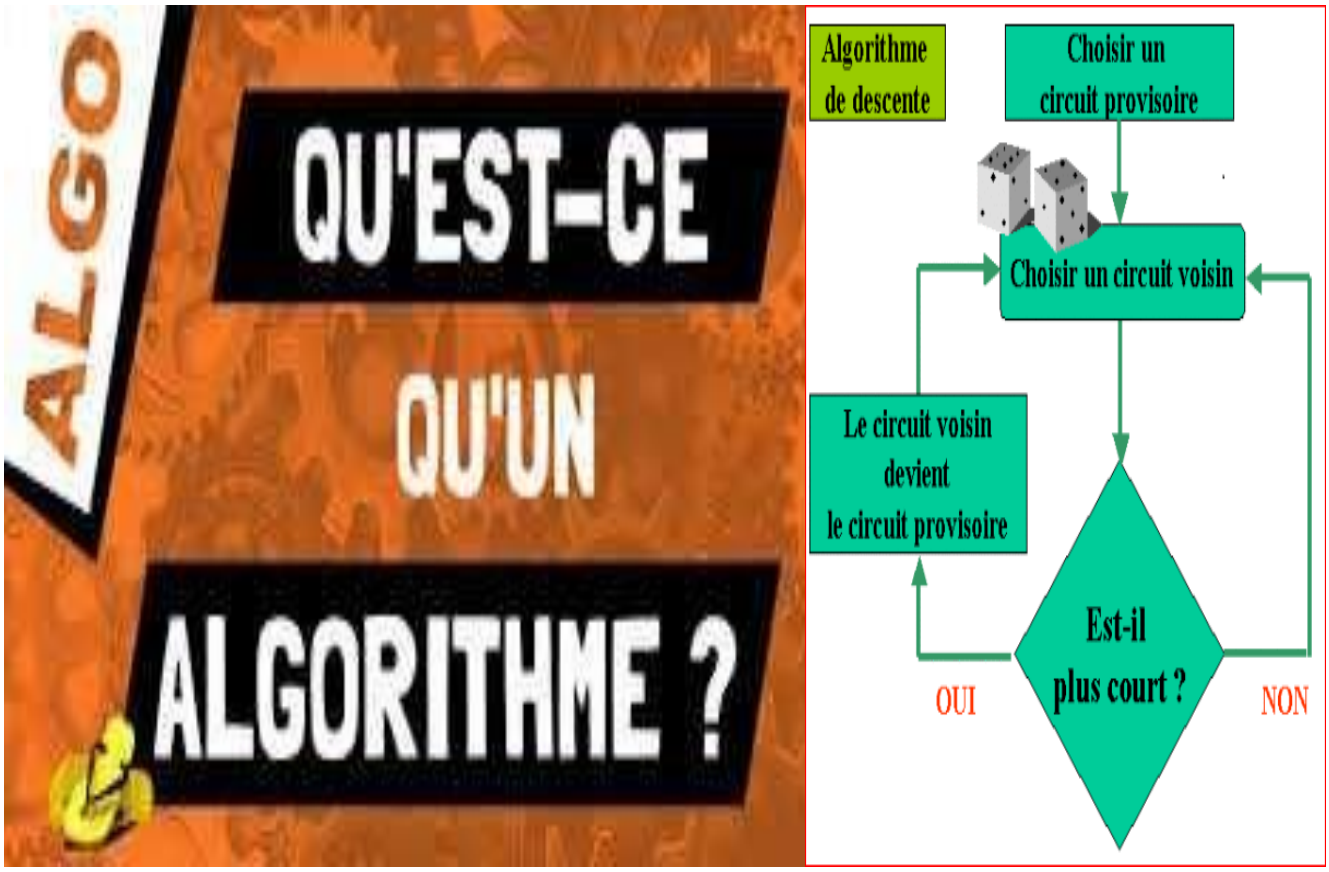
# UNITE D'APPRENTISSAGE 9 :

# EXECUTER DES ALGORITHMES

## Compétences visées :

- ✓ Ecrire les algorithmes pour résoudre des problèmes
- ✓ Exécuter les algorithmes puis interpréter les résultats.




**Leçon 19 :** Notion d'algorithme

**Leçon 20 :** Les instructions

**Leçon 21 :** Les structures et organigrammes

# 19 UNITE D'ENSEIGNEMENT 19 : NOTION D'ALGORITHME

**Objectifs pédagogiques :**

- ✓ Enoncer les étapes de résolution d'un problème ;
- ✓ Identifier et déclarer les variables et les constantes
- ✓ Décrire la structure d'un algorithme

**Contrôle de prérequis :**

1. Lire, analyser et relever les données dans l'énoncé d'un problème.

**SITUATION PROBLEME** :

Votre papa souhaite clôturer son champ rectangulaire avec du grillage. Il veut donc savoir le périmètre exact de son champ pour pouvoir acheter la longueur suffisante de grillage. Rencontrant des difficultés pour la détermination de ce périmètre, votre petit frère lui propose d'écrire un algorithme qui effectuera automatiquement la tâche. Mais votre papa ne s'y connait pas. Il fait donc appel à vous dans le but de l'aider à savoir plus sur les algorithmes.

**Consignes :**

1. Selon vous que faire pour résoudre un problème précis ?
2. Enumérer quelques étapes de la résolution d'un problème.
3. Comment appelle-t-on la démarche à suivre pour arriver au résultat dans la résolution d'un problème précis ?
4. Quelles sont les données qu'a besoin votre papa pour le calcul de ce périmètre ? Comment les appellent-on en langage algorithmique ?

5. Un algorithme doit être constitué de 03 parties. Lesquelles ? RESUME

Un **algorithme** est une suite ordonnée d'instructions qui indique la démarche à suivre pour résoudre un problème précis. La science qui étudie les algorithmes est l'algorithmique.

**Exemple :** Suivre une recette de cuisine, calculer une somme, tracer une figure dans le plan… sont autant d'activités pour lesquelles une série d'actions sont à effectuer une ou plusieurs fois afin d'obtenir un résultat.

Face à un problème quelconque, nous devons nous poser au préalable un certain nombre de questions. La réponse a ces questions facilitera la résolution du problème c'est-à--dire aboutir à un résultat. Les étapes de re solution d'un problème sont donc les suivantes :

- Comprendre l'énoncé du problème
- Décomposer le problème en sous-problèmes plus simple à résoudre
- Associer à chaque sous problème, Les données nécessaires
- Elaborer la démarche à suivre pour arriver au résultat en partant d'un ensemble de données. (Algorithme).

Les objets (ou données) qui seront manipulés dans l'exécution d'un algorithme sont entre autres des variables et des constantes.


**Une constante** : représente des objets (nombre, caractères,) dont la valeur ne peut pas être modifiée pendant l'exécution de l'algorithme.

**Les variables** : représentent les objets dont la valeur peut être modifiée au cours de l'exécution de l'algorithme.

**Remarque :** Une constante est une variable dont la valeur est fixée.

Pour utiliser les variables dans un algorithme, il faut connaitre leurs caractéristiques que sont :

- **L'identificateur (**c'est le nom de la variable ou de la constante. IL est composé de lettres et de chiffres**)**
- **La valeur (**c'est la valeur prise par la variable.**)**
- **Le type (**c'est la nature de la variable utilisée. Ce sont : l'entier, caractère, chaine des caractères, le booléen et le réel**)**

L'utilisation d'une variable dans un algorithme nécessite sa déclaration.

- Pour déclarer une variable, on utilise le mot clé **Var** ou **Variable.** La syntaxe est la suivante :

**Var NomVariable** : type ; ou **Variable NomVariable** : type ;

**Exemples** : **Var Nombre** : entier ; **Variable Nom** : chaine des caractères ;

- Pour déclarer une constante, on utilise le mot clé **Const** ou **Constante.** La syntaxe est la suivante :

**Const NomConstante** =valeur ; ou **Constante NomConstante** = valeur ;

**Exemples** : **Const cote** = 4 ; **Constante Nom** = " Touza Isaac ";

Un algorithme peut être écrit en utilisant un langage de description d'algorithme (LDA). Ce langage utilise un ensemble de mots clés et de structures permettant de décrire de manière complète et claire l'ensemble des opérations à exécuter sur des données pour obtenir des résultats. Avant d'écrire un algorithme, il est nécessaire de connaitre tout d'abord ses différentes parties. Un algorithme a généralement 03 parties :

### 1. L'entête

Elle permet tout simplement d'identifier l'algorithme en précisant son nom. La syntaxe est la suivante : **Algorithme** Nom Algorithme.

**Exemple :** L'en-tête d'un algorithme permettant de préparer un gâteau est **: Algorithme** gâteau.

**NB :** le nom de l'algorithme doit être écrit en un seul mot ou utiliser le « _ » comme séparateur des mots et non le tiret (-).

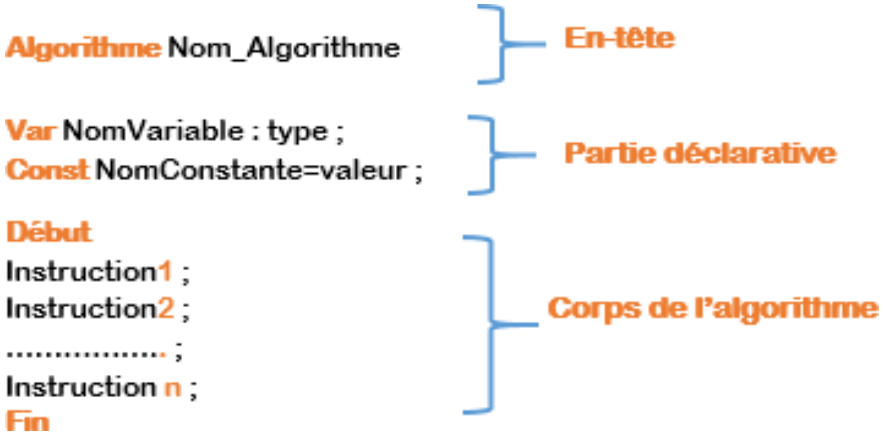
### 2. La partie déclarative

Elle présente la liste exhaustive des objets, grandeurs utilises et manipules dans l'algorithme. Ces objets peuvent être des constantes ou des variables.

### 3. Le corps d'algorithme :

Il est de limite par les termes **DEBUT** et **FIN**. Il contient un ensemble d'instructions (les tâches) à exécuter selon un ordre précis.

En résumé, la structure d'un algorithmique est donne par le schéma ci-dessous :




## SITUATION D'INTEGRATION :

Votre petite sœur souhaite apprendre à préparer les omelettes. Elle se rapproche donc de vous pour que vous l'aidiez à réussir cette tâche. En ayant des connaissances sur l'algorithmique, répondre aux questions suivantes :

1. Comment appelle-t-on les étapes de la résolution d'un problème ?
2. Donner la liste des ingrédients à utiliser pour la préparation des omelettes.
3. Comment appelle-t-on ces ingrédients en langage algorithme ?
4. Donner ses caractéristiques
5. Donner une liste des 05 taches à effectuer pour réussir un plat d'omelettes.

## REINVESTISSEMENT

Vous voulez aider votre ami à mieux résoudre le problème suivant : « passer un appel téléphonique ».

1. De quoi avez-vous besoin pour résoudre ce problème ?
2. Quel nom donne-t-on à ces éléments ?
3. Décrire toutes les taches possibles (classées par ordre d'exécution) que doit réaliser votre ami pour résoudre ce problème.

# 20 UNITE D'ENSEIGNEMENT 20 : LES INSTRUCTIONS SIMPLES

**Objectifs pédagogiques :**

- ✓ Utiliser les instructions simples pour écrire des algorithmes séquentiels

**Contrôle de prérequis :**

1. Donner la structure minimale d'un algorithme
2. Déclarer une variable

**SITUATION PROBLEME** :

Pour automatiser le calcul de la surface d'un terrain triangulaire, votre petit frère décide d'écrire un algorithme pour la réalisation de cette tâche. Dans l'écriture de son algorithme, il souhaite demander les dimensions du champ à l'utilisateur et aussi attribuer les valeurs fournies aux variables et afin il veut aussi afficher le résultat que produira son algorithme à l'utilisateur. Ne s'y connaissant pas trop en algorithmique, votre petit frère demande à cet effet votre aide pour résoudre son problème.

**Consignes :**

1. Donner la formule de calcul de la surface d'un triangle. (**Réponse** : s=(bxh)/2=)
2. Déduire de cette formule la liste des variables qu'on doit-on utiliser dans cet algorithme. (**Réponse** : hauteur (h), base (b) et surface (s) pour contenir le résultat.
3. Relever dans l'énoncé du problème ci-haut trois types de tâches que votre petit souhaite réaliser lors de l'écriture de son algorithme ? (**Réponse** : la lecture des dimensions du champ, la sauvegarde de ces valeurs en mémoire pour y effectuer les opérations et l'affichage du résultat).
4. Comment appelle-t-on ces tâches en langage algorithmique ? (**Réponse** : Les instructions)
5. Pour chacune des tâches énumérées à la question 3, donner une instruction qui lui est associée.

   (**Réponse** :

| Tâches | Instruction |
|---|---|
| La lecture des dimensions du champ | Lire () ou saisir () |
| Sauvegarde de ces valeurs en mémoire | Affectation (le symbole est ← |
| Affichage de résultat | Afficher () ou Ecrire () |

   )

**RESUME**

Le corps de l'algorithme, comporte toutes les taches possibles énumérées selon un ordre précis pour une exécution automatique d'un problème quelconque. Ces taches sont appelées les **instructions**. Elles se terminent toujours par un point-virgule (;). Les instructions de base sont celles de lecture, écriture et affectation.

**La sortie** ou **l'écriture** des données permet l'affichage des valeurs des variables (et/ou texte) après traitement. Elle est notée :

**Afficher (identificateur)** ou **afficher (‘‘ Texte à afficher ’’)** ou encore
**Écrire (identificateur)** ou **écrire (‘‘ Texte à afficher ’’).**

**Exemples :** **Afficher** (a) : affiche le contenu de la variable **a**
**Ecrire** ("J’aime l’info") : affiche le texte suivant : « **J’aime l’info** »

**Remarque :** En sortie, on peut également afficher un message suivi d’une expression ; l’instruction est alors notée **Afficher** (‘‘message’’, expression) ; ou **Ecrire** (‘‘message’’, expression).

**Exemple :**
**Ecrire** ("La somme est", s) : Affiche le texte « **La somme est** » suivi du contenu de la variable s

**L’entrée** ou **la lecture** de données correspond à l’opération qui permet à l’ordinateur d’acquérir des données à partir de l’utilisateur dans les cases mémoires bien définies (qui sont donc les variables déclarées). Les valeurs saisir sont donc de mêmes types que les variables réceptrices. Cette instruction est notée :
**Saisir** (identificateur) ou **lire** (identificateur).

**Exemples :** **Saisir** (a) : stocke dans la variable **a** la valeur fournie par l’utilisateur.
**Lire** (Nom) : sauvegarde dans la variable Nom la valeur du nom saisi au clavier.

**Remarque :** on peut lire également les valeurs de plusieurs variables à la fois. L’instruction devient donc : **Lire** (variable 1 variable 2 …. Variable n) ;

**L’affectation** c’est l’action d’attribuer une valeur à une variable. Cette instruction est notée :

**Variable** prend **valeur** ou **Variable** ← **valeur.**

Le symbole ← est l’opérateur d’affectation.

**Remarque** :

- L’expression **A←B** se lit « A reçoit B » ou encore « A prend B ». Le résultat est de mettre le contenu de la variable B dans la variable A.
- L’affectation ne vide pas la variable émettrice (à droite) de sa valeur mais par contre écrase le contenu de la variable réceptrice (à gauche).

**Exemple d’application :**

L’algorithme qui calcule et affiche la somme de deux nombres entiers saisi au clavier est :

**Algorithme** Somme
**Var** a, b, S: entier ;
**Début**
**Ecrire (**"Entrer le premier nombre") ;
**Lire(**a) ;
**Ecrire (**"Entrer le deuxième nombre") ;
**Lire** (b) ;

S← a+b ;
**Ecrire** ("La somme de ", a, "et" , b ,"est s = ",S ) ;
**Fin.**

## SITUATION D'INTEGRATION

Vous souhaitez écrire un algorithme qui prend en entrée le nom d'une personne et affiche le message : « bonjour suivie du nom de la personne ».

1. Donner la liste des variables à utiliser dans cet algorithme et leurs types.
2. Combien d'instructions de lecture et d'écriture allez-vous utiliser dans cet algorithme ?
3. Ecrire cet algorithme.

## REINVESTISSEMENT

Votre maman qui est vendeuse des beignets a acheté un seau ayant la forme de la figure ci-dessous pour y mettre ses beignets. Connaissant le volume d'un beignet, elle veut savoir la quantité totale des beignets que peut contenir ce seau. Pour cela elle désire écrire un algorithme qui lui permettra de calculer le volume de ce récipient.

On suppose que ce seau a la forme d'un cylindre et que son volume est donné par V=B * H (où B est l'aire du cercle de rayon r).

1. Faire la liste des variables qu'elle doit utiliser dans l'écriture de son algorithme, puis donner pour chaque variable, son type.
2. Décrivez la démarche algorithmique à suivre (en français facile).
3. Ecrire l'algorithme permettant de calculer le volume de ce récipient.
4. **vol** étant le volume d'un beignet et **V** celui du récipient, écrire un algorithme qui permet de trouver la quantité des beignets que peut contenir ce récipient.

# 21 UNITE D'ENSEIGNEMENT 21 : LES STRUCTURES ET ORGANIGRAMME

## Objectifs pédagogiques :

- ✓ Identifier les structures de contrôle ;
- ✓ Utiliser les structures alternatives ;
- ✓ Exécuter un algorithme simple ;
- ✓ Dessiner un organigramme simple

## Contrôle de prérequis :

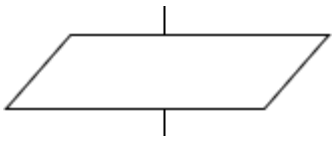
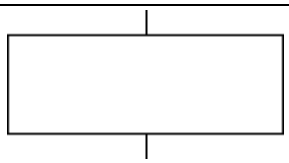
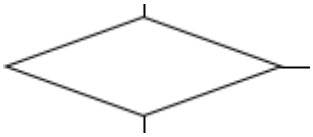
1. Donner la structure minimale d'un algorithme
2. Utiliser les instructions d'écriture, d'affichage et d'affectation pour écrire les algorithmes.

## SITUATION PROBLEME :

Votre grand frère veut aider votre papa à écrire un algorithme lui permettant de calculer les prix de vente de ses articles après son marché. Cet algorithme doit demander d'entrer le type de l'article (chaise, écran et ordinateur) et le nombre d'articles vendus jusqu'à une certaine nombre de fois, puis effectue les calculs en fonction de l'article et de son prix unitaire et enfin affiche le résultat. Pour écrire cet algorithme, votre grand frère a besoin d'utiliser les structures de contrôle. Par ailleurs, pour bien expliquer son algorithme à votre papa, il souhaite représenter cela graphiquement à l'aide des symboles normalisés. Etant donné qu'il ne connait pas grand-chose à propos, il sollicite votre aide dans le but de l'aider à résoudre ce problème.

## Consignes :

1. Citer les structures de contrôle utilisé en algorithmique
2. Comment appelle-t-on la représentation graphique d'un algorithme ?
3. Quelle partie de l'algorithme représente-t-on graphiquement ?
4. Donner le symbole utilisé en algorithme pour représenter les instructions et les mots clés suivantes :

| Instruction / mot clé | Symbole | Instruction / mot clé | Symbole |
|---|---|---|---|
| Lecture/ écriture |  | Affectation |  |
| Début/ fin |  | Condition |  |

## RESUME

Les opérations élémentaires relatives à la résolution d'un problème peuvent, en fonction de leur enchaînement, être organisées suivant quatre familles de structures algorithmiques fondamentales : Structures linéaires, structures alternatives, structures de choix et structure itératives (ou répétitives).

La **structure séquentielle** ou **linéaire** se caractérise par une suite d'actions à exécuter successivement dans l'ordre énoncé. Les actions successives sont mentionnées les unes après les autres.

Dans la **structure alternative (si…alors…sinon)**, l'exécution d'une action distincte ne dépend que du résultat d'un test effectué sur la condition qui peut être une variable ou un événement.

- Si la condition est vérifiée, seule la première action est exécutée ;
- Sinon, seule la deuxième action est effectuée.

**Exemple :** l'algorithme qui étudie le signe d'un nombre est :

```
Algorithme Nombre;
Var x: entier ;
  Début
    Écrire ("Saisir x") ;
    Lire (x);
       Si (x<0) Alors
         Écrire (x, "est négatif") ;
       Sinon
         Écrire (x, "est positif") ;
       FinSi
    Fin
```

**NB** : Le « **Sinon** » n'est pas obligatoire. S'il n'est pas présent, aucune tâche ne sera effectuée si la condition n'est pas remplie. On parle alors de **Structure alternative réduite SI… ALORS…**

Dans la **structure répétitive** les instructions sont répétées un certain nombre de fois. La répétition peut être connue ou non. Elle est subdivisée en plusieurs boucles :

- La boucle **pour** : le nombre de repetition est connu. Sa syntaxe es :
  ***Pour*** *variable* ***allant*** *de valuer_initiale* ***à*** *valeur_finale* ***faire***
  *Action1 ;*
  *Action2 ;*
  *…….*
  ***FinPour***
- La boucle **Tant que** : le nombre de répétition n'est pas connu. On teste une condition à l'entrée de la boucle avant d'exécuter les instructions : c'est une boucle conditionnelle. Sa syntaxe est :
  **Tant que** (condition vraie) **Faire**
  *Action1 ;*

*Action2 ;*
*…….*
***FinTantque***

- La boucle **Répéter** : ici, on exécute au moins une fois le bloc d'instructions avant de tester la condition. Sa syntaxe est :
  **Répéter**
  *Action1 ;*
  *Action2 ;*
  *…….*
  ***Jusqu'à*** (condition) ;

## ORGANIGRAMME

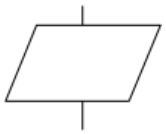
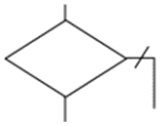
A part le LDA, on peut aussi représenter un algorithme à l'aide des symboles normalisés. Cette représentation est appelée organigramme ou algorigramme.

Un **Organigramme** : est la représentation graphique de l'algorithme. Il permet de représenter chaque opération élémentaire au moyen d'un symbole graphique normalisé. Un organigramme bien représenté doit être fléché et fermé, compris entre un début et une fin, et doit permettre de suivre facilement l'ordre d'exécution des règles de résolution du problème étudié.

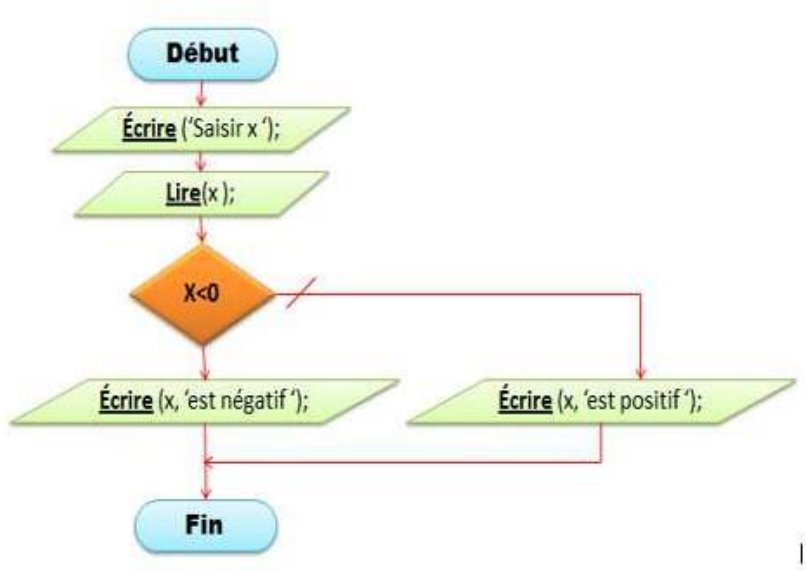
**NB :** Le sens général des lignes de liaison doit être :

- ✓ De haut en bas
- ✓ De gauche à droite
- ✓ Lorsque le sens général ne peut pas être respecté, des pointes de flèches à cheval sur la ligne indiquent le sens utilisé.

Pour construire un organigramme, on utilise des symboles normalisés donnés ci-dessous :

| Symbole | Désignation | Symbole | Désignation |
|---|---|---|---|
|  | Opération pour laquelle il n'existe aucune symbole normalisé |  | Entrée-Sortie |
|  | Début, fin ou interruption d'un organigramme |  | Conditions variables impliquant un choix parmi plusieurs |

**Exemple** : l'algorithme de l'algorithme permettant d'étudier le signe d'un nombre est :




## EXECUTION D'UN ALGORITHME

Exécuter un algorithme revient à exécuter une à une chacune des instructions que constitue cet algorithme et surtout en respectant l'ordre dans lequel ces instructions sont écrites, dans le but de savoir le résultat produit par cet algorithme. Pour exécuter un algorithme, il faut :

- Distinguer une instruction : une instruction dans un algorithme se trouve dans le corps de l'algorithme entre Début et fin et se terminant toujours par un point-virgule (;)
- Connaitre l'ordre de priorité des opérateurs pour effectuer les opérations arithmétiques.
- Donner les valeurs aux variables déclarés dans cet algorithme et remplacer chaque variable par sa valeur au cours de l'exécution puis effectuer les opérations.
- Noter le contenu des variables (Utiliser un tableau si nécessaire) à chaque étape d'exécution de l'algorithme.
- Savoir quelle est la variable de sortie et donner son contenu après l'exécution.

### Exemple d'application :

Exécutons l'algorithme suivant avec les valeurs **a=2, b=5 et c=8**:

**Algorithme** Exemple
**Var** a, b, c,s : entiers ;
**Début**
**Saisir(**a) ;
**Saisir** (b) ;
s← **(**a+b) MOD a * c / a *b;
**Ecrire** ("La valeur finale de s est :", s) ;
**Fin.**

**Solution :**

- Notre algorithme compte 04 instructions
- Les valeurs des variables avec lesquelles on exécutera l'algorithme sont a=2,b=5 et c=8
- La variable à afficher à la sortie est la variable **s**.

Exécutons alors l'algorithme :

a=2

b=5

s = (2+5) MOD 2 * 8 / 2 * 5 ***on remplace chaque variable par sa valeur***

= 7 MOD 2 * 8 / 2 * 5 ***on effectue les opérations par ordre de priorité des opérateurs.***

= 1*8/2*5 ***Les opérateurs ont les mêmes priorités, on effectue les calculs de gauche vers la droite***

= 8/2*5 ***Les opérateurs ont les mêmes priorités, on effectue les calculs de gauche vers la droite***

= 4*5

**s= 20**

L'algorithme affiche donc la valeur de s qui est **s=20.**

## SITUATION D'INTEGRATION

On veut calculer la moyenne du premier groupe des élèves de la 3ème et y attribuer la mention associée. Les coefficients des matières sont les suivantes : Mathématiques : coefficient 4 ; PCT : coefficient 3 et Informatique coefficient 2. Un élève est accrédité d'une mention bien si sa moyenne est supérieure a 14, assez bien si elle est comprise entre 10 et 13.99, et faible si elle est inferieure a 10.

a) Quelles sont les données en entrées et leurs types respectifs ?
b) Écrire un algorithme qui permet de donner la mention d'un élève connaissant ses trois notes du premier groupe.
c) Dessiner l'algorigramme de votre algorithme.

## REINVESTISSEMENT

1. Écrire l'algorithme permettant de lire la valeur de la tempe rature de l'eau et d'afficher son état :

- **GLACE** si la tempe rature inférieure à 0,
- **EAU** si la tempe rature est strictement supérieure à 0 et inferieure a 100,
- **VAPEUR** si la tempe rature supérieure à 00.