# Algorithmique et JavaScript – Introduction à l'algorithmique

## L'algorithmique

## Définition et présentation

Les algorithmes sont partout. Derrière chaque application, chaque programme, chaque sous-programme. Que ceux-ci soient utilisés en informatique, telle qu’une application ou un OS tournant sur votre PC, ou dans d’autres domaines tels que la téléphonie, l’automobile ou encore l’électroménager. Tous les appareils intégrant de l’électronique ont des programmes intégrés (du code) nous permettant de les utiliser. Des algorithmes ont été écrits afin de concevoir ces programmes.

Un exemple d'algorithmes que l'on utilise tous les jours: ceux du moteur de recherche Google. Derrière ces algorithmes se cachent des instructions indiquant aux robots de la société Google de rechercher les informations sur les serveurs adéquats afin de vous renvoyer et classer dans un certain ordre le résultat de votre requête mais aussi en affichant les publicités en fonction de votre profil (e-marketing).

***Il n y a pas de programmes informatiques sans algorithmes.***

Alors au juste c’est quoi un algorithme ? La définition la plus simple est la suivante :

***Un algorithme est un ensemble d’instructions servant à résoudre un problème*.**

C’est donc une solution pensée par un informaticien lui permettant d’écrire plus facilement son programme dans un langage informatique.

En effet, lorsque notre programme doit résoudre un problème complexe, il est hors de question de commencer à programmer sans avoir au préalable pensé à la solution. Cette solution doit être **obligatoirement** transcrite sous forme d’un algorithme de façon à être certain que notre programme fonctionnera correctement pour tous les cas de figures. Un algorithme a trois buts principaux :

* Eviter que notre programme ne contienne trop de bugs ; un bug est une e*rreur de programmation due au développeur conduisant à un mauvais fonctionnement du logiciel.*
* De gagner du temps lors de l’écriture du programme.
* De permettre à un autre programmeur de pouvoir comprendre comment a été écrit le programme (avec quelle logique) et quelles ont été les solutions choisies.

Je peux vous assurer que si vous commencez à programmer sans avoir réfléchi à la solution au préalable, vous allez très certainement suer de grosses gouttes lors de l’écriture de votre programme. Et si malgré tout avec de la chance, vous y arrivez quand même, votre temps de codage aura été 2 ou 3 fois plus long que celui d’une personne ayant utilisé un algorithme.

Dans le cadre de ce cours nous allons donc apprendre à écrire des algorithmes avant de coder nos programmes. **En d’autres termes, vous allez devoir d’abord mettre votre solution sur papier avant d’utiliser votre ordinateur.**

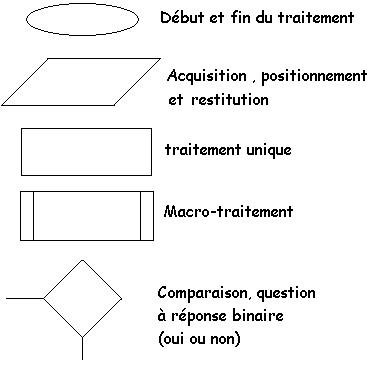
## Les types d’algorithme

Contrairement aux langages informatiques qui disposent d’une syntaxe bien spécifique et standardisée, les algorithmes ne répondent à aucun standard en particulier. On peut donc écrire les algorithmes grâce à différentes méthodes. Ces méthodes ont leurs avantages et leurs inconvénients.

Les 2 principaux types d’algorithme sont les organigrammes et les pseudo-codes.

### *Les organigrammes*

Les organigrammes sont des algorithmes que l’on représente sous forme graphique grâce à des symboles particuliers. Chaque symbole représente un type d’action que le programme doit réaliser.



Prenons un exemple : Soit le programme qui doit faire l’addition de 2 nombres et qui doit afficher le résultat à l’écran.

Notre algorithme ressemblerait à peu de chose près à ceci :

Début

Var somme, nbre1, nbre2

nbre1 et nbre2

Somme = nbre1 + nbre2

Introduire nbre1 et nbre2

Sortir somme

Fin

Comme vous pouvez le constater, écrire ce type d’algorithme est lourd et fastidieux même si il existe des logiciels spécialisés qui peuvent nous y aider (logiciel *Larp* – en téléchargement gratuit ou en Word dans Insertion  Image  Formes automatiques).

Le souci, c’est que pour des programmes complexes, un algorithme sous forme d’organigramme peut très vite prendre beaucoup de place sur le papier. De plus, il est difficile en cas d’erreur d’apporter des modifications en insérant un nouveau symbole au milieu de l’algorithme. Par contre l’avantage de cette méthode, très peu utilisée aujourd’hui dans le milieu professionnel, est qu’elle est visuelle et plus adaptée à certaines personnes. **Dans le cadre de ce cours nous n’utiliserons pas ce genre d’algorithme**.

### *Le pseudo-code*

Le pseudo code est une sorte de langage à mi-chemin entre le langage informatique et le Français. Cela nous permet de retranscrire les différentes instructions que contiendra notre programme en termes plus proches de notre langage naturel. Il n’y a pas de normes standardisées du pseudo-code, mais il existe quelques conventions permettant d’écrire en pseudo-code. Dans le cadre de ce cours nous utiliserons une de ces conventions. Ce qui importe c’est que chacun d’entres vous utilisent la même notation. C’est comme dans une entreprise, il faut se mettre d’accord sur la syntaxe à employer afin d’éviter d’avoir dans la documentation des pseudo-codes différents, ce qui poseraient des problèmes de communication au sein même de l’entreprise.

Il est important de souligner que le pseudo-code ne fait référence à aucun langage en particulier. Un algorithme pourrait donc être utilisé par des programmeurs pour être codé dans n’importe quel langage.

Le pseudo-code est aussi appelé **Langage de Description Algorithmique** ou **LDA**