Chapitre 3 : Algorithmes et organigrammes

Lorsque nous sommes confrontés à **des problèmes complexe**s, il est essentiel d'avoir une approche structurée pour les résoudre de manière efficace. C'est là que les **algorithmes et les organigrammes** entrent en jeu. Dans le domaine de l'informatique et de la programmation, les algorithmes et les organigrammes sont des outils puissants qui nous aident à concevoir et à représenter visuellement des solutions étape par étape.

*Un **algorithme** peut être défini comme une série d'instructions logiques et précises pour résoudre un problème donné. Il s'agit d'une **séquence d'étapes** bien définies qui permettent d'atteindre un objectif spécifique. Les algorithmes sont utilisés dans de nombreux domaines, de **la science informatique aux mathématiques, en passant par la biologie et l'ingénierie. Ils sont essentiels pour automatiser des processus, trouver des solutions optimales et prendre des décisions éclairées.***∗

Pour représenter graphiquement un algorithme, nous utilisons des **organigrammes**. Un organigramme est une représentation visuelle qui utilise des symboles et des formes pour représenter les différentes étapes d'un algorithme. Les organigrammes offrent une vue d'ensemble claire et structurée de l'algorithme, ce qui facilite la compréhension et l'analyse de sa logique.

L'utilisation d'algorithmes et d'organigrammes présente de nombreux avantages. Tout d'abord, ils permettent de décomposer des problèmes complexes en étapes plus petites et plus gérables, ce qui facilite leur résolution. En utilisant un algorithme bien conçu, il devient plus simple de suivre une séquence d'instructions logiques pour atteindre l'objectif souhaité.

De plus, les algorithmes et les organigrammes favorisent la ré utilisabilité du code. Une fois qu'un algorithme a été conçu et testé, il peut être utilisé à plusieurs reprises pour résoudre des problèmes similaires. Cela permet d'économiser du temps et des ressources, car il n'est pas nécessaire de réinventer la roue à chaque fois.

Enfin, les algorithmes et les organigrammes favorisent la collaboration et la communication efficace entre les programmeurs et les concepteurs de systèmes. En utilisant un langage visuel commun, il devient plus facile de partager des idées, d'expliquer des concepts et de travailler en équipe pour résoudre des problèmes complexes.

Dans ce cours, nous explorerons en détail les concepts d'algorithmes et d'organigrammes. Nous apprendrons à concevoir des algorithmes efficaces, à représenter graphiquement ces algorithmes à l'aide d'organigrammes et à analyser leur complexité. Nous verrons également comment utiliser ces outils pour résoudre des problèmes informatiques concrets([71]

1. Algorithme

1.1. Qu'est-ce qu'un algorithme ?

Définition

*Un **algorithme** est une séquence **d'instructions logiques** et précises conçue pour résoudre un problème spécifique ou atteindre un objectif donné. C'est un ensemble étape par étape d'opérations bien définies qui permet de résoudre un problème de manière efficace et fiable.*∗

[1] Introduction to the Design and Analysis of Algorithms" par Anany Levitin

Les algorithmes existent dans de nombreux domaines, de l'informatique à la mathématique, en passant par la physique, la biologie et bien d'autres encore. Ils sont utilisés pour automatiser des processus, trouver des solutions optimales, prendre des décisions éclairées et résoudre des problèmes complexes.

Un algorithme est généralement composé de plusieurs éléments clés :

Données d'entrée : Ce sont les informations fournies à l'algorithme au début du processus. Les données d'entrée peuvent être des valeurs numériques, des chaînes de caractères, des tableaux, etc.

Opérations et instructions : Ce sont les actions spécifiques effectuées par l'algorithme pour traiter les données d'entrée. Il peut s'agir de calculs mathématiques, de comparaisons, de boucles, de conditions, de lectures ou d'écritures de données, etc.

Données de sortie : Ce sont les résultats produits par l'algorithme après avoir traité les données d'entrée. Les **données de sortie** peuvent être des valeurs numériques, des messages, des résultats de calculs, des tableaux modifiés, etc.

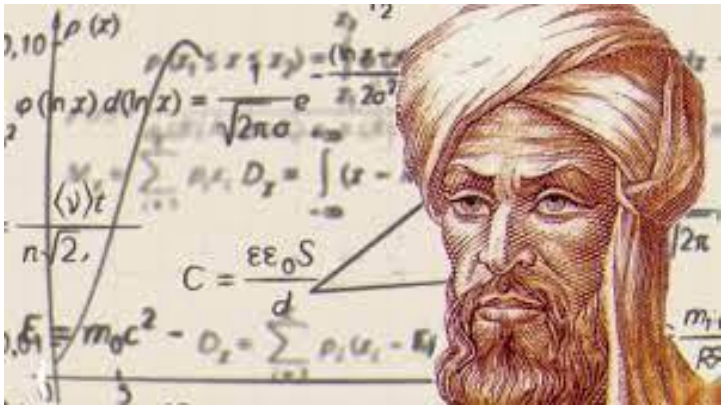
Séquentialité : Les instructions de l'algorithme sont exécutées dans un ordre spécifique, étape par étape. Chaque étape dépend généralement des résultats des étapes précédentes.

Terminaison : Un algorithme doit finir son exécution après un nombre fini d'étapes. Il doit atteindre un point où il ne reste plus d'instructions à exécuter.

Le mot algorithme vient du nom du célèbre mathématicien **arabe *Al Khawarizmi** (ABU JA'FAR**

MOHAMMED BEN MUSSA AL-KHWARISMI) origine de l"ancienne ville de KHAWARISM aujourd"hui

« KHIVA » située en ex-U.R.S.S. « algorithmique » a été conçu en 1958 par des chercheurs universitaires



Al Khawarizmi

Critère d'un algorithme ***Fondamental***

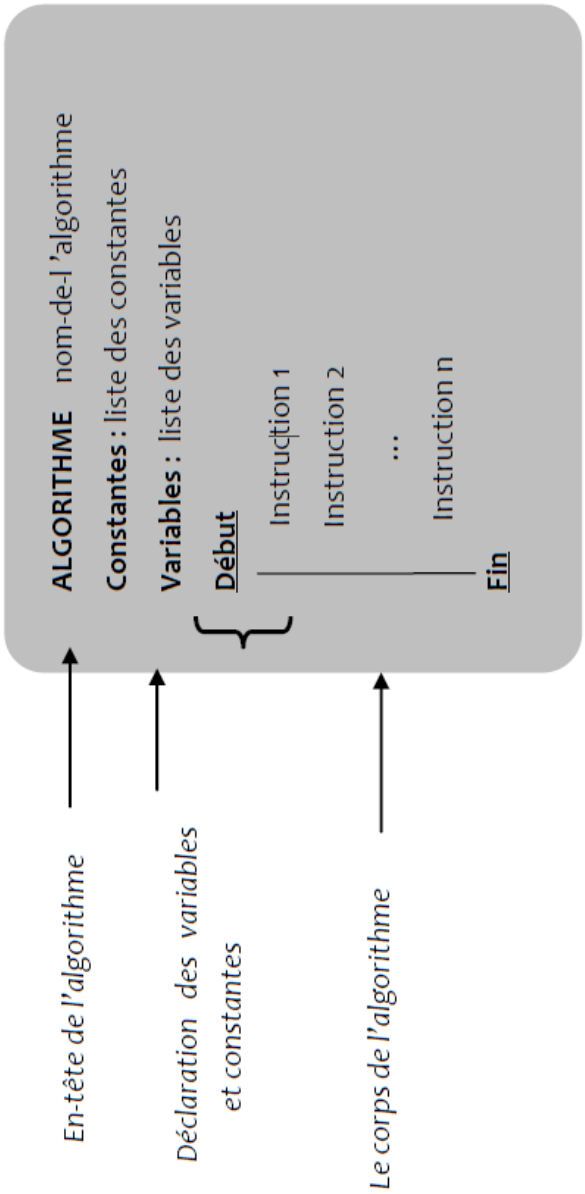
La bonne connaissance de l"algorithmique permet d"écrire des algorithmes exacts et efficaces, or c"est en écrivant des algorithmes corrects, qu"on devient un bon Programmeur.

*Un algorithme correct doit donc subit à certains critères qui sont :**

- **être fini (achevé après un nombre fini d"actions élémentaires), être précis (la machine n"a pas à choisir)**
- **être effectif (On pourrait le traiter " à la main ") et mentionner les entrées (saisie de données) et les sorties (affichage des résultats)**

1.2. Structure de l'algorithme

comment structure un algorithme ? *Définition*



Structure générale d'un algorithme

L'en-tête : permet tout simplement d"identifier l"algorithme.

Déclaration : liste de toutes les constantes et variables utilisées dans l"algorithme.

Le corps : cette partie contient les ordres (instructions) ou les tâches de l"algorithme

Fin avec le nom de l'algorithme

2. Organigramme

*Un **organigramme** est un diagramme graphique utilisé pour représenter visuellement la structure et le flux d'un algorithme, d'un processus ou d'un programme. Il utilise **des formes géométrique**s et **des connexions** pour illustrer les différentes étapes, **les décisions conditionnelles**, l**es boucles et les actions** à effectuer.*

2.1. Comment représenter un organigramme ?

Comment représenter un organigramme ? *Définition*

Un organigramme est une représentation **graphique** d"un algorithme, il permet de schématiser graphiquement la solution d"un problème. Un organigramme permet de mieux visualiser la démarche de résolution d"un problème, il est construit à partir d"un formalisme comprenant cinq simples symboles normalisés qui sont reliés entre eux par des lignes de liaisons., ces symboles sont :

Symboles normalisés *Fondamental*

les éléments clés utilisés dans la création d'un organigramme :∗

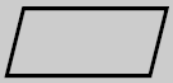
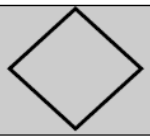
Boîtes ou rectangles : Les boîtes sont utilisées pour représenter les étapes individuelles de l'algorithme. Chaque boîte contient une instruction ou une action spécifique à effectuer.

Flèches ou lignes de connexion : Les flèches ou les lignes relient les boîtes entre elles pour indiquer le flux d'exécution de l'algorithme. Elles montrent l'ordre séquentiel des étapes à suivre.

Les diamants sont utilisés pour représenter les points de décision où une condition est évaluée. Selon le résultat de la condition, le flux de l'algorithme peut bifurquer vers différentes directions.

Boucles : Les boucles sont représentées par des formes spécifiques, telles que des cercles ou des rectangles avec des flèches de retour. Elles indiquent qu'un ensemble d'instructions doit être répété jusqu'à ce qu'une condition spécifique soit satisfaite.

Entrée/Sortie : Les formes spéciales, comme les parallélogrammes, sont utilisées pour représenter les étapes d'entrée et de sortie de données. Elles indiquent où les données sont fournies à l'algorithme ou où les résultats sont affichés.

Les différentes figures d'organigramme	
Symbole	Désignation
	Représente le début et la fin de l'organigramme.
	Entrées / Sorties : Lecture des données et écriture des résultats.
	Calculs, Traitements.
	Tests et décision : on écrit le test à l'intérieur du losange.
	Ordre d'exécution des opérations (Enchaînement).

Symboles utilises pour organigramme

3. Structure des données

En informatique, une structure de données est une manière d'organiser les données pour les traiter plus facilement. Différentes structures de données existent pour des données différentes : constantes, variables, enregistrements, structures composées finies , tableaux (sur [1..n]), listes, arbres, graphes

3.1. Comment déclarer les variables?

Comment déclarer les variables?

Une variable est une place mémoire où est stockée une donnée sous forme d"octets. L"identificateur de cette variable permet d"avoir accès à ces données sans être obligé de travailler sur les octets. Les variables peuvent être de différents types, par exemple : **entier (INTEGER), réel (REAL), Booléen (BOOLEAN), caractère (CHAR), chaîne de caractères (STRING)** ..etc.

- **Type entier** : représentant un nombre entier quelconque exemple : (1, 5, -9000, 1256, 98, -45)
- **Type réel :** représentant un nombre réel quelconque exemple :(1.5, 5.0, -90.125, 1.256, 9.8, -45.0)
- **Caractère** : représentant un caractère seul exemple : (‘a', 'b', ' ‘, '7', '/', '^', ‘R', ‘.')
- Chaîne de caractères ou String : représentant un texte de zéro, un ou plusieurs caractères. Le nombre

maximal de caractères pouvant être stockés dans une seule variable string dépend du langage utilisé. Un caractère sera noté avec une apostrophe simple (exemple 'c') et le string sera notée entre guillemets doubles (exemple "contenu de la chaine").

- **Type booléen** : représentant une valeur logique binaire oui ou non, ouvert ou fermé, vrai ou faux. On peut représenter ces notions abstraites de VRAI et de FAUX par tout ce qu'on veut : de l'anglais (***TRUE et FALSE***) ou des nombres (0 et 1). Peu importe. Ce type booléen est très économique en termes de place mémoire occupée, puisque pour stocker une telle information binaire, un seul bit suffit.

3.2. Comment déclarer les constantes ?

Comment déclarer les constantes ?

Les constantes sont des entités permettant de réserver de l'espace mémoire pour stocker des données dont

leur valeur ne peuvent pas être modifiées au cours de l"exécution de l"algorithme (fixée pour tout

l"algorithme), elles peuvent être de différentes natures : entière, réelle, booléenne, caractère, ...etc.

```
Constantes :
Ident = Valeur
```

Ident : c"est l"identificateur, c'est-à-dire le nom de la constante, il est composé de lettres et de chiffres.

Valeur : c"est la valeur de la constante, cette valeur restera inchangée pendant l"exécution de l"algorithme

4. Structures de contrôle

4.1. Les instructions conditionnelles

Les instructions conditionnelles ***Définition***

*Les **instructions conditionnelles** choisissent ou annulent l'exécution d'une suite d'ordres selon une condition bien définie. On en distingue trois types :*∗

Instructions conditionnelles[8][1]

instructions conditionnelles simple	instructions conditionnelles inter native	instructions conditionnelles en choix

`Si (Condition) Alors :`
`Instruction1`
`Instruction2`
`...`
`Instruction N`
`Fin si`
	`Si (Condition) Alors :`
	`Instruction1`
	`...`
	`Instruction N`
	`Sinon`
	`Instruction1`
	`...`
	`Instruction M`
	`Fin si`
		`Selon le cas (Variable) :`
		`Variable = Valeur1 : Instruction (s)`
		`Variable = Valeur2 : Instruction (s)`
		`...`
		`Variable = ValeurN : Instruction (s)`
		`Sinon`
		`Instruction(s)`
		`Fin selon`

4.2. Structures de contrôles répétitives

Structures de contrôles répétitives *Définition*

Les instructions répétitives, appelées aussi les instructions **itératives** ou encore les **boucles**, permettent de répéter plusieurs fois l'exécution d'une même suite d'instructions. On en distingue trois types :

- **La Boucle « Pour »**
- **La boucle « Tant que »**
- **La Boucle« Répéter »**

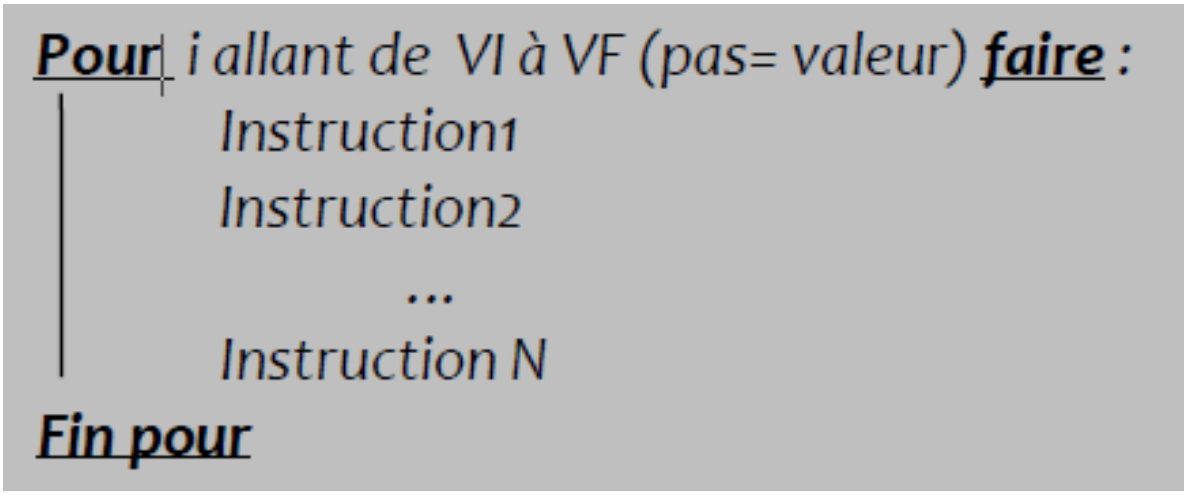
La Boucle « Pour » *Fondamental*

La boucle « **Pour**» permet de répéter l'exécution d'une suite **instructions** un nombre de fois

connu d'avance. Pour cela, il faut préciser la valeur initiale et la valeur finale et éventuellement le pas (lorsqu'il est différent de 1).

[1] https://en.wikipedia.org/wiki/Algorithm)

Cette boucle a la structure suivante :



Boucle (pour)

avec :

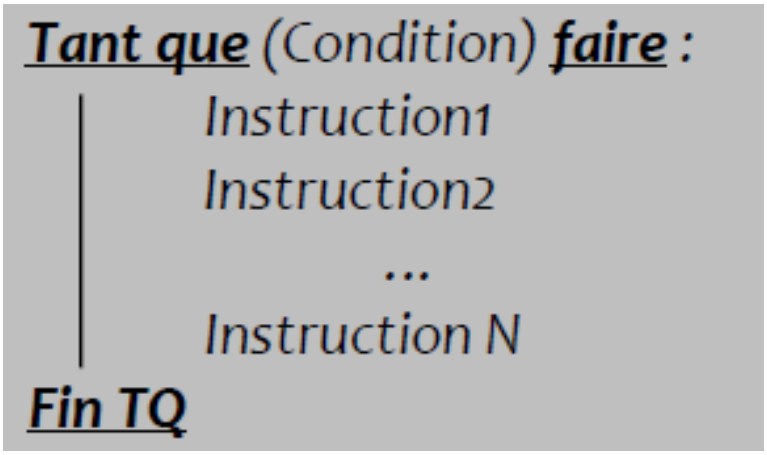
VI : valeur initiale.

VF : valeur finale.

Boucle tant que

Fondamental

La boucle « Tant que» permet de répéter l'exécution d'une suite instructions tant qu'une certaine condition est remplie. Cette boucle a la structure suivante :

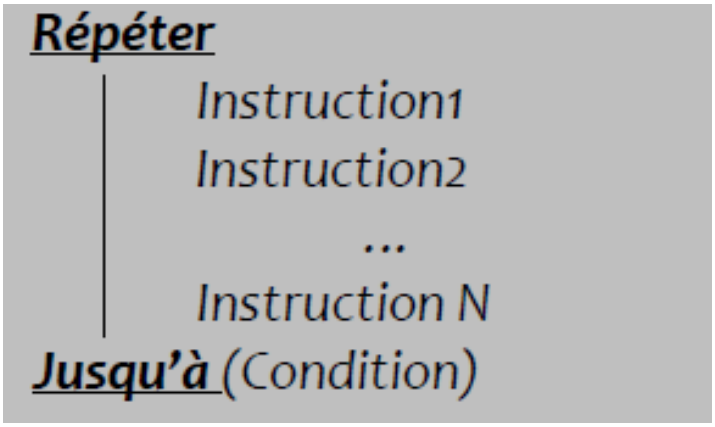


Boucle (Tant que)

La suite d'instructions de la boucle « Tant que » sera ré-exécutée tant que la condition restera vraie.

La boucle répéter *Fondamental*

La boucle « Répéter» permet de répéter l'exécution d'une suite instructions tant qu'une certaine condition n'est pas remplie, c'est-à-dire, on sort de la boucle « Répéter » quand la condition sera satisfaite. Cette boucle a la structure suivante :



la boucle « Répéter

5. Exercices

Exercice 1 : Organigramme

[solution n°13 p. 49]

Quelle est la fonction principale d'un organigramme ?

- ☐ Illustrer la hiérarchie et les relations entre les différents éléments d'une organisation.
- ☐ Représenter graphiquement les données d'un système informatique.
- ☐ Analyser les performances d'un algorithme

Exercice 2 : Algorithme

[solution n°14 p. 49]

Qu'est-ce qu'un algorithme ?

- ☐ Une commande informatique
- ☐ Un langage de programmation
- ☐ Une séquence d'instructions pour résoudre un problème