Algorithmique

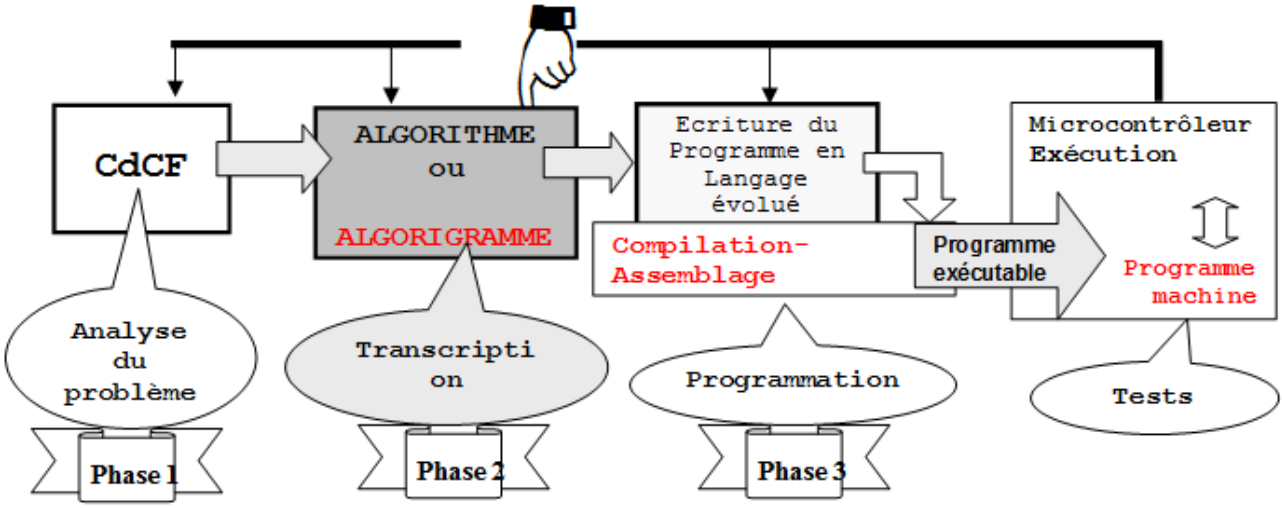
Table des matières

Un algorithme énonce une résolution sous la forme d'une série d'opérations à effectuer. La mise en œuvre de l'algorithme consiste en l'écriture de ces opérations dans un langage de programmation et constitue alors la brique de base d'un programme informatique.

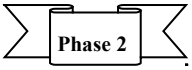
1. Introduction

Un programme informatique a pour objectif de faire exécuter une suite de travaux à une machine mais pour cela, il faut que la machine puisse comprendre les ordres d'un opérateur humain.

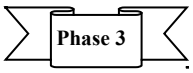
Le développement d'un programme nécessite trois phases.



Cahier Des Charges Fonctionnel (**CdCF**) : expression en français ou avec des outils de spécification du besoin.



ALGORITHMIQUE (Analyse structurée du problème) : représentation sous forme graphique ou en pseudo code.



Traduction dans un langage « de programmation »
Les mots-clés ou les symboles graphiques sont traduits en langage de programmation.

2. Algorithmique

L'algorithmique est l'ensemble des règles et des techniques qui sont impliquées dans la définition et la conception d'algorithmes.

Un algorithme est une suite finie et non-ambiguë d'instructions permettant d'arriver, en un temps fini, à un résultat déterminé, à partir d'une situation donnée.

Pour concevoir un algorithme trois étapes sont nécessaires :

1. **La préparation du traitement** : recherche des données nécessaires à la résolution d'un problème.
2. **Le traitement** : résolution pas à pas du problème après décomposition en plusieurs sous-ensembles si nécessaire.
3. **L'édition** des résultats.

2.1. Modes d'expression d'un algorithme

Un algorithme peut être écrit :

- en langage littéral (pseudo code) par l'utilisation de mots-clés et de délimiteurs. On parle de langage de description d'algorithme.

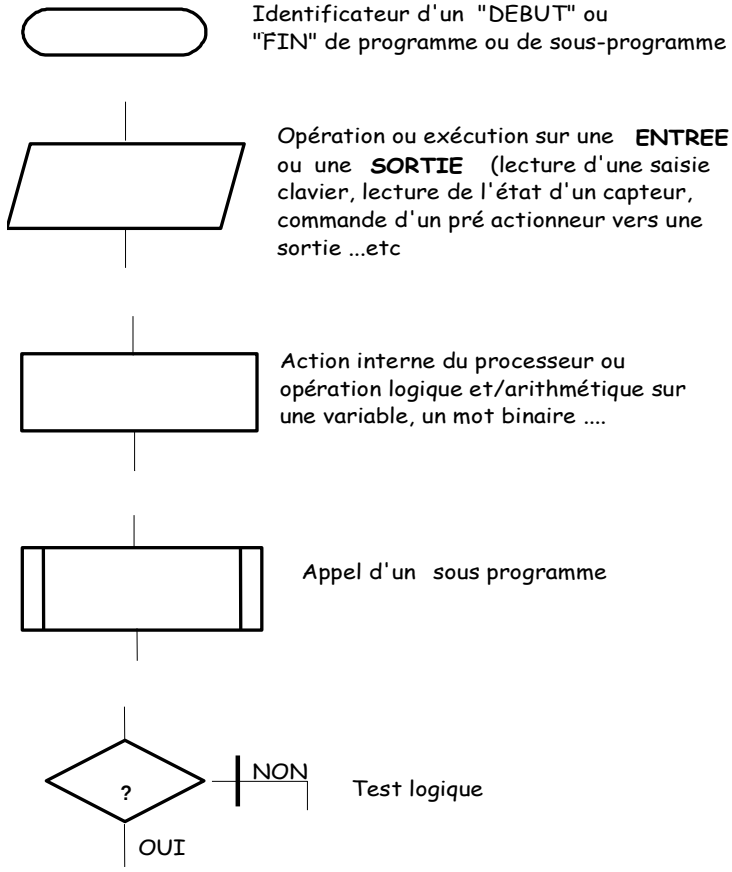
  L'algorithme est bien adapté pour le codage dans un langage de haut niveau (Ex : Java, C#, C, C++, Python, PHP, Javascript).

Exemples de mots-clés	Exemples de mots délimiteurs
Lire – Ecrire **si… alors… sinon** **tant que …… faire**	Ils fixent les bornes des entrées et des sorties des structures algorithmiques : **début - fin - finsi**

- graphiquement (utilisation de symboles normalisés), on parle d'algorigramme ou d'organigramme de programmation.

  L'algorigramme est plus adapté au codage dans un langage de bas niveau (Ex : assembleur) ou pour l'apprentissage de l'algorithmique (Ex : logiciel Flowcode).

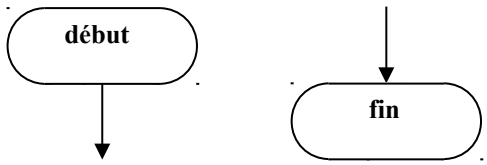
PRINCIPAUX SYMBOLES UTILISES (NF Z 67-010)



2.2. Organisation d'un algorithme

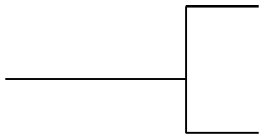
2.2.1. Le programme principal

Le programme principal consiste en une suite d'opérations élémentaires faisant souvent appel à des fonctions ou procédures.Il est délimitée par des primitives algorithmiques (mots clés) ou des schémas :

Algorithme	**Algorigramme** (NF Z 67-010)
début .... **fin**	

2.2.2. Les commentaires

Des commentaires doivent être insérés dans le programme afin d'en faciliter la relecture.

Algorithme	**Algorigramme** (NF Z 67-010)
{ Commentaire **}**	

2.3. Les structures algorithmiques

Les structures algorithmiques sont réparties en 3 catégories :

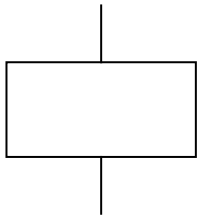
- structures linéaire d'opérations
- structures alternatives (ou conditionnelles) ou de choix : en fonction d'une condition, le programme exécute des opérations différentes
- structures itératives ou répétitives: sous contrôle d'une condition, une séquence d'opérations est exécutée répétitivement

2.3.1. Les structures linéaires

- **L'affectation**

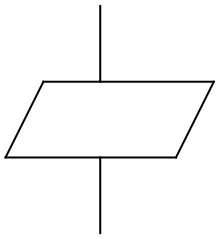
  Cette action attribue une **valeur** (une constante ou le résultat d'un traitement) à une variable.

  On notera cette action par le symbole : ← ou :=

Algorithme	**Algorigramme** (NF Z 67-010)
Nom_Variable **:=** *valeur*	

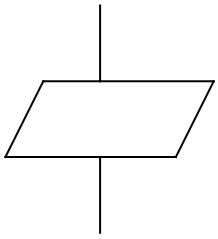
L'affectation n'a de sens que si les deux objets de part et d'autre du signe := sont de même type.

- **La saisie d'une valeur sur un terminal**

Algorithme	**Algorigramme** (NF Z 67-010)
Lire( )	

On pourra lire une variable seule, de type quelconque, ou un ensemble de variables séparées par des virgules, de type identique ou différent.

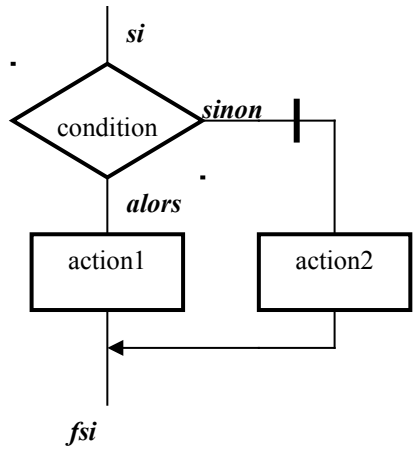
- **L'édition de résultats sur un périphérique de sortie**

Algorithme	**Algorigramme** (NF Z 67-010)
Ecrire( )	

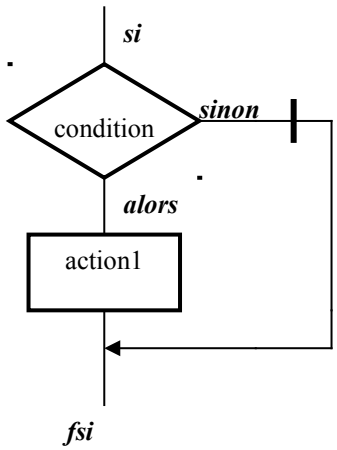
On pourra écrire les valeurs d'une ou plusieurs variables.

2.3.2. Les structures alternatives

- **La structure alternative de base**

Algorithme	**Algorigramme** (NF Z 67-010)
si (*condition vraie)* ***alors*** *action1* **sinon** *action2* **finsi**	

Si **action2** est vide, on écrira :

Algorithme	**Algorigramme (**NF Z 67-010)
si (*condition vraie)* ***alors*** *action1* ***finsi***	

2.3.3. Les structures itératives ou répétitives

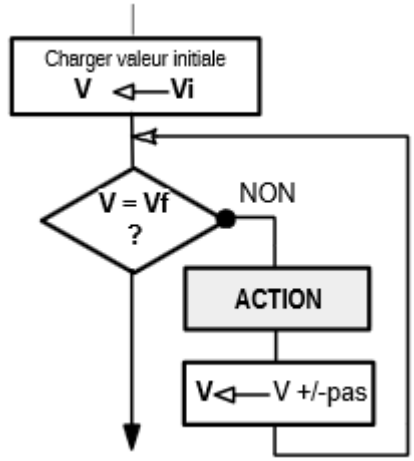
Ces structures permettent d'exécuter plusieurs fois une séquence d'instructions.

Pour que l'itération puisse se terminer, il faut que l'exécution de l'action modifie la valeur de l'expression conditionnelle évaluée.

Pour cela, il faut veiller à initialiser la ou les variables intervenant dans la condition d'évolution (test) de l'itération.

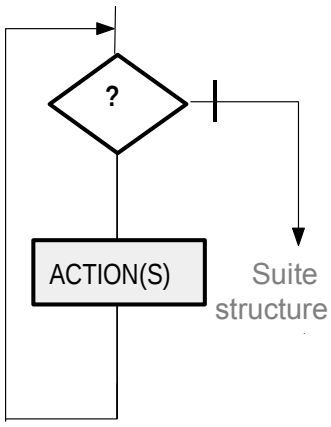
- Le nombre d'itérations est connu (boucle de comptage)

  Lorsque le nombre d'itérations est connu, on peut utiliser une variable auxiliaire (compteur de boucle) dont la valeur caractérise le nombre de passages dans la boucle.

Algorithme	**Algorigramme** (NF Z 67-010)
Variables i : entier; **pour i** <*valeur_initiale*> **à** <*valeur_finale*> **par pas de** <*n*> **Faire** *Action(s);* **FinFaire**	

  Remarque : pour des valeur de n = 1, l'instruction « par pas de » est optionnelle.

- Le nombre d'itérations est inconnu (test en tête de boucle)

Algorithme	**Algorigramme** (NF Z 67-010)
Tantque (*condition vraie*) **Faire** *Action(s)* **FinFaire**	

- Le nombre d'itérations est inconnu (test en fin de boucle)
  L'instruction est réalisée au moins UNE fois.

Algorithme	**Algorigramme** (NF Z 67-010)
Faire *Action(s)* **Tantque** (*condition vraie)*