

Introduction aux algorithmes >> Les Bases

Chapitre 1: Variables & opérateurs

Les algorithmes

Les Algorithmes

INTRODUCTION :



Les algorithmes

- Un **algorithme** est une suite finie et ordonnée d'instructions permettant de résoudre **un problème** ou d'effectuer une tâche.
- Les algorithmes sont au cœur de la programmation et sont utilisés pour **automatiser des processus, manipuler des données, ou prendre des décisions.**

Structure de base d'un algorithme

- Un algorithme peut être décomposé en plusieurs parties :

1.Initialisation : Définir les variables et les données nécessaires.

2.Traitement : Appliquer les instructions pour manipuler les données.

3.Sortie : Afficher ou retourner le résultat.

Exemple simple : Addition de deux nombres

Algorithme Addition

Variables a, b, somme : Entier

Début

a ← 5;

b ← 10;

somme ← a + b;

Afficher (somme);

Fin

Les Variables

Une **variable** est un **espace mémoire** nommé qui permet de **stocker** une **valeur**.

Chaque variable a **un nom** et **un type**.

Exemple :

➤ Algorithme ExempleVariable

Variable **age** : Entier

Début

age ← 25 ;

Afficher ("L'âge est : ", **age**);

Fin

Les Types de Données:

Les types de données définissent la nature des valeurs que peut prendre une variable. Voici quelques types courants :

- **Entier** : Pour les nombres entiers (ex. 5, -10).
- **Réel** : Pour les nombres à virgule flottante (ex. 3.14, -0.5).
- **Chaîne de caractères** : Pour le texte (ex. "Bonjour").
- **Booléen** : Pour les valeurs logiques (Vrai ou Faux).

Les Opérateurs:

Les opérateurs permettent de manipuler les valeurs des variables. Voici les principaux types d'opérateurs :

➤ Opérateurs Arithmétiques :

+ : Addition

- : Soustraction

* : Multiplication

/ : Division

% : Modulo (reste de la division)

Exemple d'utilisation :

Algorithme ExempleOperateursArithmetiques

Variables a, b, somme, produit : **Entier**

Début

a ← 10 ;

b ← 3 ;

somme ← a + b ;

produit ← a * b ;

Afficher ("Somme : ", somme);

Afficher ("Produit : ", produit);

Fin

Opérateurs de Comparaison

== : Égal à

!= : Différent de

> : Supérieur à

< : Inférieur à

>= : Supérieur ou égal à

<= : Inférieur ou égal à

Exemple :

Algorithme Opert_compar

Variables **a**, **b** : Entier ;

Début

a \leftarrow 5;

b \leftarrow 10;

Si **a** < **b** **Alors**

Afficher ("a est inférieur à b«);

Fin Si

Fin



Opérateurs Logiques :

ET : Vrai si les deux conditions sont vraies.

OU : Vrai si au moins une condition est vraie.

NON : Inverse la valeur logique.

Opérateurs Logiques :

Algorithme ExempleOperateursLogiques

Variables a, b : Entier

Début

a ← 5;

b ← 10;

Si ((a > 0) ET (b > 0)) Alors

Afficher ("a et b sont positifs«);

Fin Si

Fin

Lire et ecrire introduction à l'utilisation

Les algorithmes

Lire() & ecrire()



Lire et Écrire :

Lire : Permet de saisir une valeur depuis l'utilisateur.

Écrire (Afficher) : Permet d'afficher une valeur à l'écran.

Exemple :

```
Algorithme ExempleLireEcrire  
  Variable nom : Chaîne ;  
  Début  
    Afficher ("Entrez votre nom :");  
    Lire (nom);  
    Afficher ("Bonjour, ", nom);  
  Fin
```

Introduction aux algorithmes :

Chapitre 2: Les structures de controle

Les Basiques de l'algorithmique

Les Structures de Contrôle:

- Les structures de contrôle sont des éléments fondamentaux en programmation qui permettent **de gérer le flux d'exécution** d'un programme.
- Elles déterminent **l'ordre** dans lequel les instructions sont **exécutées** en fonction de certaines **conditions** ou de **répétitions**.

Structures conditionnelles:

Elles permettent d'exécuter des instructions en fonction de la vérification d'une condition.

Si (If):

Exécute un bloc d'instructions si une condition est vraie.

Syntaxe en algorithme :

Si (condition) **Alors**
 Instructions;
FinSi

Exemple :

Si (age >= 18) **Alors**
 Afficher("Vous êtes majeur.");
FinSi

Structure de Contrôle : Selon ...

En algorithmique, un **switch** (ou **selon** en français) est **une structure de contrôle** qui permet de simplifier les conditions multiples.

Il est utilisé pour **exécuter** différentes parties de code en fonction de la valeur d'une **variable** ou d'une expression.

Syntaxe générale

```
switch (variable) {
```

```
  cas valeur1:
```

```
    // instructions à exécuter si variable == valeur1
```

```
  fin_cas
```

```
  cas valeur2:
```

```
    // instructions à exécuter si variable == valeur2
```

```
  fin_cas
```

```
  défaut:
```

```
    // instructions à exécuter si aucune valeur ne correspond
```

```
  fin_défaut
```

```
}
```

Les Boucles :

En algorithmique, il existe trois types de boucles principaux qui permettent de répéter un ensemble d'instructions plusieurs fois.

Boucle while (boucle tant que)

La boucle **while** est utilisée quand le nombre d'itérations **dépend d'une condition**.

Elle répète **les instructions tant que** la condition est **vraie**.



Structure en pseudo-code :

```
i ← 0 ;  
tant que (i < 5) faire  
    afficher(i) ;  
    i ← i + 1 ;  
fin tant que
```

Explication :

La boucle continue tant que $i < 5$.

À chaque itération, i est affiché, puis incrémenté de 1.

La boucle s'arrête quand i atteint 5.



Boucle do-while (boucle répéter jusqu'à)

La boucle **do-while** est similaire à la boucle **while**, mais elle garantit que le bloc d'instructions est exécuté **au moins une fois**, car la condition **est vérifiée après l'exécution du bloc**.



Exemple en pseudo-code :

```
i ← 0  
répéter  
    afficher(i) ;  
    i ← i + 1;  
jusqu'à (i >= 5)
```

Explication:

Le bloc d'instructions est exécuté au moins une fois.

La condition $i \geq 5$ est vérifiée après chaque itération.

La boucle s'arrête quand i atteint 5.



La boucle for (boucle pour)

La boucle **for** est utilisée lorsque le nombre d'itérations est connu à l'avance.

Elle permet **de parcourir une séquence** ou de **répéter un bloc** d'instructions un nombre **précis** de fois.



Exemple en pseudo-code :

```
pour i de 0 à 4 faire  
    afficher(i)  
fin pour
```

Explication :

La variable i prend les valeurs de 0 à 4.

La boucle affiche les valeurs de i (0, 1, 2, 3, 4).

