

UN ORDINATEUR

Vous avez tous déjà manipulé des ordinateurs, téléphones mobiles dits *intelligents*... mais savez-vous vraiment comment ils sont structurés et le principe de leur fonctionnement ?

Ce premier chapitre est à caractère essentiellement culturel mais cela ne veut pas dire qu'il est dénué d'intérêt pratique.

1 LES ORIGINES

La première calculatrice

En 1642, Blaise Pascal crée une machine d'arithmétique que l'on appellera plus tard *la pascaline*. Le but de cette machine est alors de simplifier le travail comptable de son père qui est super-intendant de Haute-Normandie : elle permet d'additionner, soustraire, multiplier et diviser facilement des entiers.

Avec cette machine, Pascal offre le premier traitement mécanique d'une information¹.

On y trouve déjà les ingrédients essentiels de l'informatique :

- des données d'entrée : les nombres à manipuler,
- un processus (ici programmé mécaniquement avec les engrenages),
- une sortie : le résultat de l'opération.

Où l'on parle de programmation

En 1837, Charles Babbage invente la première machine à calculer programmable. Pour cela, il s'inspire des métiers à tisser Jacquard² dont on pouvait « programmer » le motif du tissage grâce à des cartes perforées.

Les ordinateurs... et les premiers bugs

Vers le milieu des années 1930 apparaissent les premiers ordinateurs. D'abord électro-mécaniques, il s'affranchissent peu à peu des composants mécaniques pour intégrer les lampes à vide puis l'électronique à partir de 1947.

La première erreur informatique est apparue sur un ordinateur à lampe³. Un insecte attiré par la chaleur est venu se loger dans les lampes et a créé un court circuit. Il a donné son nom au bug.

L'apparition des transistors et des circuits intégrés permet de réduire considérablement la taille des appareils et leur consommation en énergie : on entre dans l'aire des *micro-ordinateurs*. En 1971, le premier micro-ordinateur Kenbach 1 a une mémoire de 256 octets ! Le premier « PC » (*personal computer*) est commercialisé par IBM en 1981.

2 UN ESSAI DE DÉFINITION

On peut définir un **ordinateur** comme un dispositif électronique capable de traiter l'information.

*Le terme français **ordinateur** a été introduit en 1955 par un responsable du service publicité d'IBM, François Girard. Il s'éloigne de la simple traduction de « computer » (un calculateur) et avance l'idée d'une gestion ordonnée de l'information.*

Le principe de fonctionnement d'un ordinateur (ou de tout autre appareil informatique : smartphone, appareil photo numérique, tablette...) est basée sur la figure ci-dessous.



Remarque : On pourrait définir un processus un peu plus général, où l'entrée correspond à un *état* initial, et la sortie à un état final. Le traitement correspond ainsi à un changement d'état et il n'y a donc plus nécessairement de sortie explicite.

¹L'allemand Wilhelm Schickard avait déjà posé les plans d'une machine semblable quelques années plus tôt, mais ce projet est resté au stade du prototype contrairement à la pascaline qui fut construite en plusieurs exemplaires.

²Les premiers métiers Jacquard sont apparus en 1810.

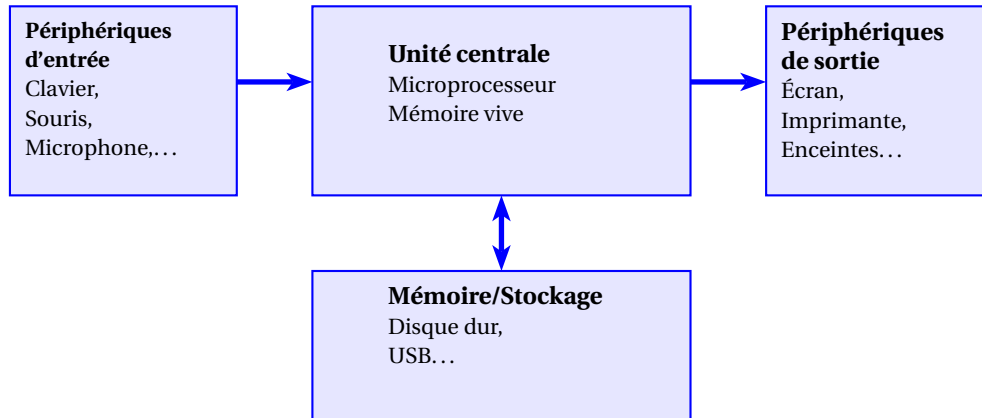
³On peut trouver une explication très simple du fonctionnement d'un tube à vide (ou lampe) sur le site <http://anjuanje.free.fr/OldHeaven/OldHeavenTubes.htm>

3 COMPOSITION D'UN ORDINATEUR

L'ordinateur est composé d'une partie matérielle (électronique) : le **hardware** ; dotée de logiciels : le **software**.

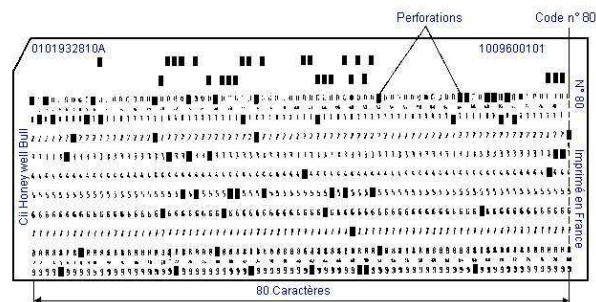
A Le hardware

La structure du hardware peut être résumée par la figure suivante :



Faisons à présent le tour de ces composants :

- Les **périphériques d'entrée et de sortie** permettent la communication avec l'unité centrale. Cela s'opère via des **unités d'échange** présentes dans l'unité centrale. L'utilisateur transmet ses instructions à l'ordinateur par l'intermédiaire des périphériques d'entrée, et l'ordinateur lui renvoie le résultat par les périphériques de sortie.
Sur les premiers ordinateurs (et jusque vers 1970) les périphériques étaient très rudimentaires et se limitaient à des cartes perforées. En fonction du placement des trous, l'ordinateur comprenait l'information. Il renvoyait ensuite le résultat par carte perforée ou impression. Cette méthode était lourde, fastidieuse et sujette à de très nombreuses erreurs humaines.



Carte perforée

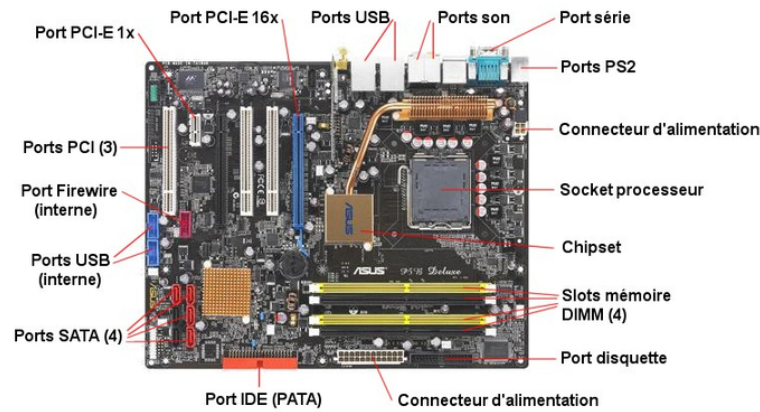
- La **mémoire de masse** assure le stockage des données. Elle correspond au disque dur.
La mémoire n'est pas apparue dès les premiers ordinateurs, mais le besoin s'est rapidement fait sentir de pouvoir « sauvegarder des programmes en interne pour des utilisations successives.
La majorité des disques durs sont composés de disques recouverts d'une couche magnétique dont le champ magnétique local code des données (0 ou 1).
La mémoire flash est plus récente, elle est utilisée sur les clés USB. Basée sur la technologie des semi-conducteurs, elle permet un accès plus rapide à toutes les données.
La baisse des coûts des mémoires flash tendent à généraliser son usage à travers les disques SSD de grande capacité. Cela accélère considérablement les accès mémoire pour rendre l'ordinateur beaucoup plus réactif.

La mémoire de masse est à distinguer de la mémoire vive (RAM) dont il sera question plus loin. La mémoire de masse (ou de stockage) a vocation à conserver les informations sur le long terme, contrairement à la mémoire vive d'accès beaucoup plus rapide, qui sert essentiellement aux transferts liés aux processus actifs.

- Le cœur de l'ordinateur est composé de la **carte mère**. Elle comporte certains composants intégrés et d'autres qui lui sont rapportés.

- Le **chipset** est le circuit électronique de la carte qui assure les communications entre les différents organes de l'ordinateur⁴. Le chipset limite les possibilités d'évolution de l'ordinateur en fonction des informations qu'il peut faire transiter (et des connecteurs). Ainsi, il est inutile de brancher un composant trop récent sur une ancienne carte mère car elle ne saura pas l'utiliser.
- Le **CMOS** (*complémentary metal-oxide semiconductor*) est une mémoire qui contient quelques données essentielles au fonctionnement de l'ordinateur, comme les secteurs disque de démarrage, les configurations matérielles...
 Cette mémoire est une cible privilégiée des virus car elle reste tout le temps allumée.
 La **pile du CMOS** est une pile plate (ou accumulateur) qui sert à alimenter le CMOS même lorsque l'ordinateur est hors tension. C'est elle qui permet à l'ordinateur d'être à l'heure au démarrage.
- L'**horloge temps réel** (RTC) est chargée de synchroniser les échanges de données. Elle est alimentée par la pile du CMOS.
- Le **BIOS** (*Basic Input/Output System*) est un programme qui communique avec le CMOS (accessible via le BIOS setup) et permet de lancer le système d'exploitation d'un ordinateur. Le BIOS est situé sur la mémoire morte **ROM** (*read-only memory* : originellement, elle ne pouvait pas être modifiée).
- Le **processeur** (ou microprocesseur) exécute les instructions des programmes. Son cadencement définit le nombre d'opérations par seconde qu'il peut exécuter, il est donné par sa fréquence en *MHz* ou *GHz*. Un processeur à 1 *GHz* permet de traiter un milliard d'informations par seconde.
L'overclocking est une accélération de la vitesse de l'horloge au delà de sa fréquence nominale afin d'augmenter les performances. Mais c'est une technique qui n'est pas sans risque (surchauffe du processeur, erreurs de calcul, instabilité de l'OS...)
 La tendance est à la multiplication des processeurs, ou au multi-cœurs. Les architectures parallèles permettent ainsi d'effectuer plusieurs tâches en parallèle et non à la suite les unes des autres.
 Le processeur ne fait pas vraiment partie de la carte mère, mais il se branche dessus via le connecteur **SOCKET**. En fonction des gammes de processeurs, les **SOCKET** ne seront pas les mêmes. Le processeur est un des éléments les plus coûteux de l'ordinateur.
- La **mémoire vive** ou **RAM** (*Random access memory*) est une mémoire d'accès très rapide pour conserver les informations d'usage immédiat. Contrairement à la mémoire de masse, la RAM est entièrement vidée lorsque l'ordinateur est mis hors tension.
Si un ordinateur ne possède pas suffisamment de RAM, alors il utilisera la mémoire de masse (disque dur) pour conserver des résultats intermédiaires. Il fera donc de nombreuses lectures/écritures sur le disque dur qui est beaucoup plus long d'accès que la RAM (environ 100 000 fois plus long pour un disque dur). Cela ralentira considérablement les processus, au point parfois, de rendre l'ordinateur inutilisable.
- Un **dissipateur thermique** est associé à la carte mère pour dissiper la chaleur (produite par effet Joule) et éviter de faire fondre les composants.
Les data centers, calculateurs, ou simplement les salles équipées de nombreux ordinateurs sont climatisées pour contrer l'apport thermique dû à l'effet Joule. Les consommations électriques des grands calculateurs et des data centers sont en effet très importantes et représentent un gros enjeu actuel. D'après l'ADEME, le secteur numérique représenterait 10% de la consommation électrique mondiale. La consommation des seuls data centers était estimé à 4% de la consommation mondiale en 2015. Le supercalculateur Pangea III acquis par l'entreprise Total en 2019, et à cette date, le plus puissant calculateur industriel au monde, consomme autant qu'une ville de 2500 habitants (1,5MW). Son prédécesseur, Pangea II, consommait trois fois plus.

⁴En général, il contient deux puces, le pont *northbridge* qui gère les communications avec le processeur et les accès mémoire, et le pont *southbridge* qui communique avec les périphériques (entrées/sorties). Aujourd'hui, le chipset intègre souvent une puce graphique et une puce audio qui peuvent remplacer la carte graphique et la carte son (mais de qualité moindre).



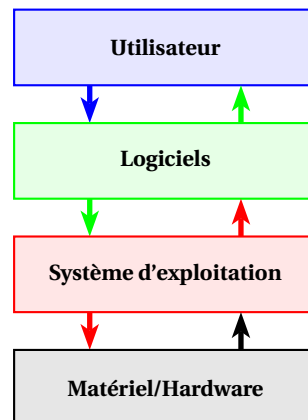
Description d'une carte mère (source : choixpc.com)

B Le software

Le **système d'exploitation** (*OS : Operating System*) est un ensemble de programmes qui fait le lien entre les logiciels (ou les programmes que vous rédigez) et les éléments matériels de votre ordinateur.

Les plus connus sont actuellement MS-Windows, iOS, Linux pour les ordinateurs personnels, Android et iOS pour les téléphones mobiles.

L'OS permet l'échange entre le processeur, la mémoire et les systèmes applicatifs (voir la figure).



Rôle du système d'exploitation

Il est démarré directement à partir d'un secteur d'amorçage après le BIOS.

Son rôle est de :

- gérer le processeur, en allouant ses ressources aux différentes applications (il peut gérer les priorités, faire fonctionner des tâches en parallèle...),
- gérer les accès mémoire (RAM et ROM) et le système de fichiers,
- gérer les droits d'accès, lecture, écriture, exécution,
- servir d'interface à la manipulation des périphériques (évite au programmeur de donner des instructions qui seraient propres à chaque périphérique). Pour cela il fait appel aux *pilotes* (pour traduire les instructions en fonction du matériel),
- Détecte et récupère les erreurs et si besoin les signale à l'utilisateur,
- ...

Il est composé

- du **noyau** (kernel) qui contient les fonctions fondamentales nécessaires au bon fonctionnement de l'ordinateur.

- du **shell** (coquille) pour communiquer directement avec le noyau en lignes de commande.
- du **système de fichiers** (*FS : File system*) qui organise les fichiers en arborescence.

Un ordinateur peut fonctionner avec différents systèmes d'exploitation. Il faut simplement que l'OS soit programmé pour savoir contrôler l'architecture matérielle et ses composants. Il est par exemple très simple d'installer un système d'exploitation Linux sur une machine Windows.

L'**arborescence fichier** qui est gérée par l'OS permet de situer physiquement le contenu des fichiers sur la mémoire. Dans la mesure du possible, l'OS veillera à placer un même fichier sur des zones mémoire contiguës. En effet, chaque déplacement sur le disque prend du temps et un fichier morcelé sur de multiples emplacements sera beaucoup plus long à lire. Cette ambition reste néanmoins utopique car les modifications de fichiers, suppressions... ont vite fait de « mettre le bazar » sur le disque. Un disque où les fichiers sont « éparpillés en petits morceaux » est dit *fragmenté*. De plus en plus, les systèmes d'exploitation intègrent des processus qui défragmentent le disque au fur et à mesure de son utilisation⁵.

Comme chaque système gère les fichiers à sa façon : un disque n'est pas structuré de la même manière selon l'OS (et son utilisation).

La méthode de gestion des fichiers est défini par un *format*⁶. Pour changer le format d'un support mémoire, il faut le *formater*.

Voici quelques uns des formats les plus courants :

- NTFS pour Windows, et récemment ReFS,
- HFS+ pour Mac,
- ext4 pour Linux,
- FAT32 développé par Microsoft et souvent utilisé pour les échanges (clés USB). C'était le format utilisé depuis Windows 95 jusqu'à Windows Millénium. Il est progressivement remplacé par extFAT qui possède des performances supérieures.
- ...

Ainsi, par exemple, un disque formaté en NTFS possède une table MFT qui contient toutes les métadonnées des fichiers (nom, date, emplacement, droits d'accès...). Sur un ordinateur, cette table peut occuper plusieurs Go d'espace disque. Les hackers peuvent chercher à modifier cette table pour faire virtuellement *disparaître* de volumineux fichiers placés sur le disque à notre insu : on ne les voit plus, mais ils sont toujours présents sur le disque.

Permissions d'accès aux fichiers et dossiers

Chaque fichier (ou dossier) est affecté de certaines permissions pour définir qui a le droit de le lire, de le modifier ou de l'exécuter. En général, vous n'avez pas besoin de vous préoccuper de la gestion des droits car le système d'exploitation les gère pour vous. En revanche, si vous êtes amené à mettre des ressources en ligne, la gestion des droits d'accès deviendra une préoccupation essentielle pour la sécurité de votre site et de ses données.

Pour un fichier (ou un dossier), les différents droits sont :

- la lecture (**r** *read*) pour lire le contenu du fichier,
- l'écriture (**w** *write*) pour écrire ou modifier le fichier,
- l'exécution (**x**) pour exécuter le fichier (par exemple un script ou un logiciel).

Attribuer des droits à un fichier, c'est associer à chaque catégorie d'utilisateurs certains droits (lecture, écriture, exécution).

Les catégories d'utilisateur sont :

- le propriétaire du fichier désigné par la lettre **u** (*user*),
- le groupe du propriétaire désigné par la lettre **g** (*group*),

⁵En raison de ces évolutions, la défragmentation manuelle n'est plus accessible directement depuis Windows 8, bien qu'elle soit encore possible par le shell.

⁶Ces formats ne sont pas liés au système physique qui est commun à tous les OS. Ainsi un même disque dur peut-être formaté dans différents formats par des logiciels spécifiques.

- les autres désignés par la lettre **o** (*other*).

On peut indiquer les permissions d'un fichier en mettant à la suite les droits de chaque catégorie. Par exemple si le fichier `file` est affecté des permissions `rwrx-x--x`, cela veut dire que le propriétaire a tous les droits (lecture, écriture et exécution), que le groupe peut lire et exécuter le fichier et que les autres ne peuvent que l'exécuter.

Les permissions peuvent aussi être traduites numériquement. À chaque droit correspond un nombre :

- 4 pour la lecture
- 2 pour l'écriture
- 1 pour exécution

Les différents droits sont additionnés pour donner le code. Avec l'exemple précédent, le propriétaire a tous les droits : $4 + 2 + 1 = 7$, le groupe peut lire et exécuter : $4 + 1 = 5$, et les autres ne peuvent qu'exécuter : 1.

Les droits du fichier s'écrivent alors `451`⁷.

En général, il existe aussi un super utilisateur, ou administrateur, qui a tous les droits. Il est désigné par `sudo` ou `root`. Les systèmes d'exploitation récents évitent de donner les droits du super utilisateur sur une session normale pour limiter les risques d'endommager le système.

4 PROGRAMMATION ET ALGORITHMIQUE

A Qu'est-ce qu'un langage de programmation

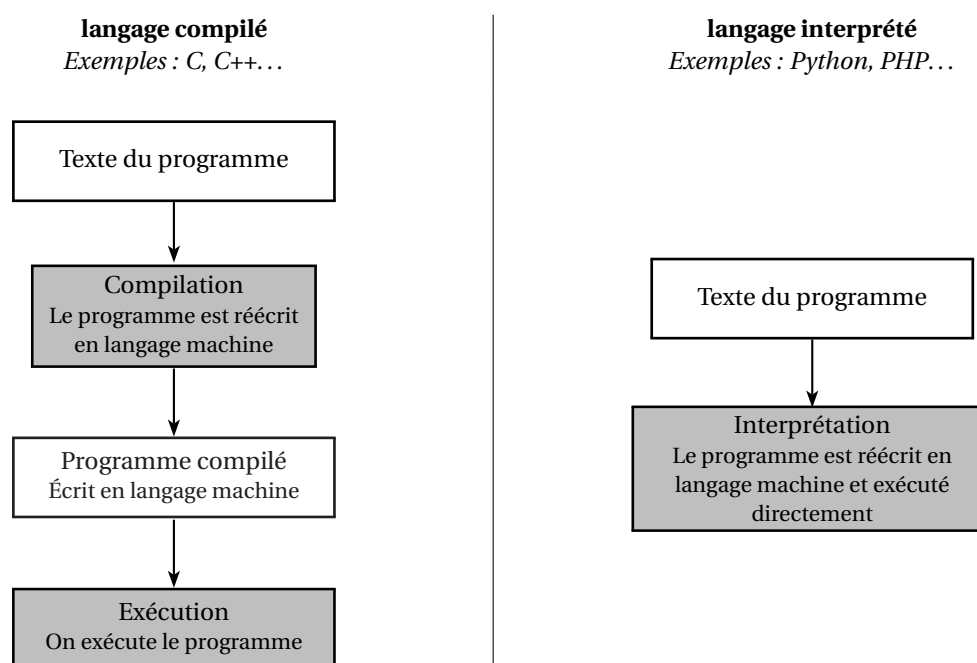
Les langages de programmation sont au fondement de toute l'informatique. Ils servent à communiquer avec la machine pour lui donner une suite d'instructions à réaliser (faire un calcul, afficher quelque chose...).

Ce sont des langues compréhensibles par l'humain et qui peuvent être ensuite traduites en langage machine.

Il existe des langages de plus ou moins haut niveau. Un langage de haut niveau permet de faire abstraction de beaucoup de problématiques purement informatiques (gestion de la mémoire...) pour se concentrer sur les instructions.

Python fait partie des langages de haut niveau.

Il existe deux⁸ types de langages, les langages compilés et les langages interprétés (voir figure).



Langage compilé - langage interprété

⁷ Cela correspond à une écriture en base 8. On peut vérifier que deux types de droits différents donnent des valeurs différentes.

⁸ En fait, cela dépend de l'implémentation du langage plus que du langage lui-même et un langage de type interprété pourrait aussi être implémenté pour fonctionner avec un compilateur. Néanmoins, il existe un usage majoritaire pour chaque langage ce qui explique la dénomination.

En réalité, de plus en plus de langages dits interprétés sont des **langages compilés en langage intermédiaire bytecode**. Cela signifie qu'au moment de l'interprétation, une compilation a lieu en bytecode. Le bytecode est une forme de langage intermédiaire entre le code source et le langage machine.

C'est le cas pour Python, et le bytecode nécessite Python pour être interprété mais est plus rapide à exécuter que le code source. L'étape de compilation n'est pas visible pour l'utilisateur, elle produit le fichier d'extension `.pyc`.

Le fichier que l'on exécute pour lancer un logiciel comme Word ou Firefox contient un programme compilé. Si on ouvre le fichier (fichier.exe sous Windows), on voit qu'il n'est pas compréhensible : on peut simplement l'exécuter. Ceux qui programment le logiciel possèdent le texte du programme, mais il vendent uniquement le programme compilé.

Quelques ordres de grandeur⁹ :

- un projet informatique en BCPST2 contient entre 100 et 500 lignes de code environ,
- un *pacemaker* : de l'ordre de 100 000 lignes de code,
- la partie logicielle avionique du Boeing 787 : environ 7 millions de lignes (on double si on considère les autres logiciels),
- Windows XP : 40 millions de lignes, iOS Tiger : 80 millions de lignes,
- Un logiciel embarqué de voiture : 100 millions de lignes...

Pas étonnant qu'il y ait des erreurs et des failles !

Il existe plus de 7000 langages de programmation différents, mais peu sont effectivement utilisés. Le fonctionnement de chacun est spécifique et répond à des objectifs particuliers. Un programmeur connaît toujours plusieurs langages et utilise l'un ou l'autre en fonction de ce qu'il veut faire.

HTML qui « code » les pages web n'est pas un langage de programmation. Il ne permet de donner aucune instruction à l'ordinateur. C'est juste une convention pour dire que tel élément est un titre, tel autre est un paragraphe... Chaque navigateur lira ce « code » et choisira comment l'afficher. Mais il n'est pas possible de faire faire des choses intelligentes à HTML. C'est simplement du texte mis en forme suivant certaines conventions.

Par contre, il existe PHP qui est un langage de programmation pour le web. Lorsqu'on affiche une page web produite par un tel site, on ne reçoit pas de PHP parce qu'il a déjà été exécuté par le serveur et celui-ci n'envoie que le résultat de son calcul : une page HTML toute simple (ou presque).

Un langage de programmation s'apprend un peu comme une langue étrangère : on apprend la syntaxe et la grammaire qui permettent de communiquer avec la machine.

Pour écrire un programme, il faut être très précis et rigoureux. En effet, la moindre faute de grammaire provoquera un résultat faux ou une erreur.

B Algorithmique

La forme et le fond

Il ne suffit pas d'apprendre un langage, il faut aussi savoir ce que l'on veut dire avec ce langage. Que veut-on que fasse l'ordinateur ? C'est l'objet de l'**algorithmique**.

Le langage concerne la forme¹⁰
L'algorithmique concerne le contenu.

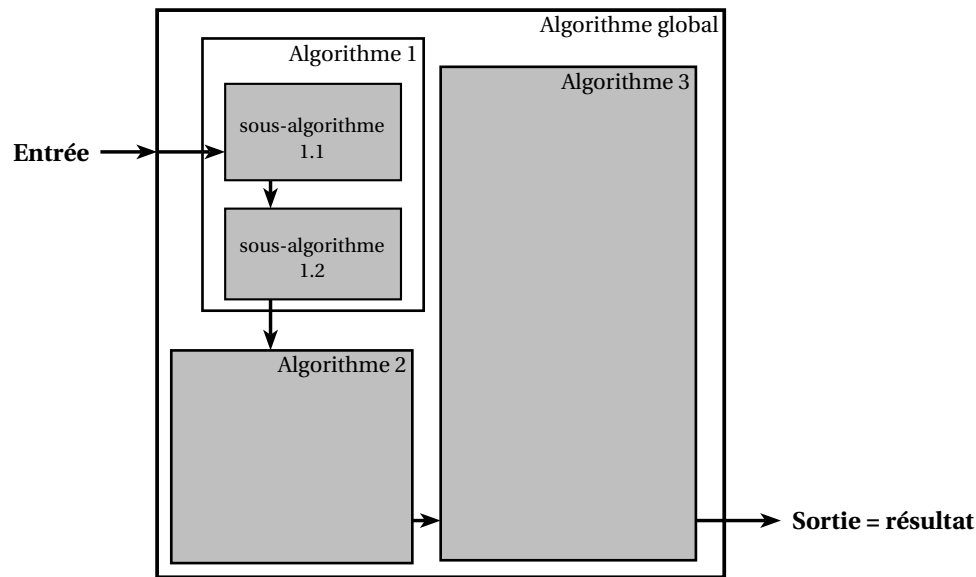
Par exemple, un même article de journal peut être écrit en plusieurs langues différentes, mais le contenu est le même. L'algorithmique sert à définir « le contenu du journal ». On pourra donc ensuite l'appliquer avec n'importe quel langage de programmation, et pas seulement avec Python.

L'architecture du code

On comprend aisément que lorsqu'on réalise de gros programmes composés de plusieurs milliers, voire millions de lignes, il n'est pas possible de mettre un programmeur qui commence à la première ligne et qui avance jusqu'à la dernière ligne comme s'il écrivait un livre. C'est la raison pour laquelle les programmes sont toujours écrits par « **blocs** » que l'on imbrique les uns dans les autres, ou que l'on accole... comme dans un jeu de construction. Chaque programmeur est alors responsable du codage de certains blocs.

¹⁰En réalité, comme pour une langue, la forme va influencer le contenu. C'est la raison pour laquelle il existe une multitude de langages de programmation différents.

Les algorithmes sont donc des structures naturellement **récur­sives**. Cela veut dire que chaque algorithme¹¹ peut être décomposé en plus petits algorithmes (qui peuvent eux-mêmes être décomposés à nouveau, et ainsi de suite). Réciproquement, chaque algorithme peut ensuite servir de bloc pour un algorithme « plus gros » qui l'englobe.



Les zones grisées peuvent être décomposées à nouveau
Schéma "digestif" d'un algorithme

Dans la réalité, ce schéma peut être implémenté de deux façons selon le type de langage.

- le langage **fonctionnel** crée les sous-algorithmes qu'il réutilise un maximum de fois dans des « méta-algorithmes »... C'est ce que nous avons proposé ici (et sans doute le plus simple). On part des briques élémentaires pour construire l'édifice.
- le langage orienté **objet** possède une logique plus algébrique. On crée des objets à l'intérieur de classes. Chaque objet possède des attributs et des méthodes. On part de l'objet le plus général possible que l'on spécifie ensuite par un système d'héritage en fonction de l'utilisation voulue. La logique est donc un peu inversée, puisque l'on part du plus général pour aller vers le particulier. Ce type de programmation est privilégié aujourd'hui, même si elle n'est pas enseignée en classes préparatoires.

Python permet de programmer suivant les deux méthodes.

Par la suite, chacun de ces blocs fonctionnera un peu comme une boîte qui « transforme » des données d'entrée en données de sortie.

On définira alors un algorithme de cette façon :

Un algorithme est un processus logique qui comporte une entrée, un traitement, et une sortie.

Un algorithme comporte un nombre fini d'instructions et **se termine en un temps fini**.

Remarque : On pourrait remplacer le terme « logique » par « déterministe » dans la définition. Cela signifie que l'algorithme n'a pas de libre arbitre¹² : il n'improvise pas. Quand on lui donne les mêmes données à l'entrée, il donne nécessairement le même résultat en sortie : il n'y a pas d'aléatoire¹³.

Comme spécifié plus haut, on pourrait parler d'état initial et d'état final au lieu d'entrée et de sortie.

Lorsqu'on veut écrire un programme, le plus important est de définir au préalable quelles seront les « briques », ce qu'elles recevront en entrée et le résultat qu'elles doivent donner en sortie. Lorsque ce travail est fait, il ne suffit plus que d'écrire les algorithmes pour chaque brique.

Avant d'écrire un algorithme, Il faut toujours se poser ces questions :

Quelle entrée vais-je donner à mon algorithme ?

Quel résultat doit-il me fournir ?

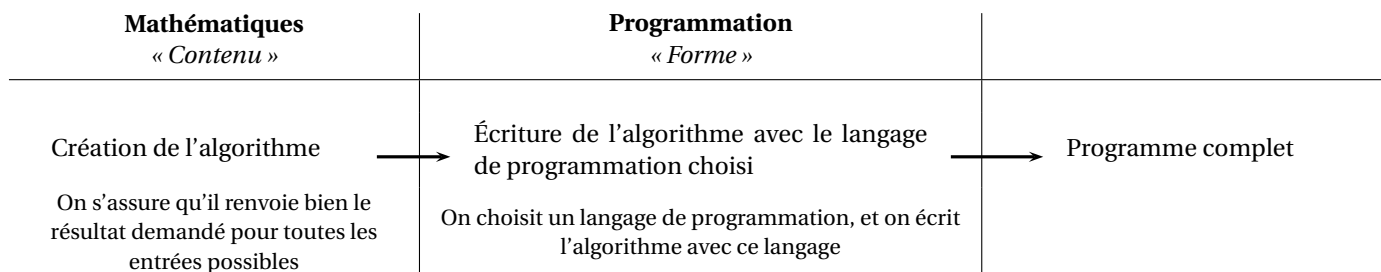
¹¹sauf un algorithme élémentaire.

¹²Il en est de même pour l'intelligence artificielle qui ne déroge pas à ce principe.

¹³Les ordinateurs ne savent pas générer d'aléatoire. Même lorsqu'on demande un nombre aléatoire à un ordinateur ou à une calculatrice, ce nombre n'est pas vraiment aléatoire, mais il est calculé de façon suffisamment complexe pour qu'il en donne l'impression. Cela peut parfois poser des problèmes lors de certaines études pointues qui demandent des nombres *vraiment* aléatoires.

C Bonne exécution d'un programme

Pour définir un algorithme, nous avons parlé de processus logique. Écrire un algorithme relève donc essentiellement du domaine mathématique dont la logique est partie intégrante. La partie « programmation », n'est plus tellement l'algorithme lui-même, mais plutôt son écriture avec le bon langage de programmation.



Cela est le signe d'une exigence : s'assurer qu'un algorithme se termine nécessairement et qu'il donne le résultat voulu dans tous les cas.

Par exemple, si un algorithme prend en entrée un nombre quelconque, il faut bien vérifier que cela marche pour tous les nombres (positifs, négatifs, nuls, entiers, décimaux...). Ainsi, avant de diviser par ce nombre, je m'assurerai qu'il n'est pas nul.

Normalement, tout algorithme doit pouvoir être prouvé (comme un théorème mathématique). C'est un exercice formel long et compliqué. Dans la réalité, les algorithmes sont rarement prouvés, car cela serait trop long à réaliser. Nous ne le ferons pas non plus.

En pratique, on remplace donc les preuves par des **tests**. Ces tests ne pourront jamais nous prouver que l'algorithme est *juste*, mais ils nous aideront à éviter de nombreuses erreurs. Cela suppose simplement de penser à faire des tests qui recouvrent le maximum de cas possibles (nombres positifs et négatifs, nuls...) et de vérifier à chaque fois que le résultat rendu est le bon : il ne suffit pas d'obtenir un résultat, il faut s'assurer que ce soit celui voulu. On commencera donc par des tests dont on peut prévoir l'issue sans utiliser l'ordinateur.

De même, dès qu'on programmez une boucle, il faudra toujours s'assurer qu'elle termine dans tous les cas et ne risque pas de tourner indéfiniment.

D Algobox

Algobox est un langage de programmation à usage pédagogique uniquement. La syntaxe (la « grammaire ») de ce langage est beaucoup trop lourde pour que l'on puisse rédiger de gros programmes avec.

L'avantage de ce langage est qu'il met bien en exergue l'idée d'un algorithme construit à partir de boîtes (box en anglais).

Dès qu'on a bien compris cette structuration des algorithmes sous forme de boîtes que l'on assemble, il est temps de passer à un langage un peu plus sérieux. C'est ce que nous faisons avec Python.

E Python

Python est un langage qui a été conçu vers 1990 par Guido van Rossum. Son nom vient de la série télévisée "Monty Python" dont le créateur était un passionné.

C'est un langage assez simple, bien documenté et donc adapté pour débiter. Mais il est également très utilisé par des programmeurs expérimentés. Si vous êtes amenés à travailler avec un langage de programmation en entreprise (dans un avenir pas trop lointain...), il y a de fortes chances que ce soit celui-ci.

Son succès, fait qu'il est enrichi de nombreuses bibliothèques (des collections d'algorithmes pour réaliser des opérations qui ne sont pas *natives* à Python : travail sur les tableaux, tracer des graphes, réaliser des dessins dynamiques...).