https://laurentbloch.org/BlogLB/22-L-informatique-tentative-d-explication

L'informatique, tentative d'explication

- Les chats ne sont pas de gauche -

Date de mise en ligne : lundi 24 octobre 2022

Chapitre précédent[/Chapitre suivant/]

L'informatique a joué un rôle important dans ma vie, elle est de ce fait très présente dans ce livre, et je dois donc au lecteur profane une explication, aussi succincte que possible mais pas trop simplificatrice quand même, de ce dont il s'agit : la voici.

L'ordinateur est un appareil pour une science, qui se nomme informatique, en cela il n'est pas une machine comme les autres. On désigne par le mot ordinateur la machine complète, munie de ses accessoires tels que clavier, écran, prise réseau, disque dur, etc. Lorsque l'on veut parler plus précisément de l'organe qui exécute les programmes, on dit le processeur, ou l'unité centrale, c'est-à-dire le cœur de l'ordinateur, réalisé aujourd'hui par un seul composant électronique d'un ou deux centimètres-carrés, le microprocesseur. Pour le dire autrement, l'unité centrale d'un ordinateur, qui il y a soixante ans pouvait peser une ou deux tonnes et était constituée de milliers de composants, se présente aujourd'hui sous la forme d'un composant unique, le microprocesseur. Aucune industrie humaine n'a connu un progrès aussi rapide soutenu pendant aussi longtemps.

Emmanuel Saint-James a pu écrire dans son livre *La Programmation applicative (de Lisp à la machine en passant par le Î»-calcul)* que « la programmation est une transmission d'un raisonnement à une machine, capable de le reproduire. Par cette reproduction, sans laquelle il n'est point de science, l'ordinateur s'affirme comme un instrument d'objectivation du raisonnement. Il s'inscrit dans la lignée des appareils qui ont permis de passer de l'expérience empirique à l'expérimentation scientifique : l'informatique se distingue des mathématiques par un outil d'expérimentation permettant d'observer, vérifier, réfuter un raisonnement. »

Cette « transmission d'un raisonnement à une machine » s'effectue par le truchement d'un langage : l'ordinateur a un langage en lui, constitué d'instructions qui effectuent des opérations sur des données. Les instructions sont des objets physiques, en l'occurrence des circuits électroniques qui réalisent ces opérations conformément aux règles de l'algèbre de Boole [1]. Quand on dit que l'ordinateur a un langage en lui, ce n'est pas une métaphore. Le langage constitué de ces circuits électroniques sera appelé *langage machine*.

Les données sont enregistrées dans la *mémoire* (l'anglais *storage*, moins anthropomorphique, serait moins source de confusion). Le texte du programme, qui mentionne les instructions à effectuer et les données auxquelles elles s'appliquent, est lui aussi enregistré dans la mémoire, l'unité de commande parcourt ce texte et exécute les instructions selon la séquence convenable. Enregistrer le texte du programme dans la même mémoire que les données est l'idée la plus révolutionnaire de l'architecture inventée par John von Neumann, elle transforme le programme d'objet matériel (les cartes perforées des métiers à tisser de Vaucanson et de Jacquard) en description textuelle d'un raisonnement abstrait, ce qui permet le développement pratique d'une science informatique dont Alan Turing avait formulé le principe théorique.

On peut programmer en langage machine : c'est difficile, laborieux, extraordinairement inefficace, et de plus chaque modèle d'ordinateur a son propre langage, qu'il faut apprendre à chaque fois. Aussi, depuis les années 1950, les informaticiens ont-ils dépensé une part considérable de leur énergie à créer des langages plus commodes, plus abstraits, moins liés aux caractéristiques physiques du processeur. Tous ces langages sont des métalangages du langage machine. Quand on a écrit un programme dans un de ces langages, afin qu'il puisse être exécuté, il faut le *traduire* en langage machine, le seul que le processeur puisse interpréter. Cette traduction est effectuée par un programme nommé *compilateur*, qui traduit le programme entier en langage machine [2]. Il existe une hiérarchie de ces langages, du plus concret (proche du langage machine, « de bas niveau ») au plus abstrait (« de haut niveau »). Au bas de cette hiérarchie se trouvent les langages assembleurs, dont les instructions correspondent à celles du langage machine (il y a donc un assembleur différent pour chaque modèle de processeur), mais écrites plus commodément, sous forme d'abréviations alphabétiques suivies de quelques opérandes, au lieu du code binaire du

langage machine.

Plus on monte dans la hiérarchie des langages, plus on gagne en abstraction, c'est-à-dire que l'on a moins à se préoccuper du fonctionnement technique de l'ordinateur, et que les programmes peuvent fonctionner indifféremment sur des ordinateurs de types différents, mais cette abstraction a un prix qui se paie en efficacité. Et pour écrire les programmes du système d'exploitation, qui établissent le lien entre ce que perçoivent les programmeurs et ce qui se passe dans les circuits, on aura toujours besoin de langages de bas niveau, tels que C, Rust ou les langages assembleurs.

Cette propriété de l'ordinateur, avoir un langage en soi, en fait une machine universelle. L'ordinateur est un automate qui peut traiter tous les problèmes pour lesquels existent des solutions calculables (il y a des problèmes sans solution et des solutions incalculables). La méthode de calcul d'une telle solution, cela s'appelle un algorithme ; la réalisation pratique de cet algorithme est un programme informatique. La programmation est l'art de réaliser des algorithmes pour calculer ces solutions. Entendons-nous bien : depuis Pascal et plus encore depuis Leibniz nous savons que le calcul ne se limite pas aux nombres, il s'étend à la logique, et Leibniz a été suivi sur ce terrain par Boole, Frege, Church et Turing, pour finalement donner naissance au versant théorique de l'informatique. Pour le versant pratique, que Pascal et Leibniz ont également parcouru, on nommerait plutôt von Neumann.

Voilà pourquoi l'ordinateur n'est pas du tout une machine comme les autres.

Afin d'expliquer la même chose sous un autre angle, on peut dire que pour obtenir d'un ordinateur qu'il vous fournisse des résultats, par exemple les tableaux statistiques du recensement de la population (puisque je travaille à l'Insee), il faut lui fournir des données, en l'occurrence les bulletins individuels recueillis par les enquêteurs de l'Insee et dûment enregistrés par des dames dactylocodeuses (c'était un métier presque exclusivement féminin) sur un support lisible par ordinateur, à l'époque des cartes perforées, plus tard des bandes magnétiques, puis toutes sortes de supports magnétiques. Avoir les données sur support informatique ne suffit pas, il faut également fournir à l'ordinateur les instructions précises et détaillées qui décrivent les opérations à effectuer pour obtenir les résultats voulus à partir des données disponibles.

Ce processus de description des opérations à effectuer, comme mentionné ci-dessus, se nomme la programmation, on écrit des programmes qui lisent les données et énumèrent, dans un langage spécial, les opérations qui mènent aux résultats (enfin on l'espère). Ces langages spéciaux, ou langages de programmation, diffèrent radicalement des langages humains : ils ne tolèrent ni l'ambiguïté ni le sous-entendu que l'ordinateur serait bien en peine d'interpréter, ils sont rigoureusement univoques, et très limités dans leur expressivité. En cette fin des années 1960 les principaux langages de programmation se nomment Fortran, Algol, Cobol, Lisp et PL/1, plus tard apparaîtront C, Pascal, Java, Ada, Python et Rust, peu importe, ils sont plus ou moins agréables selon les goûts mais finalement tous équivalents. Le travail de l'informaticien consiste donc, au premier chef, à écrire des programmes pour obtenir des résultats à partir de données. Un programme est un texte qui décrit les opérations qui mènent des données aux résultats. La programmation est proprement magique : il suffit de soumettre ces phrases à l'ordinateur pour qu'elles déclenchent des actions. Ainsi, dire c'est faire, ce qui relève d'un pouvoir divin.

On l'aura compris, l'informatique est présente tout au long de cette histoire, puisqu'elle a rempli ma vie, mais d'autres chapitres en développent plus particulièrement la description, les chapitres notamment.

[/Chapitre suivant/]

[1] L'algèbre de Boole, du nom de son inventeur George Boole (1815-1864), permet de modéliser des raisonnements logiques selon un formalisme qui se prête bien à la réalisation par des circuits électroniques. Toute l'informatique repose sur de tels circuits.

[2] Pour certains langages, tels que Python, le processus est différent, la traduction est faite ligne de texte par ligne de texte, au fur et à mesure que le programmeur les saisit. On dit que Python est un langage *interprété*, par opposition aux langages compilés ; le logiciel qui permet cette opération est un *interpréteur*. C'est plus facile pour les débutants, mais le programme exécutable produit est moins efficace. Le langage Scheme, dont il sera question plus loin, peut être soit interprété, soit compilé.