**Notion d’algorithmique : définition, syntaxe**

Algorithme vient du nom du célèbre mathématicien arabe Al Khawarizmi (Abu Ja'far Mohammed Ben Mussa Al-Khwarismi).

Un algorithme est une suite d’instructions ayant pour but de résoudre un problème donné et atteindre un résultat. Ces instructions doivent être exécutées de façon automatique par un ordinateur.

Un algorithme peut se comparer à une recette de cuisine :

* Le résultat c’est comme le plat à cuisiner.
* Les données sont l’analogue des ingrédients de la recette.
* Les règles de transformations se comparent aux directives ou instructions de la recette.
* **algorithme = méthode de résolution.**

Un algorithme doit donc être :

1. Lisible: l'algorithme doit être compréhensible même par un non-informaticien.
2. De haut niveau: l'algorithme doit pouvoir être traduit en n'importe quel langage de programmation, il ne doit donc pas faire appel à des notions techniques relatives à un programme particulier ou bien à un système d'exploitation donné.
3. Précis: chaque élément de l'algorithme ne doit pas porter à confusion, il est donc important de lever toute ambiguïté.
4. Concis: un algorithme ne doit pas dépasser une page. Si c'est le cas, il faut décomposer le problème en plusieurs sous-problèmes.
5. Structuré: un algorithme doit être composé de différentes parties facilement Identifiables.

**Notre objectif : Apprendre à écrire des programmes.**

****

1. **Analyse d’un problème**
* Comprendre le problème.
* Dégager les données en entrée.
* Dégager les données en sortie.
* Formaliser la solution trouvée par un algorithme.
1. **Génération d’un algorithme**

Exemple d’algorithme : Addition de deux nombres A et B.

* Introduire le premier nombre A
* Introduire le deuxième nombre B
* Faire l’addition A+B et mettre le résultat dans C
* Afficher C

Une opération dans un algorithme est équivalente à une action sur un objet. Dans l’algorithme précédent :

**Actions :** Introduire, faire l’addition, mettre le résultat dans, Afficher.

**Objets :** les nombres A, B, C.

**Eléments de base d’un algorithme**

**Les objets dans un algorithme**

* Un objet possède trois caractéristiques : le nom, le type et la valeur.
* Le nom (identificateur) de l’objet sert à le désigner dans l’algorithme. Il peut être alphanumérique et doit commencer par une lettre.
* Le type de l’objet caractérise l’ensemble des valeurs permises pour cet objet et les opérations qui lui sont autorisées.
* La valeur est le contenu de l’objet.
* Les constantes : une constante est un objet qui a un nom fixe, un type fixe, et une valeur fixe.
* Les variables : une variable est un objet qui a un nom fixe, un type fixe, et une valeur variable.

**Notion de variable**

* Dans les langages de programmation une variable sert à stocker la valeur d’une donnée.
* Une variable désigne en fait un emplacement mémoire dont le contenu peut changer au cours d’un programme (d’où le nom variable).
* Chaque emplacement mémoire a un numéro qui permet d'y faire référence de façon unique : c'est l'adresse mémoire de cette cellule.
* **Une variable est l'association d'un nom avec un type, permettant de mémoriser une valeur de ce type.**

Les variables doivent être déclarées avant d’être utilisées, elles doivent être caractérisées par :

1. Un nom (Identificateur).
2. Un type (entier, réel, caractère, chaîne de caractères, …).
3. Une valeur.

**Types des variables**

Le type d’une variable détermine l’ensemble des valeurs qu’elle peut prendre, les types offerts par la plus part des langages sont:

1. **Type numérique (entier ou réel)**

• **Entier :** 2, -4,….

• **Réel :** 2.5, 4.56, ….

1. **Type logique ou booléen:**

Une variable de ce type peut contenir une des valeurs logiques **VRAI** ou **FAUX**

1. **Type caractère:**

Lettres majuscules, minuscules, chiffres, symboles, …

Exemples: ’A’, ’a’, ’?’, …

1. **Type chaîne de caractère:**

Toute suite de caractères.

Exemples: " Nom, Prénom", "code postale: 1000", …

**Déclaration des variables**

* En pseudo-code, on va adopter la forme suivante pour la déclaration de variables :

**<Variables> <liste d'identificateurs> : <type>**

Exemple:

Variables i, j, k : entier

x, y : réel

OK: booléen

ch1, ch2 : chaîne de caractères

**Notion de Constante**

* Une constante est une variable dont la valeur ne change pas au cours de l'exécution du programme, elle peut être un nombre, un caractère, ou une chaine de caractères.
* En pseudo-code, on va adopter la forme suivante pour la déclaration de variables

**<Constante> <identificateur> = <valeur> : <type>**

Exemple :

Pour calculer la surface des cercles, la valeur de pi est une constante mais le rayon est une variable.

Constante PI=3.14 : réel

**Expressions et opérateurs**

* Une expression peut être une valeur, une variable ou une opération constituée de variables reliées par des opérateurs

Exemples :

1, b, a\*2, a+ 3\*b-c, …

* L'évaluation de l'expression fournit une valeur unique qui est le résultat de l'opération
* Les opérateurs dépendent du type de l'opération, ils peuvent être :

Des opérateurs arithmétiques: +, -, \*, /, % (modulo), ^ (puissance)

Des opérateurs logiques: NON ( !), OU (||), ET (&&)

Des opérateurs relationnels: =, , <, >, <=, >=

Des opérateurs sur les chaînes: & (concaténation)

* Une expression est évaluée de gauche à droite mais en tenant compte de priorités

Pour les opérateurs arithmétiques donnés ci-dessus, l'ordre de priorité est le suivant (du plus prioritaire au moins prioritaire) :

• () : les parenthèses

• ^ : (élévation à la puissance)

• \* , / (multiplication, division)

• % (modulo)

• + , - (addition, soustraction)

Exemple :

2 + 3 \* 7 vaut 23

**Instructions élémentaires**

1. **Affectation**
* L’affectation consiste à attribuer une valeur à une variable (ça consiste en fait à remplir où à modifier le contenu d'une zone mémoire).
* En pseudo-code, l'affectation se note avec le signe ←

Var← z : attribue la valeur de z à la variable Var

* z peut être une valeur, une autre variable ou une expression
* Var et z doivent être de même type ou de types compatibles
* L’affectation ne modifie que ce qui est à gauche de la flèche

Exemple :

Pour i, j, k : entier; x : réel; ok : booléen; ch1, ch2 : chaine de caractères)

Affectation valide :

 i ←1 j ←i k ←i+j

 x ←10.3 OK ←FAUX ch1 ←"SM"

 ch2 ←ch1 x ←4 x ←j

Affectation non valide :

i ←10.3 OK ←"SM" j ←x

**Remarques**

Beaucoup de langages de programmation utilisent le signe égal (=) pour l’affectation ←. Attention aux confusions:

• L'affectation n'est pas commutative : A=B est différente de B=A.

• L'affectation est différente d'une équation mathématique :

A=A+1 a un sens en langages de programmation.

A+1=2 n'est pas possible en langages de programmation et n'est pas équivalente à A=2-1.

Exemple :

x← 4

y←6

z←x+y 🡺 z prend la valeur de x+y

Ecrire(z)🡺10

y←20

Ecrire(z)🡺10 la modification de y après affectation n’a aucun effet sur la valeur de z

 **2- Instruction d’entrée/sortie**

Les instructions de lecture et d'écriture permettent à la machine de communiquer avec l'utilisateur.

**La lecture**

C’est l’action par laquelle nous pouvons introduire des données en utilisant le clavier. En pseudo-code, elle est notée par :

Lire (x) : Ce qui signifie que la machine va mettre dans la zone mémoire x la donnée tapée sur le clavier.

**L’écriture**

C’est l’action par laquelle nous pouvons communiquer un résultat ou un message à l’utilisateur par l’intermédiaire de l’écran. En pseudo-code, elle est notée par:

Écrire (z) : Ce qui signifie évaluer l’expression z et afficher le résultat.

**Structures de contrôle**

1. **La séquence**

Dans un algorithme, les instructions sont exécutées séquentiellement, c'est-à-dire, l’une après l’autre.

1. **Le test**

Il permet de choisir suivant une condition entre deux blocs d'instructions.

En pseudo-code on note :

Si (condition) Alors

 Instruction ou suite d'instructions1

Sinon

 Instruction ou suite d'instructions2

FinSi

* Une condition est une expression logique qui ne peut prendre que deux valeurs VRAI ou FAUX.
* Si la condition est vraie, se sont les instructions1 qui seront exécutées
* Si la condition est fausse, se sont les instructions2 qui seront exécutées

Exemple

Algorithme AffichageValeurAbsolue1

Variable x : réel

Début

Ecrire (" Entrez un réel : ")

Lire (x)

Si (x < 0) alors

 Ecrire ("la valeur absolue de ", x, "est:", -x)

Sinon

 Ecrire ("la valeur absolue de ", x, "est:", x)

Finsi

Fin

**Tests imbriqués**

Les tests peuvent avoir un degré quelconque d'imbrications. Un sinon se rapporte toujours au dernier si qui n’a pas encore de sinon associé.

Exemple :

Ecrire un algorithme qui teste si un entier n est positif, négatif ou bien nul.

Algorithme test

Variable n : entier

Début

Ecrire ("entrez un nombre : ")

Lire (n)

Si (n < 0) alors

 Ecrire ("Ce nombre est négatif")

Sinon

 Si (n = 0) alors

 Ecrire ("Ce nombre est nul")

 Sinon Ecrire ("Ce nombre est positif")

 Finsi

Finsi

Fin

1. **Les boucles**

Les boucles servent à répéter l'exécution d'un groupe d'instructions un certain nombre de fois.

**Les boucles Tant que**

**Syntaxe:**

Tant que (condition) Faire

Bloc d’instructions

Fin TantQue

* On y répète des instructions tant qu'une certaine condition est vérifiée.
* Si la condition est vérifiée, alors le bloc d’instructions est exécuté puis on retourne à la condition pour l’examiner.
* Si elle est vraie alors on refait le bloc d’instructions et ainsi de suite.
* Si la condition n’est pas vérifiée (fausse) alors on sort de la boucle.

Exemple :

Ecrire un algorithme qui calcule la somme des nombres de 1 à N.

Algorithme Somme\_entiers

Variable N, SOM, i : Entier

Début

Lire (N)

i←1

SOM ←0

Tant que (i<= N) Faire

SOM ←SOM + i

i←i+ 1

Fin Tantque

Écrire (SOM)

Fin

**Les boucles Pour**

**Syntaxe :**

Pour i allant de 1 à N Faire

Bloc d’instructions

Fin Pour

* Ces deux instructions permettent d’exécuter un bloc d’instructions N fois.
* Le nombre d'itérations dans une boucle Pour est connu avant le début de la boucle.
* Un compteur est utilisé pour compter le nombre courant.

Exemple :

Calcul de (X\*X\*X…\*X) N fois

Algorithme calcul

Variables X, N, Res, i : entier

Début

Lire (X)

Lire (N)

Res←1

Pour i allant de 1 à N Faire

Res←Res\* X

Fin Pour

Écrire (Res)

FIN

**Les boucles imbriquées**

Exemple: Exécution

Pour i allant de 1 à 5 OK

 Pour j allant de 1 à i OOK

 Ecrire ("O") OOOK

 FinPour OOOOK

 Ecrire ("K") OOOOOK

FinPour

**Les tableaux**

* Un tableau est un ensemble d'éléments de même type désignés par un identificateur unique
* Une variable entière nommée indice permet d'indiquer la position d'un élément donné au sein du tableau et de déterminer sa valeur.
* La déclaration d'un tableau s'effectue en précisant le type de ses éléments et sa dimension (le nombre de ses éléments)

En pseudo code :

**Variable tableau identificateur[dimension] : type**

Exemple :

Variable tableau notes[30] : réel

* L'accès à un élément du tableau se fait au moyen de l'indice.

Exemple :

notes[i] donne la valeur de l'élément i du tableau notes.

Exercice :

Pour le calcul du nombre d'étudiants ayant une note supérieure à 10 avec les tableaux, on peut écrire :

Algorithme calcul\_tab

Variables i ,nbre : entier

Tableau notes[30] : réel

Début

nbre ← 0

Pour i allant de 0 à 29

Si (notes[i] >10) alors

nbre ←nbre+1

FinSi

FinPour

Ecrire ("le nombre de notes supérieures à 10 est : ", nbre)

Fin

**Les tableaux suivants contiennent les instructions Fortran équivalentes aux mots clés utilisés dans les algorithmes :**

|  |  |
| --- | --- |
| Algorithme  | Program Fortran  |
| ← | = |
| ≤ | <= |
| ≥ | >= |
| = | == |
| ≠ | /= |
| x MOD y | MOD (x,y) |
| et | .and. |
| ou | .or. |
| non | .not. |
| Constante | Parameter () |
| vrai | .true. |
| faux | .false. |
| longueur () | Len () |
| Lire | READ \*, |
| Ecrire | PRINT \*, |
| Fin | END PROGRAM |
| Si | IF |
| Sinon | Else |
| Finsi | Endif |
| Alors | Then |
| Entier | INTEGER |
| Réel | REAL |
| Caractère | CHARACTER |
| Chaîne | CHARACTER\* |
| Booléen | LOGICAL |
| Pour i de 1 à n faire | DO i = 1, n |
| Finpour | ENDDO |
| Tantque Cond. faire | DO WHILE (Cond.) |
| FinTQ | ENDDO |
| Pour i de 1 à n pas de 5 faire | DO i = 1, n, 5 |