Algorithmique 1

Notion d'Informatique et d'Algorithme

Tarek Boutefara

2025-10-31

Table of Contents

Chapitre 1. Introduction

Cette partie introductive définit l'informatique en tant que discipline et présente l'algorithmique comme la méthodologie au cœur de la résolution de problèmes.

Chapitre 2. L'informatique

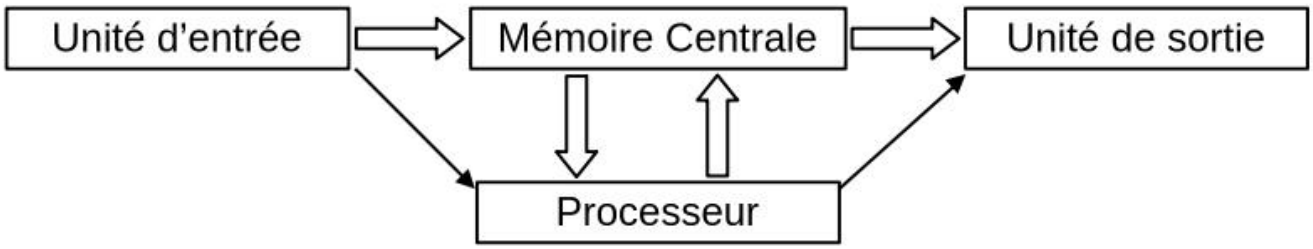
L'informatique a cessé d'être une simple discipline scientifique pour devenir l'un des piliers fondamentaux de notre société contemporaine. En l'espace de quelques décennies, elle a radicalement transformé notre façon de vivre, de travailler et d'interagir. Sa présence est désormais omniprésente : des appareils mobiles que nous consultons au réveil aux systèmes complexes qui gèrent les transports, la finance et la santé, en passant par les outils de communication qui abolissent les distances. Elle n'est plus seulement un outil de calcul, mais une infrastructure essentielle qui façonne le monde moderne, rendant sa compréhension indispensable pour quiconque souhaite décrypter et participer activement à l'ère numérique.

Le terme « **informatique** » lui-même est d'origine française. Proposé par l'ingénieur Philippe Dreyfus en 1962, ce mot-valise résulte de la contraction de deux concepts clés : « **information** » et « **automatique** ». Cette étymologie capture l'essence même de la discipline. L'informatique est la science du traitement rationnel, et notamment automatique, de l'information, considérée comme le support des connaissances et des communications. Elle englobe l'étude des méthodes, des techniques et des outils (principalement l'ordinateur) permettant de collecter, conserver, traiter et transmettre efficacement des données.

L'outil principal et indissociable de l'informatique est l'ordinateur. Faisant écho au paragraphe précédent, le terme « **ordinateur** » est également d'origine française. Il fut suggéré en 1955 par le philologue Jacques Perret, à la demande d'IBM France, pour remplacer le terme anglais "computer" (jugé trop restrictif, car axé sur le calcul) et dérive du latin ordinator, désignant celui qui met de l'ordre. Fondamentalement, un ordinateur est une machine électronique programmable conçue pour manipuler des données. Son fonctionnement essentiel peut être résumé comme un système qui accepte des informations en **entrée** (via un clavier, une souris, un réseau), exécute une suite d'opérations logiques et arithmétiques sur celles-ci (le **traitement**), et enfin restitue les résultats obtenus en **sortie** (sur un écran, une imprimante ou vers un autre système).

Pour réaliser ce traitement, l'ordinateur s'appuie sur des composants matériels essentiels. Au cœur de ce système se trouve le Processeur (ou CPU, Central Processing Unit), véritable cerveau de la machine chargé d'exécuter les instructions. Celui-ci est principalement composé de deux sous-éléments :

Architecture d'un système informatique



1. **L'Unité Arithmétique et Logique (UAL)**, qui effectue toutes les opérations de calcul (comme les additions, les soustractions) et les opérations logiques (comme les comparaisons).

2. **L’Unité de Contrôle (UC)**, qui joue le rôle de chef d’orchestre ; elle lit les instructions, les décode et commande les autres composants (UAL, mémoire…) pour les exécuter dans le bon ordre.

À côté du processeur se trouve la **Mémoire Principale** (souvent appelée RAM, Random Access Memory). Il s’agit d’un espace de stockage rapide mais volatil (elle perd son contenu lors de la mise hors tension) où l’ordinateur place les instructions et les données qu’il est en train de traiter activement.

Chapitre 3. L’algorithme

Pour que le processeur et la mémoire puissent effectuer les traitements attendus, ils doivent recevoir une séquence d’ordres précis. Cette suite d’instructions, directement compréhensible par la machine ou son système d’exploitation, est appelée un programme. Cependant, avant de pouvoir écrire un programme, il est indispensable de concevoir la logique de la solution au problème posé. C’est le rôle de l’algorithme. L’algorithme est une description abstraite et théorique des étapes à suivre, formulée de manière indépendante de tout langage de programmation. Le programme, quant à lui, est la traduction de cet algorithme dans un langage de programmation spécifique (comme Pascal, C, Python ou Java), le rendant ainsi bas niveau et concret, prêt à être exécuté.

Bien avant d’être un concept central en informatique, il faut comprendre que les algorithmes sont omniprésents dans notre vie quotidienne. Une recette de cuisine détaillée, un mode d’emploi pour assembler un meuble, ou même l’itinéraire que nous suivons pour nous rendre au travail sont, dans leur essence, des algorithmes. Ces exemples illustrent que l’algorithme est avant tout un outil intellectuel fondamental et puissant, utilisé par l’être humain pour décrire la solution d’un problème. Il permet de formaliser une démarche de résolution en la décomposant en une suite d’étapes élémentaires, claires, ordonnées et finies.

Chapitre 4. Exemple d’un algorithme

Voici un exemple très classique illustrant une démarche algorithmique dans la vie courante : la recette du **gâteau au yaourt**. Elle est souvent considérée comme simple car le pot de yaourt vide sert d’unité de mesure principale.

Objectif (Sortie) : Obtenir un gâteau cuit.

Ingrédients et Matériel (Entrées) :

- 1 pot de yaourt nature
- 2 pots de sucre
- 3 pots de farine
- 1/2 pot d’huile
- 2 œufs
- 1 sachet de levure chimique
- 1 saladier, 1 moule, 1 four

Instructions (Traitement) :

Début

1. Préchauffer le four à 180°C (Thermostat 6).
2. Vider le pot de yaourt dans le saladier.
3. Utiliser le pot vide comme mesureur.
4. Ajouter dans le saladier : les 2 œufs et les 2 pots de sucre.
5. Mélanger le tout.
6. Ajouter les 3 pots de farine et le sachet de levure.
7. Mélanger jusqu’à obtenir une pâte homogène (sans grumeaux).
8. Ajouter le 1/2 pot d’huile et mélanger une dernière fois.
9. Beurrer et fariner le moule.
10. Verser la pâte dans le moule.
11. Mettre au four pendant environ 30 minutes.
12. Vérifier la cuisson : Si la lame d’un couteau plantée au centre ressort sèche, le gâteau est cuit.
13. Sortir le gâteau du four et laisser refroidir.

Fin

Cet exemple de recette met en lumière des aspects essentiels qui définissent la nature même d’un algorithme. Pour être applicable, une telle procédure doit se composer d’une suite d’instructions qui s’achève nécessairement ; elle ne peut s’étendre à l’infini si l’on souhaite obtenir un résultat. L'**ordre** d’exécution de ces instructions est également primordial : intervertir des étapes (comme

cuire la pâte avant de l'avoir mélangée) conduit inévitablement à l'échec. Enfin, ces actions n'ont de sens que parce qu'elles s'appliquent à des **objets clairement définis** dans un univers précis (les ingrédients de cuisine). Tenter d'exécuter ces mêmes actions sur des objets totalement différents (par exemple, remplacer la farine par du ciment) rendrait la procédure incohérente et le résultat inutilisable.

Chapitre 5. Définitions de l’Algorithme

Il existe plusieurs façons de définir ce concept central :

5.1. Définition Générale :

Un algorithme est une suite finie et ordonnée d’instructions, précises et non ambiguës, permettant de résoudre un problème donné. C’est une description logique de la manière de parvenir à un résultat en partant de données initiales.

5.2. Définition Computationnelle :

C’est une spécification abstraite de la méthode de résolution d’un problème, conçue pour être exécutée par un agent (humain ou machine). Elle décrit le traitement exact à appliquer aux données d’entrée pour produire les données de sortie souhaitées.

Le terme lui-même est une déformation du nom du mathématicien perse du IXe siècle, Muhammad Ibn Mūsā al-Khwārizmī. Il désignait à l’origine les méthodes systématiques qu’il a décrites pour effectuer les calculs arithmétiques (notamment avec les chiffres arabes) et résoudre les équations.

Chapitre 6. Propriétés d’un Algorithme

Pour qu’une suite d’instructions soit considérée comme un algorithme valide, elle doit impérativement respecter les propriétés suivantes (souvent attribuées à Donald Knuth) :

- La Finitude (ou Terminaison) : Un algorithme doit toujours se terminer après un nombre fini d’étapes, quel que soit l’ensemble des données fournies en entrée.
- La Précision (Non-ambiguïté) : Chaque étape de l’algorithme doit être définie de manière rigoureuse et sans la moindre ambiguïté. L’action à exécuter doit être parfaitement compréhensible et ne laisser aucune place à l’interprétation.
- Les Entrées (Inputs) : Un algorithme opère sur des objets. Il doit posséder zéro, une ou plusieurs données en entrée, qui sont fournies avant le début de son exécution. C’est l’univers des objets sur lesquels il travaille.
- Les Sorties (Outputs) : Un algorithme doit produire au moins un résultat (une sortie). Ces sorties sont la solution au problème posé et ont une relation spécifiée avec les entrées.
- L’Efficacité (Effectiveness) : Chaque instruction doit être suffisamment élémentaire pour être réalisable en pratique. En théorie, chaque étape doit pouvoir être exécutée par un humain en un temps fini, armé uniquement d’un crayon et de papier.

Chapitre 7. Langage Algorithmique

Bien que l'exemple de la recette de cuisine illustre l'idée d'un algorithme, il met aussi en évidence les limites du langage naturel (comme le français). Le langage naturel est intrinsèquement sujet à l'ambiguïté : une instruction peut être imprécise, incomplète ou interprétée de différentes manières. Pour garantir la propriété de précision (non-ambiguïté) indispensable à un algorithme, nous avons besoin d'un outil plus formel. C'est pourquoi nous introduisons un Langage Algorithmique, souvent désigné sous le terme de pseudo-code. Ce langage se situe à mi-chemin entre le langage naturel et un langage de programmation ; il utilise une syntaxe et des mots-clés rigoureux pour décrire les actions sans ambiguïté, tout en restant suffisamment abstrait pour être universel et indépendant de toute machine ou langage d'implémentation.

7.1. Définition

Ainsi, le Langage Algorithmique (LA) Le Langage Algorithmique (ou pseudo-code) est un ensemble de conventions d'écriture, s'apparentant à un langage de programmation simplifié et semi-formel, utilisé pour décrire un algorithme de manière claire, structurée et non ambiguë.

Il n'est pas exécutable directement par une machine ; son unique objectif est de servir d'outil de conception et de communication. Il permet au concepteur de se concentrer exclusivement sur la logique de résolution (les étapes, les structures de contrôle, les données) sans se soucier des contraintes syntaxiques spécifiques d'un langage de programmation particulier.

Pour utiliser le langage algorithmique (pseudo-code), nous devons maîtriser ses composants de base. Ils définissent la structure et les actions possibles.

7.2. Mot-clé

Un mot-clé est un terme **réservé** du langage algorithmique. Il possède une signification sémantique fixe et ne peut pas être utilisé à d'autres fins (comme nom de variable, par exemple). Ces mots-clés (`Début`, `Fin`, `Si`, `Alors`, `TantQue`, `Pour`, `Variable`, etc.) constituent la grammaire de notre langage et servent à définir la structure de l'algorithme.

7.3. Déclaration

La déclaration est l'acte par lequel nous annonçons les objets (données) que l'algorithme va manipuler. C'est une étape obligatoire dans la partie "déclarative" de l'algorithme, avant le début des actions. Déclarer un objet consiste à lui donner un **nom** (un identifiant) et à préciser son **type** (Entier, Réel, Chaîne, etc.).

7.4. Instruction

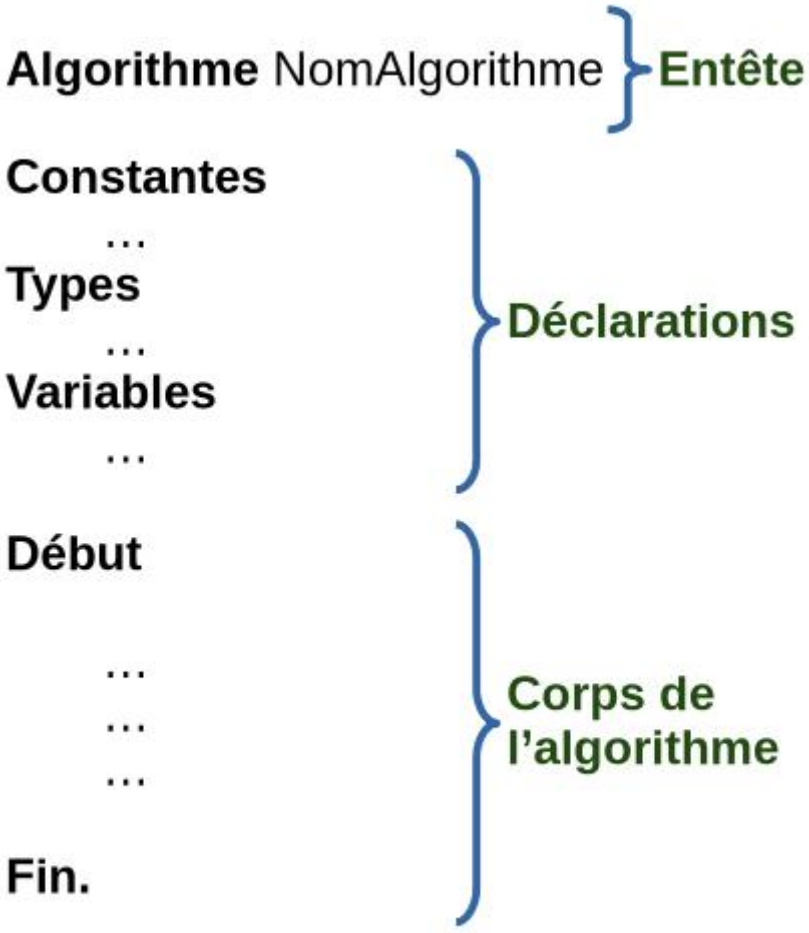
Une instruction est un ordre d'action élémentaire que l'algorithme doit exécuter. C'est la phrase d'action de l'algorithme. Les instructions se trouvent dans le corps de l'algorithme (entre `Début` et `Fin`) et opèrent sur les objets qui ont été préalablement déclarés (par exemple : `Lire(X)`, `Y ← X + 5`, `Écrire("Bonjour")`).

Chapitre 8. Structure d’un algorithme

Comme l’illustre le schéma de structure ci-après, un algorithme écrit en pseudo-code respecte une organisation rigoureuse en trois parties distinctes.

Tout d’abord, **l’En-tête** sert à nommer l’algorithme ; il commence généralement par le mot-clé `Algorithme` suivi d’un identifiant qui décrit son rôle (par exemple, `CalculMoyenne`). Ensuite, la **partie Déclarative** est essentielle : elle regroupe la définition de tous les objets (variables, constantes) que l’algorithme manipulera, en précisant leur nom et leur type. Finalement, le **Corps de l’algorithme** représente la partie active du traitement. Il est toujours délimité par les mots-clés `Début` et `Fin`, et contient la séquence d’instructions (actions) qui seront exécutées dans l’ordre pour résoudre le problème posé.

Structure d’un Algorithme



La partie déclarative est fondamentale car elle sert à "enregistrer" tous les objets de données que l’algorithme aura le droit de manipuler. Chaque objet (constante ou variable) doit être identifié par un nom unique appelé **identifiant**. Le choix de cet identifiant est crucial : il doit être descriptif et respecter des règles syntaxiques strictes. En particulier, un identifiant **ne doit jamais commencer par un chiffre** et **ne peut pas contenir de caractères spéciaux** (tels que `+`, `*`, `&`, `?`), à la seule exception notable du caractère de soulignement (`_`).

On distingue deux types de déclarations d'objets :

1. **Les Constantes :** Une constante est un objet dont la valeur est fixée au moment de la déclaration et ne peut **jamais** être modifiée durant l'exécution de l'algorithme. On la déclare en utilisant le symbole `=` pour lui assigner sa valeur définitive. Par exemple :
    - `PI = 3.14159`
    - `TVA = 20.0`
    - `Message_Erreur = "Le compte est bloqué"`
2. **Les Variables :** Une variable est un "conteneur" dont la valeur est susceptible de changer au cours de l'exécution (via une lecture ou une affectation). La déclaration d'une variable consiste à associer son identifiant à son type de données (ce qu'elle peut contenir), en utilisant le symbole `:`.
    - `Age_Utilisateur : Entier`
    - `Moyenne_Generale : Réel`
    - `Nom_Etudiant : Chaîne de caractères`