# FICHE DE RÉVISION : ALGORITHMIE

## INTRODUCTION À L'ALGORITHMIE

- Un algorithme est une suite d'instructions données à un ordinateur

- Plusieurs langages existent, mais les principes fondamentaux restent les mêmes

- Un langage procédural exécute les instructions séquentiellement, de haut en bas

- Pour un même problème, plusieurs solutions algorithmiques sont possibles

- L'optimisation est importante mais secondaire par rapport à la résolution du problème

## PRINCIPES FONDAMENTAUX

- La phase de réflexion est universelle, quel que soit le langage de programmation

- La traduction en code dépend du langage choisi (JavaScript, PHP, etc.)

- Analogie avec les langues : réfléchir dans sa langue maternelle puis traduire

## STRUCTURES DE BASE

- \*\*Instructions séquentielles\*\* : s'enchaînent les unes après les autres

- \*\*Structures répétitives\*\* : permettent de répéter des instructions (boucles)

- \*\*Structures conditionnelles\*\* : permettent d'exécuter des instructions selon des conditions

## VARIABLES

### Déclaration de variable

- Une variable n'a pas de valeur tant qu'elle n'est pas définie

- Chaque variable doit être définie par :

\* Son nom

\* Son type

\* Sa valeur (optionnelle à la déclaration)

- Syntaxe : `Var entier NomdeLaVariable = 5;`

- Utiliser `let` si la valeur change souvent, `const` si elle ne change pas

### Types de variables

- \*\*Entier\*\* : Nombre entier positif ou négatif

\* Exemple : `var entier chiffre = 1845;`

- \*\*Réel\*\* : Variable numérique avec décimale

\* Exemple : `var réel Chiffres = 5.555;`

- \*\*Chaîne de caractères\*\* : Texte entre guillemets

\* Exemple : `var chainedecaracteres prenom = "Gwenhael";`

- \*\*Booléen\*\* : Valeur True ou False

\* Exemple : `var bool estMajeur = True;`

### Opérations

- \*\*Arithmétiques\*\* : +, -, \*, /, MOD (reste d'une division)

- \*\*Comparaison\*\* : >, <, >=, <=, =, != (différent)

- \*\*Logiques\*\* : && (ET), || (OU), ! (NON)

### Affectation de variable

```pseudocode

Var NomDeLaVariable;

NomDeLaVariable = 3;

NomDeLaVariable = NomDeLaVariable \* 3;

Afficher(NomDeLaVariable) // Affiche 9

```

## FONCTIONS

### Structure d'une fonction

```pseudocode

Algorithme Nom\_De\_la\_Fonction {

Déclarations :

// déclaration des variables, éventuellement de fonctions

Début {

// Instructions

} Fin

}

```

### Exemple de fonction

```pseudocode

Algorithme afficher\_variable

Var entier entierASaisir;

Debut {

Saisir(entierASaisir);

Afficher(entierASaisir);

} Fin

```

## EXEMPLES PRATIQUES

### Exemple 1 : Recette de gâteau

- Plusieurs façons d'arriver au même résultat

- L'ordre des ingrédients peut varier

- L'optimisation peut influencer l'ordre des étapes

### Exemple 2 : Montage de meuble

- Les pieds peuvent être montés dans n'importe quel ordre

- Le résultat final reste le même

- L'important est d'arriver à une table stable

## CONSEILS POUR LA RÉFLEXION

1. Commencer par réfléchir au problème en français

2. Décomposer le problème en étapes logiques

3. Traduire ensuite en pseudo-code

4. Adapter le pseudo-code au langage de programmation choisi

Bien sûr ! Améliorer l'apparence de votre prise de notes peut faire toute la différence dans votre compréhension et votre mémorisation des concepts d'algorithmie. Tout d'abord, il serait bénéfique d'organiser vos informations de manière claire et structurée. Par exemple, vous pourriez utiliser des titres et des sous-titres pour chaque section, cela aidera à naviguer plus facilement dans vos notes. L'ajout de listes à puces ou numérotées renforce la lisibilité, tout en permettant de mettre en avant les points clés, comme les types de variables et les opérations.

En outre, l'utilisation de couleurs et de typographies différentes pour mettre en évidence les concepts importants peut rendre vos notes encore plus attrayantes. Par exemple, vous pourriez utiliser le gras pour les termes techniques et des couleurs distinctes pour chaque catégorie d'information (comme des variables, des fonctions ou des exemples pratiques). Cela non seulement ajoutera une touche esthétique, mais facilitera également la révision en permettant de distinguer rapidement les informations.

Enfin, n'oubliez pas que l'optimisation de votre prise de notes doit aussi inclure des éléments pratiques, tels que des exemples et des analogies claires, qui donnent vie à des notions parfois abstraites de l'algorithmie. En utilisant des métaphores accessibles, comme celles de la recette de gâteau ou du montage de meuble, vous renderez vos notes non seulement plus engageantes, mais également plus mémorables. En somme, en mettant un peu de créativité dans vos méthodes de prise de notes, vous vous préparez à devenir un expert en algorithmie avec enthousiasme et clarté !