ALGORITHMIQUE

1. Qu’est-ce qu’un algorithme ?

Objectifs :

- Comprendre ce qu’est un algorithme.
- Identifier des algorithmes dans la vie quotidienne.
- Apprendre à décomposer un problème en étapes simples.

Définition :

Un **algorithme** est une suite d'instructions claires et précises permettant de résoudre un problème ou d’accomplir une tâche.

Exemple du quotidien : Une recette de cuisine

Prenons l’exemple de la préparation d’un sandwich.

Les étapes peuvent être décrites comme suit :

1. Prendre deux tranches de pain.
2. Mettre une tranche de jambon sur une tranche de pain.
3. Ajouter une tranche de fromage.
4. Mettre la deuxième tranche de pain par-dessus.

Résultat attendu : Un sandwich prêt à être mangé.

Exemple en informatique : Trouver le plus grand entre deux nombres

1. Lire le premier nombre (appelons-le `nombre1`).
2. Lire le second nombre (`nombre2`).
3. Si `nombre1` est plus grand que `nombre2`, afficher `nombre1`.
4. Sinon, afficher `nombre2`.

Les caractéristiques d’un bon algorithme :

1. **Précis :** Chaque étape doit être claire et sans ambiguïté.
2. **Fini :** L'algorithme doit avoir une fin.
3. **Efficace :** Les étapes doivent être organisées pour résoudre le problème rapidement.

Importance des algorithmes en informatique :

Les algorithmes sont utilisés pour :

- Trier des données (par exemple, trier une liste de noms par ordre alphabétique).
- Rechercher une information (comme trouver un contact dans un téléphone).
- Résoudre des problèmes complexes (calculs, jeux, simulations).

Travaux pratiques (TP)

TP 1 : Écrire un algorithme pour se brosser les dents

1. Demandez aux participants d'écrire (sur papier) les étapes pour se brosser les dents.

   Exemple attendu :

   1. Prendre une brosse à dents.
   2. Mettre du dentifrice sur la brosse.
   3. Ouvrir la bouche.
   4. Brosser les dents pendant 2 minutes.
   5. Rincer la bouche avec de l'eau.
2. Discussion : Quels sont les points importants (exemple : bien rincer la bouche) ?

TP 2 : Écrire un algorithme pour préparer un verre de jus

Enoncé : Décrivez les étapes pour préparer un verre de jus avec du sirop et de l'eau.

Exemple attendu :

1. Prendre un verre.
2. Mettre 2 cuillères de sirop dans le verre.
3. Ajouter de l’eau.
4. Remuer avec une cuillère.
5. Servir le verre.

TP 3 : Identifier un algorithme dans la vie courante

Demandez aux participants de réfléchir à une situation du quotidien où ils suivent des étapes précises, comme :

- Faire du café.
- Aller à l’école ou au travail.
- S’habiller le matin.

Exercice interactif : Compléter un algorithme manquant

Proposez un algorithme avec des étapes incomplètes et demandez aux participants de le terminer.

Exemple : Préparer une omelette (incomplet)

1. Prendre une poêle.
2. _____ (réponse attendue : Mettre la poêle sur le feu).
3. _____ (réponse attendue : Casser les œufs dans un bol).
4. Mélanger les œufs.
5. _____ (réponse attendue : Verser les œufs dans la poêle).

---

2. Les bases des algorithmes

Objectifs :

- Comprendre ce qu’est une variable et comment l’utiliser.
- Utiliser des instructions de base : calculs, entrées, sorties.
- Introduire les conditions pour prendre des décisions.

2.1 Variables et calculs simples

Définition :

Une **variable** est comme une boîte dans laquelle on stocke une information.

- Elle a un **nom** : pour savoir ce qu'elle contient.
- Elle a une **valeur** : ce qu'elle stocke (nombre, texte, etc.).

Exemple :

- Variable `age` = 18
- Variable `nom` = "Ali"

Exemple concret :

1. On demande à une personne son âge.
2. On stocke cet âge dans une variable appelée `age`.
3. On affiche la valeur de `age`.

Calculs simples :

On peut utiliser des variables pour effectuer des calculs.

Exemple : Calculer la somme de deux nombres

1. Lire le premier nombre (`nombre1`).
2. Lire le second nombre (`nombre2`).
3. Ajouter les deux nombres et stocker le résultat dans `somme`.
4. Afficher `somme`.

Pseudocode :

```
Entrer nombre1
Entrer nombre2
somme ← nombre1 + nombre2
Afficher somme
```

Travaux pratiques (TP)

TP 1 : Calculer la somme de deux nombres

1. Demander à l'utilisateur de donner deux nombres.
2. Ajouter les deux nombres.
3. Afficher la somme.

Exemple attendu :

Si les nombres donnés sont `10` et `5`, le programme affiche : **15**.

TP 2 : Calculer le périmètre d'un rectangle

1. Lire la longueur (`longueur`).
2. Lire la largeur (`largeur`).
3. Calculer le périmètre avec la formule : `périmètre = 2 × (longueur + largeur)`.
4. Afficher le périmètre.

Exemple attendu :

Si la longueur est `5` et la largeur est `3`, le programme affiche : **16**.

2.2 Entrées et sorties

Entrées :

Ce sont les informations que l'algorithme demande à l'utilisateur.

Exemple : Demander un nombre.

Sorties :

Ce sont les résultats affichés par l'algorithme.

Exemple : Afficher un message ou un résultat.

Travaux pratiques (TP)

TP 3 : Calculer une note moyenne

1. Demander trois notes (par exemple, `note1`, `note2`, `note3`).
2. Calculer la moyenne avec la formule : `moyenne = (note1 + note2 + note3) / 3`.
3. Afficher la moyenne.

Exemple attendu :

Si les notes sont `10`, `15` et `20`, la moyenne affichée est : **15**.

2.3 Les décisions (conditions)

Définition :

Les conditions permettent à un algorithme de choisir entre plusieurs actions.

Exemple : Si on a faim, on mange. Sinon, on ne mange pas.

Structure d'une condition :

```
Si (condition) Alors
    Faire une action
Sinon
    Faire une autre action
```

Exemple : Vérifier si un nombre est pair ou impair

1. Lire un nombre (`nombre`).
2. Si `nombre % 2 = 0`, alors afficher "Nombre pair".
3. Sinon, afficher "Nombre impair".

Pseudocode :

```
Entrer nombre
Si nombre % 2 = 0 Alors
    Afficher "Nombre pair"
Sinon
    Afficher "Nombre impair"
```

Travaux pratiques (TP)

TP 4 : Vérifier si un nombre est positif ou négatif

1. Lire un nombre.
2. Si le nombre est supérieur ou égal à 0, afficher : "Nombre positif".

3. Sinon, afficher : "Nombre négatif".

Exemple attendu :

Si le nombre est `-5`, afficher : **"Nombre négatif"**.

TP 5 : Trouver le plus grand entre deux nombres

1. Lire deux nombres (`nombre1`, `nombre2`).
2. Comparer les deux :
   - Si `nombre1` > `nombre2`, afficher : "Nombre1 est le plus grand".
   - Sinon, afficher : "Nombre2 est le plus grand".

Exemple attendu :

Si `nombre1 = 8` et `nombre2 = 12`, afficher : **"Nombre2 est le plus grand"**.

---

3. Répéter des actions (les boucles)

Objectifs :

- Comprendre l'utilité des boucles pour éviter les répétitions inutiles.
- Découvrir les deux types de boucles : **"Tant que"** et **"Pour"**.
- Appliquer les boucles à des problèmes simples.

3.1 Qu'est-ce qu'une boucle ?

Une boucle permet de **répéter une action plusieurs fois** sans avoir à écrire les instructions encore et encore.

Exemple du quotidien :

Compter de 1 à 5.

Sans boucle :

1. Afficher "1".
2. Afficher "2".
3. Afficher "3".
4. Afficher "4".

5. Afficher "5".

Avec boucle :

1. Commencer à "1".
2. Tant que le nombre est inférieur ou égal à "5", afficher le nombre.
3. Ajouter "1" au nombre et recommencer.

3.2 Types de boucles

a. La boucle "Tant que" (While loop)

- Elle répète une action **tant qu'une condition est vraie**.
- Quand la condition devient fausse, la boucle s'arrête.

Structure :

```
Tant que (condition) Faire
    Action
Fin Tant que
```

Exemple : Compter de 1 à 5

1. Définir une variable `nombre` = 1.
2. Tant que `nombre ≤ 5` :
    - Afficher `nombre`.
    - Ajouter 1 à `nombre`.

Pseudocode :

```
nombre ← 1
Tant que nombre ≤ 5 Faire
    Afficher nombre
    nombre ← nombre + 1
Fin Tant que
```

---

b. La boucle "Pour" (For loop)

- On connaît à l'avance **le nombre de répétitions**.

- Elle est souvent utilisée pour parcourir une liste ou une plage de nombres.

Structure :

```
Pour (variable allant de début à fin) Faire
    Action
Fin Pour
```

Exemple : Compter de 1 à 5

1. Pour chaque `nombre` allant de 1 à 5 :
    - Afficher `nombre`.

Pseudocode :

```
Pour nombre allant de 1 à 5 Faire
    Afficher nombre
Fin Pour
```

3.3 Travaux pratiques (TP)

TP 1 : Afficher les nombres de 1 à 10 avec une boucle "Tant que"

1. Créer une variable `nombre` = 1.
2. Tant que `nombre ≤ 10`, afficher `nombre`.
3. Ajouter 1 à `nombre` à chaque itération.

Exemple attendu :

```
1
2
3
...
10
```

TP 2 : Afficher les nombres pairs entre 1 et 10 avec une boucle "Pour"

1. Pour chaque `nombre` allant de 1 à 10 :
    - Si `nombre % 2 = 0`, afficher `nombre`.

Exemple attendu :

```
2
4
6
8
10
```

3.4 Applications concrètes des boucles

a. Calculer la somme des nombres de 1 à N

1. Lire un nombre `N`.
2. Initialiser une variable `somme` = 0.
3. Pour chaque nombre de 1 à `N` :
    - Ajouter le nombre à `somme`.
4. Afficher `somme`.

Exemple attendu :

Si `N = 5`, la somme affichée est **15** (1 + 2 + 3 + 4 + 5).

---

b. Calculer la table de multiplication d'un nombre

1. Lire un nombre `N`.
2. Pour chaque nombre `i` de 1 à 10 :
    - Calculer `produit = N × i`.
    - Afficher `N × i = produit`.

Exemple attendu :

Si `N = 3`, la table affichée est :

```
3 × 1 = 3
3 × 2 = 6
3 × 3 = 9
```

```
...
3 × 10 = 30
```

Travaux pratiques avancés

TP 3 : Trouver la factorielle d'un nombre

1. Lire un nombre `N`.
2. Initialiser une variable `factorielle` = 1.
3. Pour chaque nombre de 1 à `N` :
    - Multiplier `factorielle` par le nombre.
4. Afficher `factorielle`.

Exemple attendu :

Si `N = 4`, la factorielle affichée est **24** (1 × 2 × 3 × 4).

TP 4 : Deviner un nombre (jeu simple)

1. Choisir un nombre secret (par exemple, `7`).
2. Tant que l'utilisateur ne devine pas le nombre :
    - Demander un nombre.
    - Si c'est correct, afficher : "Bravo, vous avez trouvé !"
    - Sinon, afficher : "Réessayez !".

Exemple attendu :

Utilisateur entre `5` → "Réessayez !"

Utilisateur entre `7` → "Bravo, vous avez trouvé !"

4. Les structures de données simples

Objectifs :

- Comprendre comment organiser et manipuler plusieurs valeurs avec des structures simples.
- Découvrir les **tableaux** (ou listes) pour gérer des collections de données.

- Résoudre des problèmes simples avec des tableaux.

4.1 Qu'est-ce qu'un tableau ?

Un **tableau** (ou liste) est une structure qui permet de stocker plusieurs valeurs dans une seule variable.

Chaque valeur dans un tableau a une **position** appelée **indice**.

Exemple : Tableau de nombres

Un tableau peut contenir les notes d'un élève :

`notes = [12, 15, 14, 18]`

- `notes[0]` = 12 (première note)
- `notes[1]` = 15 (deuxième note)
- `notes[2]` = 14 (troisième note)
- `notes[3]` = 18 (quatrième note)

4.2 Manipuler un tableau

a. Déclarer un tableau

Un tableau est créé en définissant ses valeurs.

Exemple :

```
tableau = [2, 4, 6, 8]
```

b. Accéder à un élément

On utilise l'indice pour accéder à un élément.

Exemple :

Si `tableau = [2, 4, 6, 8]`, alors :

- `tableau[0]` renvoie **2**.
- `tableau[3]` renvoie **8**.

c. Parcourir un tableau avec une boucle

On utilise une boucle pour parcourir les éléments.

Exemple : Afficher tous les éléments

```
Pour i allant de 0 à longueur(tableau) - 1 Faire
    Afficher tableau[i]
```

4.3 Travaux pratiques (TP)

TP 1 : Afficher les éléments d'un tableau

1. Créer un tableau contenant 5 nombres : `[3, 7, 1, 9, 5]`.
2. Utiliser une boucle pour afficher chaque nombre.

Exemple attendu :

```
3
7
1
9
5
```

TP 2 : Trouver la somme des éléments d'un tableau

1. Créer un tableau contenant les nombres `[3, 7, 1, 9, 5]`.
2. Initialiser une variable `somme` à 0.
3. Parcourir le tableau avec une boucle et ajouter chaque élément à `somme`.
4. Afficher la somme.

Exemple attendu :

Si le tableau est `[3, 7, 1, 9, 5]`, la somme affichée est **25**.

TP 3 : Trouver le plus grand élément d'un tableau

1. Créer un tableau contenant les nombres `[3, 7, 1, 9, 5]`.
2. Initialiser une variable `max` avec la première valeur du tableau.
3. Parcourir le tableau et mettre à jour `max` si un élément est plus grand.
4. Afficher `max`.

Exemple attendu :

Si le tableau est `[3, 7, 1, 9, 5]`, le plus grand nombre est **9**.

4.4 Applications concrètes des tableaux

a. Calculer la moyenne des notes d'un élève

1. Créer un tableau contenant les notes `[12, 15, 14, 18]`.
2. Calculer la somme des notes.
3. Diviser la somme par le nombre de notes pour obtenir la moyenne.
4. Afficher la moyenne.

Exemple attendu :

Si les notes sont `[12, 15, 14, 18]`, la moyenne affichée est **14.75**.

---

b. Rechercher un élément dans un tableau

1. Créer un tableau contenant les noms `[Ali, Sara, Karim, Fatou]`.
2. Demander à l'utilisateur de saisir un nom.
3. Parcourir le tableau pour vérifier si le nom existe.
4. Afficher un message :
    - Si le nom est trouvé : "Nom trouvé".
    - Sinon : "Nom non trouvé".

Exemple attendu :

Si le tableau contient `[Ali, Sara, Karim, Fatou]` et l'utilisateur entre `Sara`, afficher **"Nom trouvé"**.

Travaux pratiques avancés

TP 4 : Trier un tableau (ordre croissant)

1. Créer un tableau contenant les nombres `[7, 3, 9, 1, 5]`.
2. Utiliser une méthode simple pour trier les nombres dans l'ordre croissant.
    - Comparer deux nombres adjacents et échanger leurs positions si nécessaire.

3. Afficher le tableau trié.

Exemple attendu :

Si le tableau est `[7, 3, 9, 1, 5]`, le tableau trié est **[1, 3, 5, 7, 9]**.

TP 5 : Inverser un tableau

1. Créer un tableau contenant `[3, 7, 1, 9, 5]`.
2. Utiliser une boucle pour inverser les éléments du tableau.
3. Afficher le tableau inversé.

Exemple attendu :

Si le tableau est `[3, 7, 1, 9, 5]`, le tableau inversé est **[5, 9, 1, 7, 3]**.

---

5. Résolution de problèmes : mettre tout ensemble

Objectifs :

- Appliquer les concepts étudiés pour résoudre des problèmes réels.
- Structurer les solutions en utilisant les bases de l'algorithmique.
- Gagner en confiance pour identifier et résoudre des problèmes étape par étape.

5.1 Les étapes de résolution de problèmes

1. **Comprendre le problème**
   - Lire et analyser le problème.
   - Identifier les données d'entrée et de sortie.
2. **Décomposer le problème en étapes simples**
   - Écrire les étapes nécessaires pour arriver à la solution.
3. **Écrire l'algorithme**
   - Rédiger le pseudocode pour les étapes définies.
4. **Tester la solution**
   - Utiliser des cas concrets pour vérifier si l'algorithme fonctionne.

5.2 Exemples et Travaux Pratiques (TP)

TP 1 : Vérifier si un nombre est pair ou impair

Problème :

Écrire un algorithme qui demande à l'utilisateur de saisir un nombre et affiche si ce nombre est pair ou impair.

Étapes :

1. Lire un nombre `n`.
2. Si `n % 2 = 0`, afficher "Le nombre est pair".
3. Sinon, afficher "Le nombre est impair".

Pseudocode :

```
Lire n
Si n % 2 = 0 Alors
    Afficher "Le nombre est pair"
Sinon
    Afficher "Le nombre est impair"
Fin Si
```

Cas de test :

- Entrée : `4` → Résultat : "Le nombre est pair".
- Entrée : `7` → Résultat : "Le nombre est impair".

TP 2 : Trouver le maximum de trois nombres

Problème :

Écrire un algorithme qui demande à l'utilisateur de saisir trois nombres et affiche le plus grand.

Étapes :

1. Lire trois nombres `a`, `b`, et `c`.
2. Si `a` est plus grand que `b` et `c`, afficher `a`.
3. Sinon, si `b` est plus grand que `a` et `c`, afficher `b`.
4. Sinon, afficher `c`.

Pseudocode :

```
Lire a, b, c
Si a > b et a > c Alors
    Afficher a
Sinon Si b > a et b > c Alors
    Afficher b
Sinon
    Afficher c
Fin Si
```

Cas de test :

- Entrée : `5, 12, 9` → Résultat : `12`.
- Entrée : `10, 3, 7` → Résultat : `10`.

TP 3 : Calculer la moyenne des nombres saisis par l'utilisateur

Problème :

Écrire un algorithme qui demande à l'utilisateur de saisir 5 nombres, calcule leur moyenne et l'affiche.

Étapes :

1. Initialiser une variable `somme` à 0.
2. Répéter 5 fois :
    - Lire un nombre.
    - Ajouter ce nombre à `somme`.
3. Diviser `somme` par 5 et afficher le résultat.

Pseudocode :

```
somme ← 0
Pour i allant de 1 à 5 Faire
    Lire nombre
    somme ← somme + nombre
Fin Pour
```

```
moyenne ← somme / 5
Afficher moyenne
```

Cas de test :

- Entrée : `10, 15, 20, 25, 30` → Résultat : `20`.
- Entrée : `5, 5, 5, 5, 5` → Résultat : `5`.

5.3 Résolution de problèmes avancés

TP 4 : Inverser les caractères d'un mot

Problème :

Écrire un algorithme qui demande à l'utilisateur de saisir un mot et affiche ce mot à l'envers.

Étapes :

1. Lire un mot `mot`.
2. Initialiser un mot vide `motInverse`.
3. Parcourir les caractères du mot à l'envers et les ajouter à `motInverse`.
4. Afficher `motInverse`.

Pseudocode :

```
Lire mot
motInverse ← ""
Pour i allant de longueur(mot) - 1 à 0 Faire
    motInverse ← motInverse + mot[i]
Fin Pour
Afficher motInverse
```

Cas de test :

- Entrée : `chat` → Résultat : `tahc`.
- Entrée : `algo` → Résultat : `ogla`.

TP 5 : Trouver tous les multiples d'un nombre jusqu'à N

Problème :

Écrire un algorithme qui demande à l'utilisateur de saisir un nombre `x` et une limite `N`, puis affiche tous les multiples de `x` jusqu'à `N`.

Étapes :

1. Lire `x` et `N`.
2. Initialiser une variable `i` à 1.
3. Tant que `i × x ≤ N` :
   - Afficher `i × x`.
   - Incrémenter `i`.

Pseudocode :

```
Lire x, N
i ← 1
Tant que i × x ≤ N Faire
    Afficher i × x
    i ← i + 1
Fin Tant que
```

Cas de test :

- Entrée : `x = 3, N = 15` → Résultat : `3, 6, 9, 12, 15`.
- Entrée : `x = 5, N = 20` → Résultat : `5, 10, 15, 20`.

5.4 Résolution de mini-projet

Mini-Projet : Jeu "Deviner le nombre"

Problème :

Créer un jeu où l'ordinateur choisit un nombre entre 1 et 100, et l'utilisateur doit deviner ce nombre. À chaque essai, l'ordinateur indique si le nombre est trop grand ou trop petit.

Étapes :

1. Choisir un nombre aléatoire `secret` entre 1 et 100.
2. Répéter jusqu'à ce que l'utilisateur trouve :
   - Lire une proposition `guess`.

- Si `guess` = `secret`, afficher "Bravo !" et arrêter.
- Si `guess` < `secret`, afficher "Trop petit".
- Sinon, afficher "Trop grand".

Pseudocode :

```
secret ← NombreAléatoire(1, 100)
Tant que vrai Faire
    Lire guess
    Si guess = secret Alors
        Afficher "Bravo !"
        Quitter
    Sinon Si guess < secret Alors
        Afficher "Trop petit"
    Sinon
        Afficher "Trop grand"
    Fin Si
Fin Tant que
```

Cas de test :

- Secret : `45`, utilisateur devine `30, 50, 45` → Résultat : "Trop petit", "Trop grand", "Bravo !".
