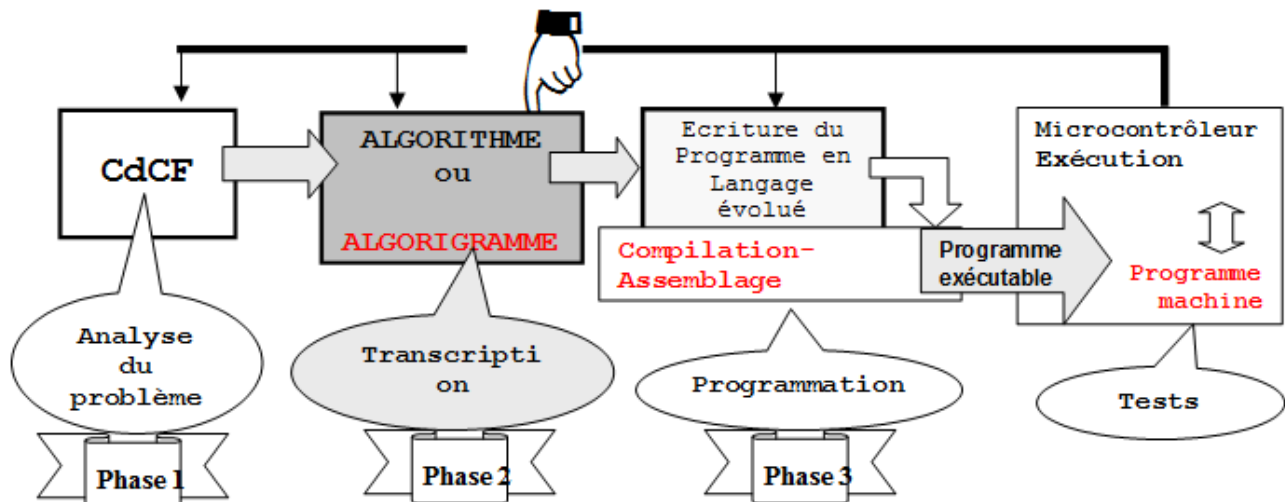


ALGORITMIQUE

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**.



- Phase 1 :** **Cahier Des Charges Fonctionnel (CdCF)** : expression en français ou avec des outils de spécification du besoin.
- Phase 2 :** **ALGORITHMIQUE** (Analyse structurée du problème) : représentation sous forme graphique ou en pseudo code.
- Phase 3 :** **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.

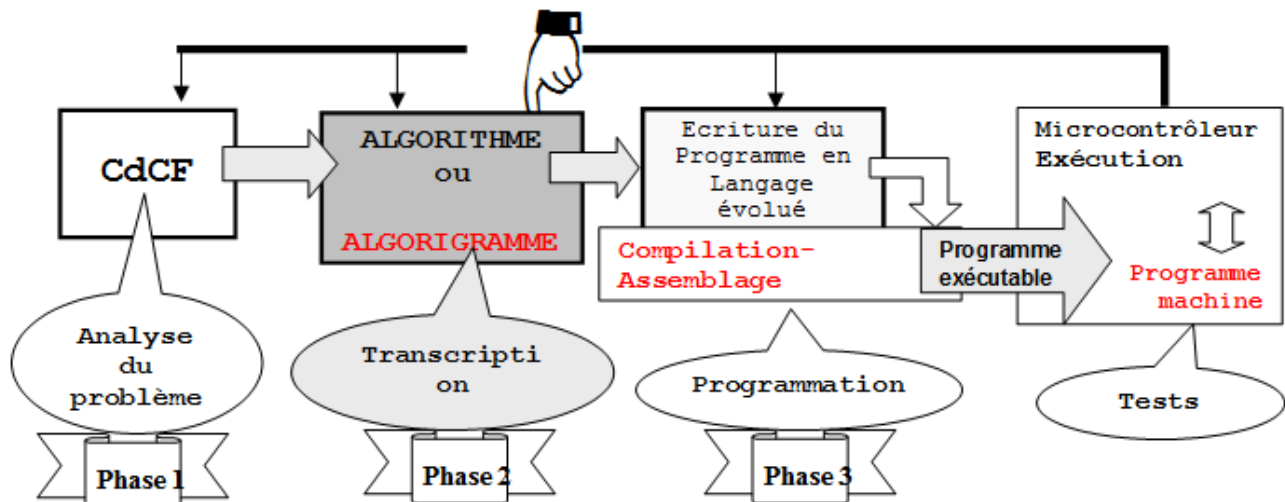
Pour concevoir un algorithme **trois étapes** sont nécessaires :

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.

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**.



- Phase 1** : **Cahier Des Charges Fonctionnel (CdCF)** : expression en français ou avec des outils de spécification du besoin.
- Phase 2** : **ALGORITHMIQUE** (Analyse structurée du problème) : représentation sous forme graphique ou en pseudo code.
- Phase 3** : **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.

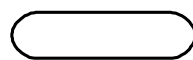
L'algorithme est bien adapté pour le codage dans un langage de **haut niveau** (Ex : Java, C#, C, C++, Python, PHP, Javascript).

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 - fins

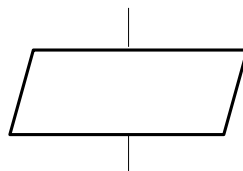
- graphiquement (utilisation de symboles normalisés), on parle d'algorithmigramme ou d'**organigramme** de programmation.

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

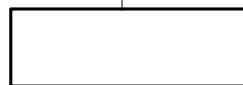
PRINCIPAUX SYMBOLES UTILISES (NF Z 67-010)



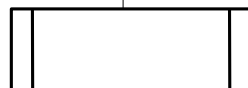
Identificateur d'un "DEBUT" ou "FIN" de programme ou de sous-programme



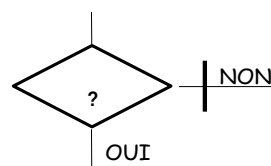
Opération ou exécution sur une **ENTREE** ou une **SORTIE** (lecture d'une saisie clavier, lecture de l'état d'un capteur, commande d'un pré actionneur vers une sortie ...etc



Action interne du processeur ou opération logique et/arithmétique sur une variable, un mot binaire



Appel d'un sous programme

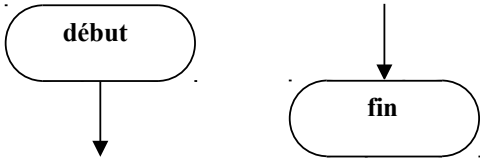


Test logique

2.2. Organisation d'un algorithme

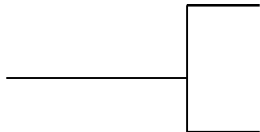
2.2.1. Le programme principal

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 :

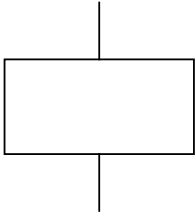
- structures linéaire d'opérations
- structures alternatives (ou conditionnelles) ou de choix : en fonction d'une condition, le programme exécute des opérations différentes
- structures itératives ou répétitives: sous contrôle d'une condition, une séquence d'opérations est exécutée répétitivement

2.3.1. Les structures linéaires

- **L'affectation**

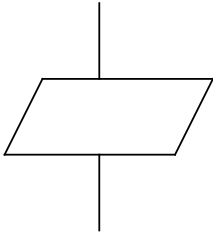
Cette action attribue une **valeur** (une constante ou le résultat d'un traitement) à une variable.

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

Algorithme	Algorigramme (NF Z 67-010)
<i>Nom_Variable := valeur</i>	

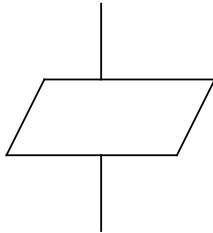
L'affectation n'a de sens que si les deux objets de part et d'autre du signe := sont de même type.

- **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)
<pre>si (condition vraie) alors action1 sinon action2 finsi</pre>	<pre>graph TD Start(()) -- si --> Cond{condition} Cond -- alors --> Act1[action1] Cond -- sinon --> Act2[action2] Act1 --> Join(()) Act2 --> Join Join -- finsi --> End(())</pre>

Si **action2** est vide, on écrira :

Algorithme	Algorigramme (NF Z 67-010)
<pre>si (condition vraie) alors action1 finsi</pre>	<pre>graph TD Start(()) -- si --> Cond{condition} Cond -- alors --> Act1[action1] Cond -- sinon --> Join(()) Act1 --> Join Join -- finsi --> End(())</pre>

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	Algorithme (NF Z 67-010)
Variables <i>i</i> : entier; pour <i>i</i> <valeur_initiale> à <valeur_finale> par pas de <n> Faire <i>Action(s)</i> ; FinFaire	<pre> graph TD Start(()) --> Init[Charger valeur initiale V ← Vi] Init --> Decision{V = Vf ?} Decision -- NON --> Action[ACTION] Action --> Update[V ← V +/- pas] Update --> Decision Decision -- OUI --> Exit(()) </pre>

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	Algorithme (NF Z 67-010)
Tantque (condition vraie) Faire <i>Action(s)</i> FinFaire	<pre> graph TD Start(()) --> Decision{?} Decision -- Vrai --> Action[ACTION(S)] Action --> Decision Decision -- Faux --> Exit[Suite structure] </pre>

- 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 <i>Action(s)</i> Tantque (<i>condition vraie</i>)	<pre> graph TD Entry(()) --> Action[ACTION (S)] Action --> Decision{?} Decision --> Entry Decision --> Exit[Suite structure] </pre>

1. Réaliser un algorithme :

- Lire le code barre d'un produit puis afficher le prix et le nom.
- Mesurer le poids d'une cartouche d'encre si la cartouche pèse plus de 100 grammes, elle sera détruite.
- idem b sauf que si la cartouche pèse 100 grammes ou moins elle sera étiquetée conforme.