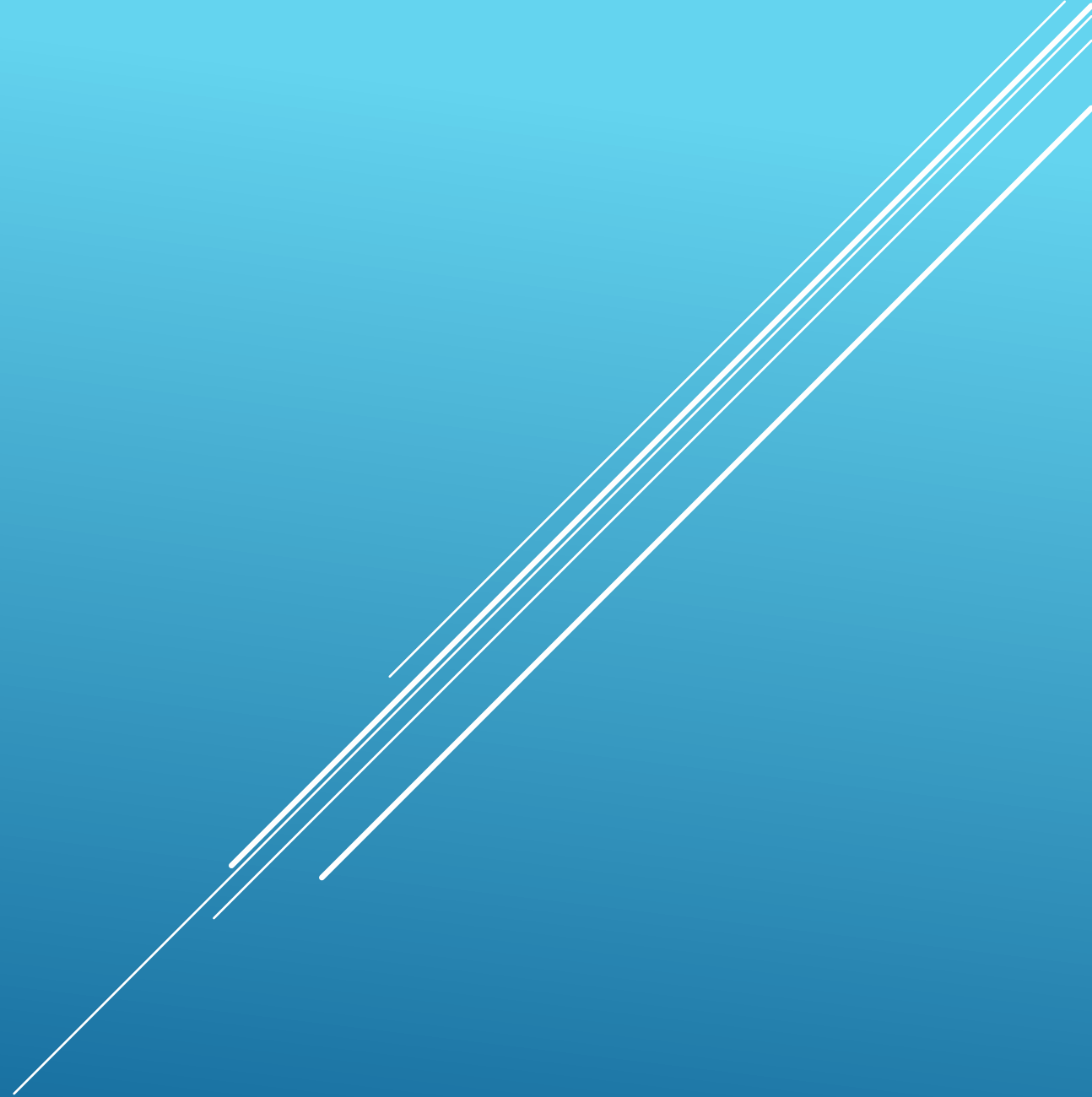


# ALGORITHMES ET PROGRAMMES



# Qu'est-ce que la programmation ?

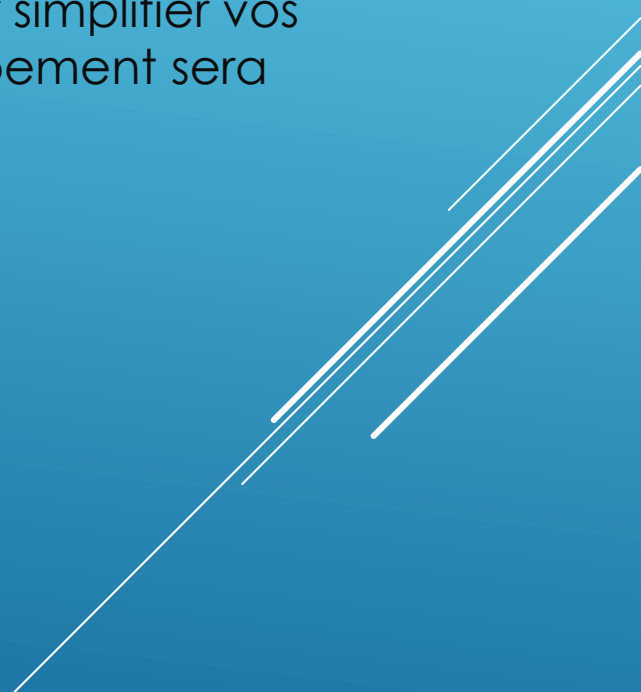
La programmation est avant tout une méthode d'analyse et non l'apprentissage d'un langage.

Pour bien comprendre le fonctionnement d'un programme, il convient de le décomposer en modules indépendants et élémentaires qui se contentent d'effectuer une action spécifique.

En utilisant cette technique, la conception devient aisée, vous faites moins d'erreurs et la maintenance du code est simplifiée.

# Ne pas tout réécrire

- Lorsque vous écrivez un code informatique, vous pouvez bien souvent vous appuyer sur des "bibliothèques" pour éviter de tout réécrire.
- Tous les langages proposent des bibliothèques complémentaires pour simplifier vos développements. N'hésitez pas à les utiliser. Votre temps de développement sera réduit d'autant ...



Un algorithme n'est pas forcément destiné à décrire la solution d'un problème pour la programmation et l'informatique ...

- Un algorithme en cuisine s'appelle une recette
- Un algorithme en musique s'appelle une partition
- Un algorithme en tissage s'appelle un point

Un **algorithme** est donc une suite finie et non ambiguë d'opérations ou d'instructions permettant de résoudre une problématique



# L'analyse

L'analyse est une phase de réflexion préalable qui permet d'identifier précisément le problème :


- données à traiter
- résultats attendus
- cas particuliers
- traitements à effectuer,
- etc.

L'analyse permet également de découper le problème en une succession de tâches simples à enchaîner pour arriver jusqu'à la résolution.

**L'analyse est fondamentale.** En effet, la résolution d'un problème ne peut être dissociée de sa compréhension. Mieux vous comprendrez le problème, plus facilement vous le résoudrez.

# Base de l'Algorithme

La définition d'un algorithme consiste en la mise en œuvre d'actions élémentaires à l'aide d'une notation dédiée :

- Déclaration de variables
  - Lecture / Ecriture
  - Tests
  - Boucles
- 
- Several white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

# Les opérateurs :

## Les opérateurs arithmétiques

Opérateur	Opération
+	L'addition
-	La soustraction
*	La multiplication
:	La division entière
/	La division réelle

pour calculer le reste de la division il existe l'opérateur modulo: %

# Les opérateurs :

## Les opérateurs de comparaison

Opérateur	Opération
==	Egal à
!=	Différent de
<	Inférieur à
<=	Inférieur ou égal à
>	Supérieur à
>=	Supérieur ou égal à

## Les opérateurs logiques

Opérateur	Opération
ET - &&	ET logique
OU -	OU logique inclusif
NON - !	Négation logique
XOR	OU logique exclusif

## Les opérateurs divers

Opérateur	Opération
r	
<-	L'affectation
&	La concaténation



# Les opérateurs :

Le test d'égalité « = » s'écrit == (ex: « x == 5 »)

Différent « ≠ » s'écrit != (ex: x != 5)

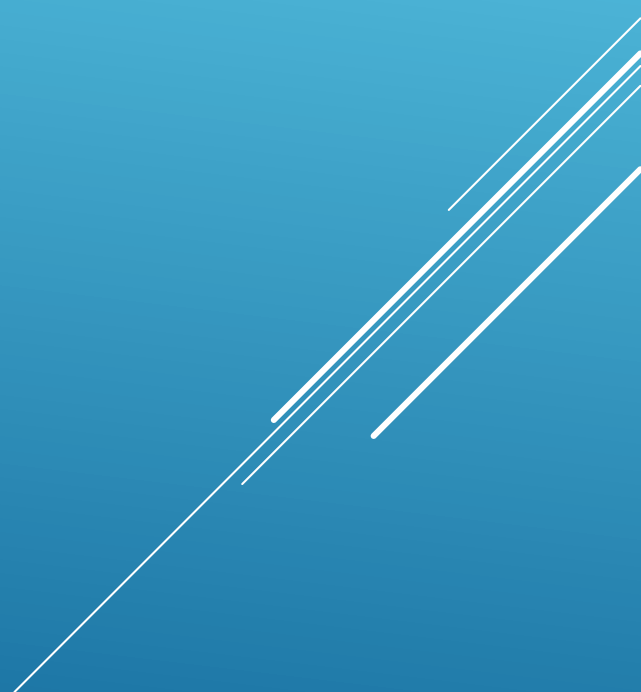
« ≤ et ≥ » s'écrivent <= et >= (ex: x <= 5)

# Ordre des Instructions

L'ordre va de ligne en ligne

Même si c'est logique, on commence par la première ligne, on l'exécute, puis on passe à la suivante, puis à la suivante, jusqu'à la fin de l'algorithme.

Cela s'appelle la synchronicité



# Variable

Une **variable** est une zone mémoire qu'un programme va utiliser pour stocker temporairement une valeur.

Une variable est un triplet (identificateur, type, valeur)

- identificateur : nom, qui permet de la désigner tout au long de l'algorithme.
- type est fixe.
- valeur variera au cours de l'exécution de l'algorithme

Une **constante** est soit une valeur "brute" soit une zone mémoire qu'un programme va utiliser pour stocker temporairement une valeur qui ne changera pas durant l'exécution du programme.

A les mêmes caractéristique qu'une variable, mais ne peut pas être modifié !

# Variable

La première chose à faire pour utiliser une variable, c'est de la déclarer en précisant son type!

cela permet à l'ordinateur de créer l'espace (le contenant) où sera mémorisé la valeur de celle-ci.

cette étape permet de préparer l'espace mémoire suffisant en fonction du type de la variable. En effet celui-ci ne sera pas le même si la variable est un entier ou une chaîne de caractères.



# Variables: les différent types

## Type Numérique

byte (octet)

Entier simple

Entier long

Réel simple

Réel double

## Plage

0 à 255

-32 768 à 32 767

-2 147 483 648 à 2 147 483 647

$-3,40 \times 10^{38}$  à  $-1,40 \times 10^{45}$  pour les valeurs négatives  
 $1,40 \times 10^{-45}$  à  $3,40 \times 10^{38}$  pour les valeurs positives

$1,79 \times 10^{308}$  à  $-4,94 \times 10^{-324}$  pour les valeurs négatives  
 $4,94 \times 10^{-324}$  à  $1,79 \times 10^{308}$  pour les valeurs positives

Attention pour une raison d'économie de ressource, il est important d'utiliser le bon type numérique!

# Variables: les différent types

Le type caractère annoter entre simple quote : 'c'

Le type chaine de caractère (String)

Il est possible de concaténer deux ou plusieurs variable de type alphanumériques:

a ← « hello »

b ← « world »

c ← a & « » & b → c aura pour valeur « hello world »

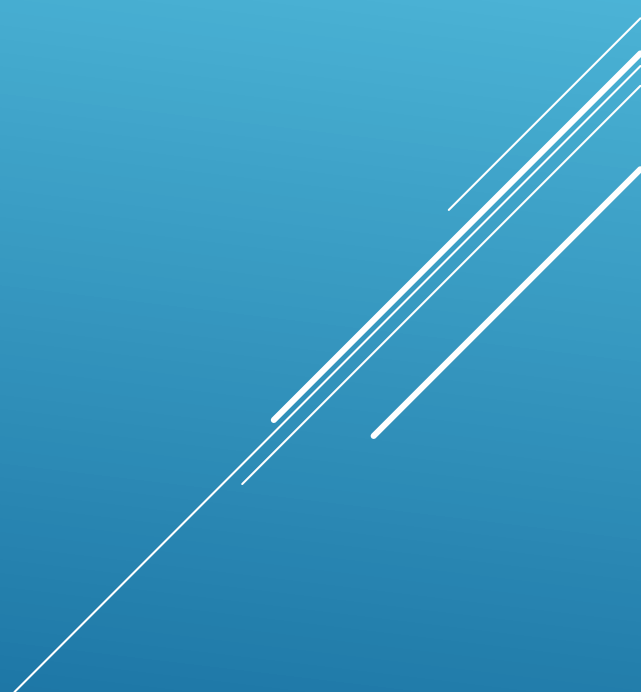
Il est possible de déclarer un ou une série de chiffres comme alphanumériques:

« 1234 » la valeur sera de type alphanumériques et non de type numérique

# Variables: les différent types

Le types booléen :

Type ne prenant comme valeur uniquement vrai (TRUE) ou faux (FALSE)



# Syntaxe d'affectation

L'affectation **variable**  $\leftarrow$  **valeur** est une instruction qui permet de changer la valeur d'une variable. L'affectation modifie le contenu du récipient désigné par la variable.

- Exemple : carré  $\leftarrow$  0 « se lit » le récipient carré reçoit la valeur 0.

Avertissement :

- L'affectation est une instruction qui est dite «destructrice».
- L'ancienne valeur de la variable est détruite, écrasée, effacée par la nouvelle valeur !
  - n  $\leftarrow$  12
  - carré  $\leftarrow$  n
  - $\rightarrow$  copie de la valeur de la variable n dans la variable carré qui n'est plus égale à 0.



# BONNE PRATIQUE

Les noms de variables doivent être écrits en respectant le camelCase, à savoir :

- Le nom d'une variable commence toujours par une minuscule et jamais par un nombre ex: variable ←3
- Pas de caractère accentués ou de ponctuation
- Toujours nommer ses variables de façon précise
- Ne pas commencer les noms de variable par des chiffres ou n'utiliser que des chiffres comme nom de variable
- Si le nom d'une variable est composé de plusieurs mots alors on fera commencer le nom de la variable par une minuscule et on « collera » chaque mot suivant en les faisant commencer par une Majuscule  
Ex: maVariable ←3

# BONNE PRATIQUE

Les noms de CONSTANCE doivent être écrits en respectant le **SNAKE\_CASE**, à savoir :

- Le nom d'une constante doit être écrit en Majuscule
- Pas de caractère accentués ou de ponctuation
- Toujours nommer ses Constantes de façon précise
- Ne pas commencer les nom de Constante par des chiffres ou n'utiliser que des chiffres comme nom de Constante
- Si le nom d'une constante est composé de plusieurs mots alors on séparera chaque mots par un **underscore**: **MA\_CONSTANTE**

# EXERCICES 1



# LECTURE ET ECRITURE

## En pseudo code

Pour afficher quelque chose, l'instruction en pseudo code est:

**Ecrire**

Exemple: la commande **Ecrire « HELLO WORLD »** affichera **HELLO WORLD** dans la console

Pour Demander à l'utilisateur d'encoder quelque chose, l'instruction en pseudo code est:

**Lire**

Exemple: la commande **Lire message** → l'utilisateur tapera quelque chose qui sera stocker dans la variable message

# Structures de contrôle

```
Si (condition) Alors  
    Instructions 1  
Sinon  
    Instruction 2  
FinSi
```

```
If(condition){  
    Instruction 1  
}else{  
    Instruction2  
}
```

**EXERCICES 2**

**EXERCICES 3**



# Structures itératives : Les BOUCLES

Ces structures permettent de répéter un ensemble d'instructions tant qu'une condition d'arrêt n'est pas atteinte!

ATTENTION DONC à la condition d'arrêt afin d'éviter de rentrer dans ce qu'on appelle on boucle infinie!

Il existe 2 principales Structure itératives:

- instruction TANT QUE
- instruction POUR

BONNE PRATIQUE: comme pour les structures conditionnelles, il est fortement recommandé d'indenter le code dans les structures itératives

# Structures itératives : TANT QUE

## Syntaxe :

```
TantQue (Condition booléen) FAIRE  
    Instructions  
FinTantQue
```

## Exemple :

```
Début  
    i  $\leftarrow$  10  
    TantQue(i > 0) FAIRE  
        Ecrire(i)  
        i  $\leftarrow$  i-1  
    FinTantQue  
Fin
```



# EXERCICES 4



# Les structures itératives : POUR

Syntaxe :

```
Pour i = 0 jusqu'à 10 FAIRE
    Instructions
FinPour
```

Exemple :

```
Début
    Pour i = 0 Jusqu'à 10 FAIRE
        Ecrire(i)
    FinPour
Fin
```

Des boucles peuvent être **imbriquées** ou **successives** .

Voilà un exemple typique de boucles imbriquées :

Une entreprise possède plusieurs employés et chaque employé traite plusieurs commandes on devra programmer une boucle principale (celle qui prend les employés un par un) et à l'intérieur, une boucle secondaire (celle qui prend les commandes de cet employé une par une).

Dans la pratique de la programmation, la maîtrise des boucles imbriquées est nécessaire, même si elle n'est pas suffisante.



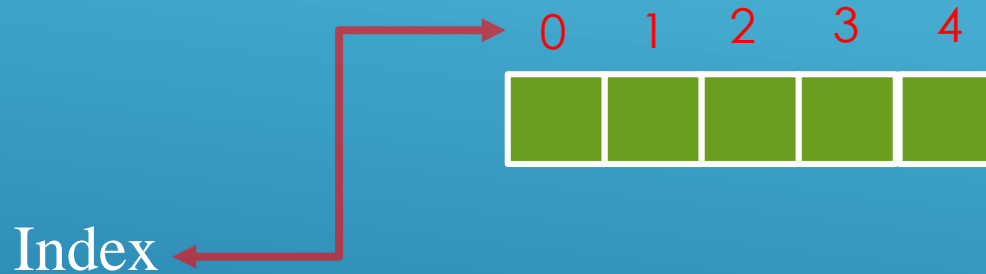
# EXERCICES 5



# Les Tableaux

Il existe un autre type de variable possible: **les tableaux**

Ils permettent d'enregistrer plusieurs données, valeurs dans une même variables et peuvent se schématiser comme suit pour un tableau de 5 cases:



Chaque case du tableau possède un index permettant l'identification de celle-ci.  
Dans la quasi-totalité des langage l'index de la 1ere case d'un tableau est 0  
(sauf Larp qui commence à 1)

Il existe 2 types de tableau :

- Les Tableaux numérotés
- Les tableaux associatifs

1°) Les Tableaux numérotés :

On peut déclarer les tableaux de 2 façons en PHP

```
$langages = ['C++','Java','JavaScript','PHP','C#'];
```

```
$langages = array('C++','Java','JavaScript','PHP','C#');
```

On peut afficher un tableau de plusieurs façons :

```
print_r($langages);
```

```
for($i=0;$i<count($langages);$i++){  
    echo $langages[$i] . "\n";  
}
```

```
foreach($langages as $valeur){  
    echo $valeur. "\n";  
}
```

Afficher un tableau avec index :

```
foreach($langages as $key => $valeur){  
    echo "Index " . $key . " : " . $valeur.  
    "\n";  
}
```

Pour récupérer la valeur d'une case précise d'un tableaux il est nécessaire d'indiquer l'index de cette case:

Exemple pour récupérer la valeur de la troisième case: `monTableau [2]` ici 2 correspond à l'indice 2 du tableau et comme le 1<sup>er</sup> indice est 0, 2 correspond bien à l'indice de la 3eme case

L'énorme avantage des tableaux, c'est qu'on va pouvoir les traiter en faisant des boucles.  
Par exemple, pour effectuer notre calcul de moyenne...





# EXERCICES 7



# TECHNIQUE A CONNAITRE

## LE TRI D'UN TABLEAU :

2 méthodes possibles:

- Le tri par sélection
- Le tri à bulles

Ces algorithmes sont souvent demandés en entretien technique auprès des entreprises

## Le principe du tri par sélection

Le principe du tri par **sélection/échange** (ou *tri par extraction*) est d'aller chercher le plus petit élément du tableau pour le mettre en premier, puis de repartir du second élément et d'aller chercher le plus petit élément du tableau pour le mettre en second, etc...

45	122	12	3	21	78	64	53	89	28	84	46
----	-----	----	---	----	----	----	----	----	----	----	----

3	122	12	45	21	78	64	53	89	28	84	46
---	-----	----	----	----	----	----	----	----	----	----	----

On recommence à chercher le plus petit élément, mais cette fois, **seulement à partir du deuxième** (puisque le premier est maintenant correct, on n'y touche plus). On le trouve en troisième position (c'est le nombre 12). On échange donc le deuxième avec le troisième :

3	12	122	45	21	78	64	53	89	28	84	46
---	----	-----	----	----	----	----	----	----	----	----	----

On recommence à chercher le plus petit élément à partir du troisième (puisque les deux premiers sont maintenant bien placés), et on le place correctement, en l'échangeant, ce qui donnera in fine :

Et ainsi de suite.... Jusqu'à l'avant dernier

# LE TRI PAR SELECTION:

```
for($i=0;$i<count($tab);$i++){  
    for($j=$i+1;$j<count($tab);$j++){  
        if($tab[$j]<$tab[$i]){  
            $temp = $tab[$i];  
            $tab[$i] = $tab[$j];  
            $tab[$j] = $temp;  
        }  
    }  
}
```

<http://lwh.free.fr/pages/algo/tri/tri.htm>

On peut décrire le processus de la manière suivante :

**Boucle principale** : prenons comme point de départ le premier élément, puis le second, etc., jusqu'à l'avant dernier.

**Boucle secondaire** : à partir de ce point de départ mouvant, recherchons jusqu'à la fin du tableau quel est le plus petit élément. Une fois que nous l'avons trouvé, nous l'échangeons avec le point de départ.

# TECHNIQUE A CONNAITRE

## LE TRI A BULLE:

L'idée de départ du tri à bulles consiste à se dire qu'un tableau trié en ordre croissant, c'est un tableau dans lequel **tout élément est plus petit que celui qui le suit**.

En effet, prenons chaque élément d'un tableau, et comparons-le avec l'élément qui le suit. Si l'ordre n'est pas bon, on permute ces deux éléments. Et on recommence jusqu'à ce que l'on n'ait plus aucune permutation à effectuer. Les éléments les plus grands « remontent » ainsi peu à peu vers les dernières places

### Notion a prendre en compte pour ce tri:

Le flag, en anglais, est un petit drapeau, qui va rester baissé aussi longtemps que l'événement attendu ne se produit pas. Et, aussitôt que cet événement a lieu, le petit drapeau se lève (la variable booléenne change de valeur). Ainsi, la valeur finale de la variable booléenne permet au programmeur de savoir si l'événement a eu lieu ou non

## LE TRI A BULLE:

```
while(!$trie){  
    $trie = true;  
    for($i=0;$i<count($tab)-1;$i++){  
        if($tab[$i] > $tab[$i+1]){  
            $temp = $tab[$i];  
            $tab[$i] = $tab[$i + 1];  
            $tab[$i+1] = $temp;  
            $trie = false;  
        }  
    }  
}
```

**Mais il ne faut pas oublier un détail capital** : la gestion de notre flag. L'idée, c'est que cette variable va nous signaler le fait qu'il y a eu au moins une permutation effectuée. Il faut donc :

lui attribuer la valeur Vrai dès qu'une seule permutation a été faite (il suffit qu'il y en ait eu une seule pour qu'on doive tout recommencer encore une fois).

la **remettre à Faux** à chaque tour de la boucle principale (quand on recommence un nouveau tour général de bulles, il n'y a pas encore eu d'éléments échangés),

dernier point, il ne faut pas oublier de lancer la boucle principale, et pour cela de donner **la valeur Vrai au flag** au tout départ de l'algorithme.

# TECHNIQUE A CONNAITRE

## La recherche dichotomique

Pour une machine, quelle est la manière la plus rationnelle de chercher dans un dictionnaire ? C'est de comparer le mot à vérifier avec le mot qui se trouve pile poil au milieu du dictionnaire. Si le mot à vérifier est antérieur dans l'ordre alphabétique, on sait qu'on devra le chercher dorénavant dans la première moitié du dico. Sinon, on sait maintenant qu'on devra le chercher dans la deuxième moitié.

[https://www.infoforall.fr/art/algo/animation-de-la-recherche-dichotomique/#partie\\_1](https://www.infoforall.fr/art/algo/animation-de-la-recherche-dichotomique/#partie_1)

Several white diagonal lines of varying lengths and thicknesses are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

# EXERCICES 8

