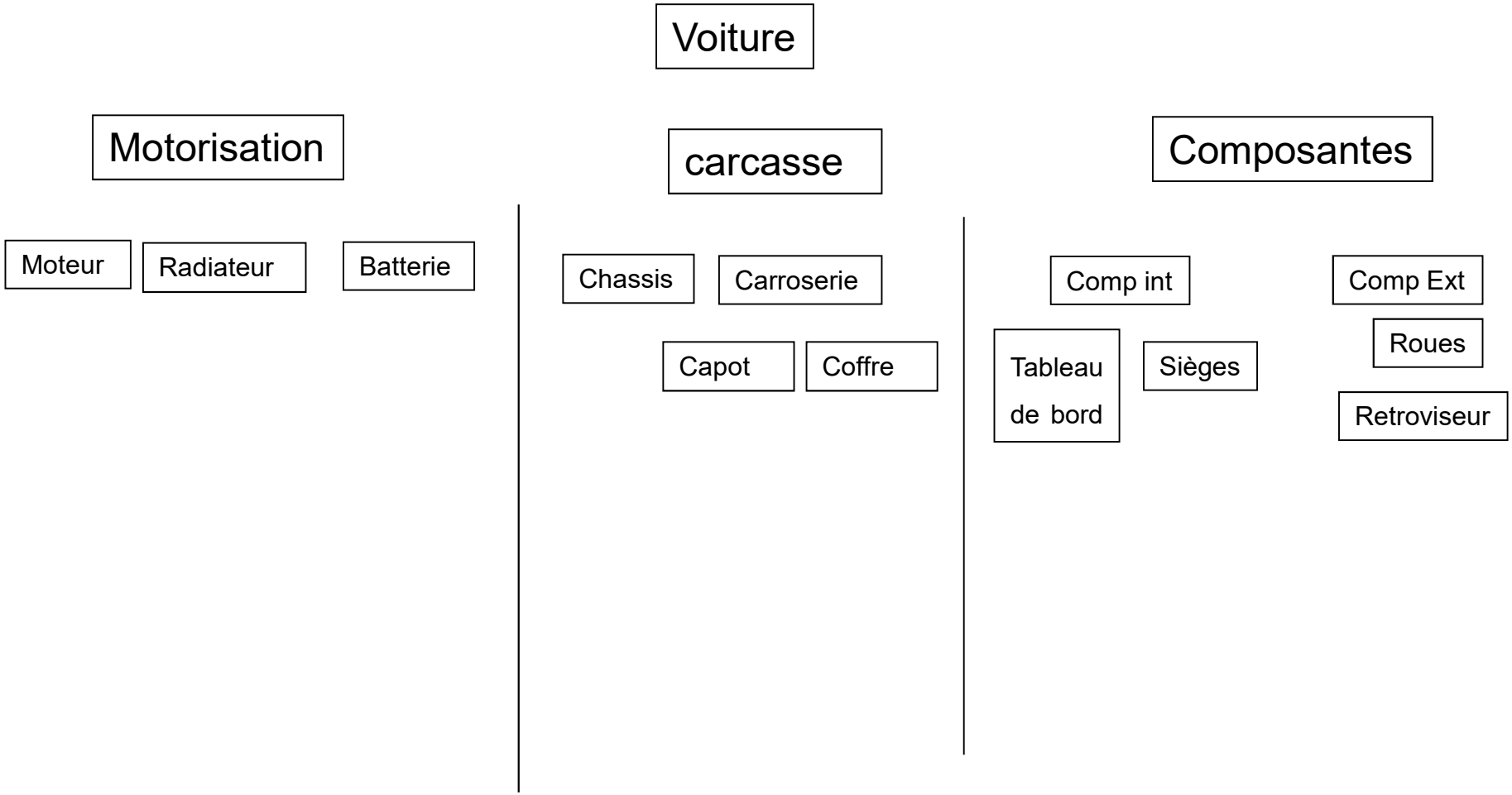
ANALYSE ET CONCEPTION DES ALGORITHMES

- Analyse ascendante :
  - À chaque phase finale (indissociable - indivisible) on passe à la réalisation.
  - Le résultat est fournis à la phase d'avant (niveau plus haut dans l'arborescence)
  - Ex: Tri par Fusion



RAPPEL

- Exemple approximatif: construction d'une voiture



Rappel

- Opérations logiques élémentaires:
  - Affectation
  - Lecture / Écriture
  - Test
  - Boucle

Rappel: Pseudo code

- Pseudo code (Pseudo langage):
  - Permet la lisibilité de l'algorithme
  - Composé d'un en-tête (identification) et d'un corps (description)

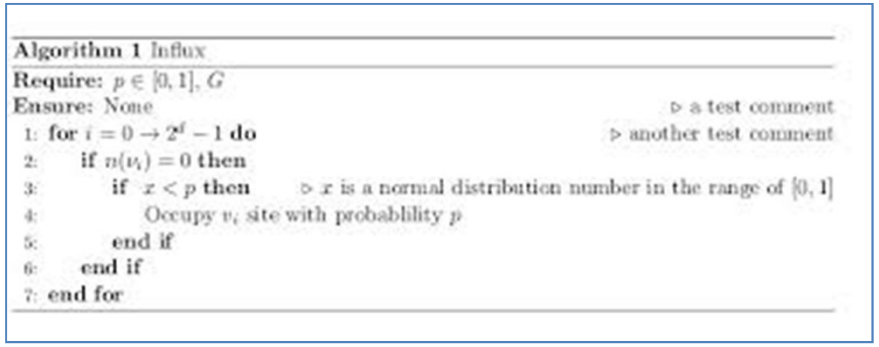
Attention: il n'existe pas une norme d'écriture

Rappel: Pseudo code

- En-tête:
  - Nom de l'algorithme (**Nom :)**
  - Utilité de l'algorithme **(Rôle :)**
  - Données "en entrée": éléments indispensables au bon fonctionnement **(Entrée:)**
  - Données "en sortie": éléments calculés, produits, par l'algorithme **(Sortie :)**
  - Données locales à l'algorithme qui lui sont indispensables **(Déclaration :)**

RAPPEL: PSEUDO CODE

- Corps: contient
  - Mot clé **Début (Begin)**
  - Une suite d'**instructions**
  - Mot clé **Fin (End)**
- Indentation (style): ajout tabulations et espaces pour une **bonne lisibilité du code**

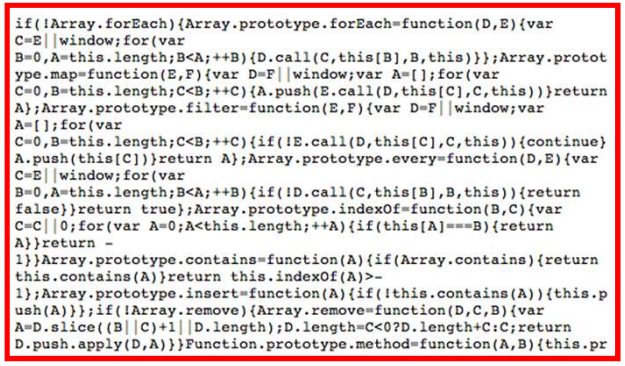


Algorithm 1 Influx

Require: $p \in [0, 1], G$

Ensure: None ▷ a test comment

1: **for** $i = 0 \rightarrow 2^d - 1$ **do** ▷ another test comment
2: **if** $n(v_i) = 0$ **then**
3: **if** $x < p$ **then** ▷ x is a normal distribution number in the range of $[0, 1]$
4: Occupy v_i site with probablility p
5: **end if**
6: **end if**
7: **end for**



```
if(!Array.forEach){Array.prototype.forEach=function(D,E){var
C=E||window;for(var
B=0,A=this.length;B<A;++B){D.call(C,this[B],B,this)}};Array.prototy
pe.map=function(E,F){var D=F||window;var A=[];for(var
C=0,B=this.length;C<B;++C){A.push(E.call(D,this[C],C,this))}return
A};Array.prototype.filter=function(E,F){var D=F||window;var
A=[];for(var
C=0,B=this.length;C<B;++C){if(!E.call(D,this[C],C,this)){continue}
A.push(this[C])}return A};Array.prototype.every=function(D,E){var
C=E||window;for(var
B=0,A=this.length;B<A;++B){if(!D.call(C,this[B],B,this)){return
false}}return true};Array.prototype.indexOf=function(B,C){var
C=C||0;for(var A=0;A<this.length;++A){if(this[A]===B){return
A}}return -
1}}Array.prototype.contains=function(A){if(Array.contains){return
this.contains(A)}return this.indexOf(A)>-
1};Array.prototype.insert=function(A){if(!this.contains(A)){this.p
ush(A)}};if(!Array.remove){Array.remove=function(D,C,B){var
A=D.slice((B||C)+1||D.length);D.length=C<0?D.length+C:C;return
D.push.apply(D,A)}}Function.prototype.method=function(A,B){this.pr
```

Rappel: Pseudo code

- Exemple d'algorithme écrit en pseudo-code

```
Nom          : Le nom de l'algorithme
Role         : que fait l'algorithme
Entrée       : les données nécessaires
Sortie       : les résultats produits par l'algorithme

Variables    : Déclarations des variables
Début
    Instruction 1
    Instruction 2
    …            /* utiliser les commentaires comme aide mémoire */
    Instruction n
Fin
```

RAPPEL: VARIABLES

- Outils d'identification de l'information
- Déclaration: nom + type de la variable
- Nom:
  - Comporte des lettres et des chiffres,
  - Ne comporte pas des signes de ponctuation, en particulier les espaces.
  - Doit commencer par une lettre.
  - Le nombre maximal de caractères dépend du langage utilisé.
  - Ne pas utiliser les **mots clés**.

Rappel: Variables

- **Types:**
  - **Types numériques classiques**
  - **Type alphanumérique**
  - **Type booléen**

- Types numériques classiques
  - Entier
  - Réel

- Ex Pseudo:

variable h : Entier

variables PrixHT, TauxTVA, PrixTTC : Réel

Rappel: Variables

- **Types alphanumériques** (caractère, chaîne)

- contient des caractères:
  - Lettres
  - signes de ponctuation
  - Espaces
  - ou même de chiffres.

RAPPEL: VARIABLES

- Un groupe de caractères est appelé chaîne de caractères.
- En pseudo-code, une chaîne de caractères est toujours notée entre guillemets " ".

❑ Attention à la confusion entre les nombres et les suites de chiffres.

- 123 : représente le **nombre** cent vingt-trois
- "123": représente la **suite** de caractères 1, 2, et 3

- Ex Pseudo Code:
  - variable nom : chaîne
  - variables x, y : caractère.

Rappel: Variables

- **Types booléen** (George Boole → Claude Shannon)
  - Stocker variables logiques: VRAI ou FAUX
  - Peut être représenté par 0 ou 1
- Ex Pseudo Code:
  - variable VF: Booléen

RAPPEL: OPÉRATEURS (BINAIRES)

- signe qui relie deux valeurs pour produire un résultat
- Opérateurs numériques:
  - \+ addition
  - \- soustraction
  - * multiplication
  - / division
  - ^ puissance (3 ^2 vaut 9)
- Opérateur alphanumérique:
  - \+ concaténation
  - , concaténation

  Ex: "Assalamo " + "Alaykom" vaut "Assalamo Alaykom"
- Opérateur d'affectation (←):

  (cf.plus loin)

RAPPEL: EXPRESSION - AFFECTATION

- Opérateurs de comparaison = , <> , > , >= , < , <=

- Opérateurs logique: AND, OR , XOR, NOT (on garde la notation anglaise)
- Les opérateurs booléens «AND» et «OR » sont court-circuitants: Quand on évalue l'expression « x AND y », on commence par x , si elle vaut FAUX, pas la peine de continuer.
  (Attention à l'associativité)

- *Une expression est un ensemble de valeur(s), reliées éventuellement par des opérateurs, équivalent à une seule valeur:*
  - *a + B*
  - som <> A + b

PRIMITIVES: AFFECTATION

- L'affectation est l'opération de donner une valeur à une variable.
- Exprimée par l'opérateur ← (= en C et := en Pascal)
- Prendre la valeur se trouvant du côté droit (rvalue) et la copier dans la variable au côté gauche (lvalue).
  - *Une **rvalue** représente toute constante, variable ou expression capable* de produire une valeur
  - Une **lvalue** doit représenter un emplacement physique pour ranger le résultat:
    - A ← 3 ✓
    - 3 ← A ✗
    - `i ← j ← k` signifie:
      - `j ← k (puis)`
      - `i ← j`
- Remarque: La valeur de l'expression de l'affectation est la valeur affectée.

PRIMITIVES: AFFECTATION

- 1er Exemple: Echanger 2 variables sans utiliser une variable d'aide

  Nom : échange_entiers

  Rôle : échanger deux valeurs entières sans utiliser variable d'aide

  Entrée : x et y

  Sortie : x et y

  Variables x, y : nombres

  Debut

  x ← 5 /* initialisation de x */

  y ← 2 /* initialisation de y */

  x ← x + y

  y ← x - y

  x ← x - y

  Fin

LECTURE ET ECRITURE D'INFORMATIONS

- Lire
- Ecrire
- Exemple:

```
Variable n : Numérique
Début
Ecrire "Saisir un nombre: "
Lire n          /* ou lire(n)       */
Ecrire "Le nbr saisi:  ", n          /* Ecrire "Le nbr saisi:  " + n  */
Fin
```

VALEURS NUMÉRIQUE ET LOGIQUE D'UNE EXPRESSION

- *Rappel : Une expression est un ensemble de valeur(s), reliées **éventuellement** par des opérateurs, équivalent à une seule valeur:*
  - *a + B*
  - *"You get what you work for, not what you wish for"*
  - *w*
  - som <> A + b

- Valeur numérique d'une expression: **valeur arithmétique**
  - Exemples:
    - si une variable x vaut 7, alors sa valeur numérique est 7
    - La valeur numérique de l'expression de l'affectation : s ← 13 est égale à 13

VALEURS NUMÉRIQUE ET LOGIQUE D'UNE EXPRESSION

- Valeur logique d'une expression: **valeur booléenne**
- **Deux possibilités:**
  - Vrai ou 1 : la valeur logique d'une expression est "Vrai" si sa valeur numérique est différente de 0, (y compris les valeurs négatives)
  - Faux ou 0: la valeur logique d'une expression est "Faux" si sa valeur numérique est égale à 0

- Exemples

Expression	Valeur numérique	Valeur logique
ba ← -3	-3	1
s ← a ← 0	0	0
x = 9 **(supposons que x vaut 7)**	0	0
ba AND x (-3 AND 7)	1	1
Mod(17,3)	2	1

TYPE D'UNE EXPRESSION 1/3

- Si l'expression est réduite à une variable, alors son type est le type de la variable,

  Ex: si r est une variable double qui vaut 7.2, alors l'expression r est de type double

- Si l'expression est une opération arithmétique à deux opérandes (pour simplifier), alors le type de l'expression est celui de l'opérande qui a le type le plus fort.

  Ex: - si a est de type entier et x de type réel alors l'expression:

  a + x est de type réel

On parle de **conversion implicite** de la variable a vers le type réel

- le type de l'expression d ← 3 + (a < 10) est Entier

TYPE D'UNE EXPRESSION 2/3

- Le type d'une expression d'affectation est le type de sa **Lvalue**

Exemple:

```
Nom : Type affectation
Variables   a,b : Entiers
            x,y : Réels
Début
a ← 3
x ← 7.2
b ← x
y ← a
Ecrire ("La valeur de b est : ",b," et la valeur de y est: ",y)
End
```

- Le programme ci-dessus affiche :

La valeur de b est : 7 et la valeur de y est: 3.00

TYPE D'UNE EXPRESSION 3/3

- Explications: (autres conversions implicites)
  - `L'expression b ← x est de type Entier:`

  `Conversion implicite (ici avec troncature) de la valeur de x (7.2) à la valeur entière: 7`

  - `L'expression y ← a est de type réél:`

  `Conversion implicite de la valeur de a (3) à la valeur réelle: 3.00`

- Remarques:
  - On peut **forcer la conversion** du type d'une expression en la précédant du type voulu, écrit entre parenthèses:

    Ex: (Réel) a

- Attention: faire la distinction entre (Réel) a/b et (Réel) (a/b)

Rappel: Test

- **Si** (condition) **Alors**
  Bloc Instruction(s)
  FinSi

- **Si** (condition) **Alors**
  Bloc Instruction(s) 1
  Sinon
  Bloc Instruction(s) 2
  FinSi

On évalue la valeur logique de l'expression (Condition)

RAPPEL: TEST IMBRIQUÉS

- **Variable** Temp **: Entier**
  Début
  Ecrire "Entrez la température de l'eau : "
  Lire Temp
  Si Temp <= 0 Alors
    Ecrire "C'est de la glace"
  FinSi
  Si Temp > 0 Et Temp < 100 Alors
    Ecrire "C'est du liquide"
  Finsi
  Si Temp > 100 Alors
    Ecrire "C'est de la vapeur"
  Finsi
  Fin

ET SI C'ETAIT DE LA GLACE ???

Et s'il y avait 723 autres tests à évaluer ???

Rappel: Test imbriqués - Nécessité

- **Variable** Temp **: Entier**
  Début
  Ecrire "Entrez la température de l'eau :"
  Lire Temp
  Si Temp <= 0 Alors
    Ecrire "C'est de la glace"
  Sinon
    Si Temp < 100 Alors
      Ecrire "C'est du liquide"
    Sinon
      Ecrire "C'est de la vapeur"
    Finsi
  Finsi
  Fin

Rappel: Boucles

- Tant que (Tester puis exécuter)

```
TantQue condition Faire
...
Instruction(s)
...
FinTantQue
```

Rappel: Boucles

- Répéter … jusqu'à (Exécuter puis tester)

```
Répéter
...
Instruction(s)
...
jusqu'à condition
```

Rappel: Boucles

- Pour (Nombre d'itérations connu au préalable)

```
Pour i allant de début à fin
...
Instruction(s)
...
FinPour
```

Rappel: Boucles imbriquées

- Boucle dans une boucle
- Exemple: 10 itérations telles que 6 opérations / itération

```
Variables i, j: entier
Pour i allant de 1 à 10
    écrire("Itération : ", i)
    Pour j allant de 1 à 6
      écrire("Opération : ", j, "de l'itération : ",i)
    Finpour
Finpour
```

Rappel: Tableau 1D

- Structure de données statique (taille fixe)
- Un ensemble de valeurs de même type, portant le même nom de variable.
- Le nombre qui sert à repérer chaque élément du tableau s'appelle 'indice'.
- Permet la possibilité du traitement basé sur les boucles.

Rappel: Tableau 1D

- Pour déclarer un tableau il faut préciser le nombre et le type de valeurs qu'il contiendra.

- EX: `Note[35] : réels`
  - Cette déclaration réserve l'emplacement de 35 éléments de type réel.
  - Chaque élément est repéré par son indice (position de l'élément dans le tableau).

- **Convention** : la première position porte le numéro 0
  - `Premier élément du tableau: Note[0]`
  - `Dernier élément du tableau: Note[34]`

Rappel: Tableau 1D - Exemple

```
Nom: Calcul de la moyenne des élements d'un tableau
Variables Note[35],i : entier
      somme, moyenne : réel
Début
pour i allant de 0 à 34 faire          /* saisir les notes */
   ecrire("entrer une note :")
   lire(Note[i])
finpour
somme  ← 0              /* effectuer la moyenne des notes */
pour i allant de 0 à 34 faire
   somme ←  somme + Note[i]
finPour
moyenne ← somme / 35
ecrire("la moyenne des notes est : ",moyenne) /* afficher la moyenne */
Fin
```

Rappel: Tableau 1D

- Applications:
  - Ecrire un algorithme qui cherche une valeur dans un tableau et qui retourne 1 si elle y existe et 0 sinon
  - Et si le tableau était trié ?
  - Ecrire un algorithme qui évalue un tableau et retourne 1 si ce tableau est palindrome et 0 sinon.

Rappel: Tableau 1D Dynamique

- Examinons l'exemple suivant:

```
variable vect[30],i,n : entiers
Début
écrire("donner la taille du tableau (<= 30):")
lire(n)
écrire("donner les éléments du vecteur: ")
pour i allant de 0 à n-1 faire
   lire(vect[i])
Finpour
Fin
```

et si on veut que n > 30 ???

Par mesure de **précaution, remplacer** `vect[30]` par `vect[100]`

et si on veut que n > 100 ???

RAPPEL: TABLEAU 1D DYNAMIQUE

- Solution: **Allocation dynamique** de l'espace mémoire.
- Ce n'est qu'au **cours du programme**, qu'on va déterminer la taille de mémoire à allouer.
- **Exemple: (1/2)**

```
Variables Notes[], moyenne, somme : réels
                                i, n : entiers
Début
écrire("entrer le nombre de notes à saisir : ")
lire(n)
              /* allouer n nombres de types réels */
allouer(Notes,n)
              /* saisir les notes */
écrire("entrer les ",n," notes :")
```

RAPPEL: TABLEAU 1D DYNAMIQUE

- **Exemple: (2/2)**

```
pour i allant de 0 à n-1 faire
       lire(Note[i])
Finpour
/* effectuer la moyenne des notes */
somme ← 0
Pour i allant de 0 à n-1 faire
       somme ← somme + Note[i]
FinPour
moyenne ← somme / n
/* affichage de la moyenne */
écrire("la moyenne des ",n," notes est :",moyenne)
/* liberer le tableau Notes */
libèrer(Notes)
Fin
```

Rappel: Tableau 2D

- Pour ranger les notes de **nb** élèves de la classe de :
  - matière 1 → tab_1 (nb colonnes)
  - matière 2 → tab_2 (nb colonnes)
  - . . .
  - matière m → tab_m (nb colonnes)

- Si m est grand ↗ alors manipulation délicate de plusieurs variables.

- **Solution**: rassembler tout dans un seul tableau 2D de taille m lignes et nb colonnes (m x nb)

Rappel: Tableau 2D

- Structure de données statique. (taille fixe)

- Un ensemble de valeurs de **même type,** portant le même nom de variable.

- Pour repérer un élément du tableau (matrice) il **suffit** d'indiquer:
  - 2 positions: indice ligne et indice colonne.
  - 1 position : entre 0 et mxn -1

- Permet la possibilité du traitement basé sur les boucles.

Rappel: Tableau 2D

- Déclaration:

```
Variable nom_tab[nb_lg][nb_col] : type
```

- Initialisation: Pour initialiser un tableau 2D (matrice) on peut utiliser les instructions suivantes :
  - `T1[3][3] ← { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } }`
  - `T2[3][3] ← { 1, 2, 3, 4, 5, 6, 7, 8, 9 }`
  - `T3[4][4] ← { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } }`
  - `T4[4][4] ← { 1, 2, 3, 4, 5, 6, 7, 8, 9 }`

RAPPEL: TABLEAU 2D

Exemples: Lecture, écriture et somme de matrices

- **Lecture d'une matrice :**

```
variable Tableau mat[10][10],i,j,m,n : entiers
Début
écrire("donner le nombre de ligne de la matrice :")
lire(m)
écrire("donner le nombre de colonne de la matrice :")
lire(n)
pour i allant de 0 à m-1 faire
   écrire("donner les éléments de la ",i," ligne:")
   pour j allant de 0 à n-1 faire
       lire(mat[i][j])
   finpour
Fipour
Fin
```

Rappel: Tableau 2D

Exemples: Lecture, écriture et somme de matrices

- **Écriture d'une matrice :**

```
variables Tableau Mat[10][10],i,j,n,k : entier
début
pour i allant de 0 à n-1 faire
pour j allant de 0 à k-1 faire
écrire(Mat[i][j]," ")
finpour
écrire("\n") /* retour à la ligne */
finpour
fin
```

Rappel: Tableau 2D

Exemples: Lecture, écriture et somme de matrices

- **Somme de deux matrices : (1/3)**

```
constante N 20
variables A[N][N],B[N][N],C[N][N],i,j,n : entier
début
écrire("donner la taille des matrices(< 20) :")
lire(n)
/* lecture de la matrice A */
pour i allant de 0 à n-1 faire
   Ecrire ("donner les éléments de la ",i," ligne:")
   pour j allant de 0 à n-1 faire
       lire(A[i][j])
   Finpour
Fipour
```

Rappel: Tableau 2D

Exemples: Lecture, écriture et somme de matrices

- **Somme de deux matrices : (2/3)**

```
/* lecture de la matrice B */
pour i allant de 0 à n-1 faire
   écrire("donner les éléments de la ",i," ligne:")
   pour j allant de 0 à n-1 faire
       lire(B[i][j])
   finpour
fipour
/* la somme de C = A + B */
pour i allant de 0 à n-1 faire
   pour j allant de 0 à n-1 faire
   C[i][j] ← A[i][j]+B[i][j]
   finpour
Fipour
```

Rappel: Tableau 2D

Exemples: Lecture, écriture et somme de matrices

- **Somme de deux matrices : (3/3)**

```
/* affichage de la matrice de C */
pour i allant de 0 à n-1 faire
   pour j allant de 0 à n-1 faire
       écrire(C[i][j]," ")
   finpour
   écrire("\n") /* retour à la ligne */
finpour
Fin
```

Exercice: Lecture, écriture et produit de deux matrices