Notion de programme en tant que donnée. Calculabilité, décidabilité.

Notion de programme en tant que donnée

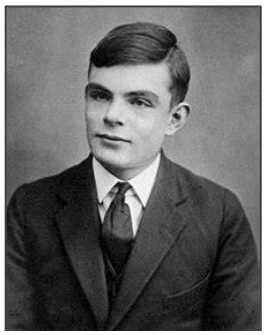
La **programmation** consiste à créer des suites d'instructions codées en binaire (le **langage machine**) destinées à être exécutées par une **machine numérique**. Le programmeur utilise des **langages de haut niveau** qui sont ensuite traduits en binaire.

Pour transformer le langage de haut niveau en code binaire, il existe trois type de langage de programmation :

- Les **langages compilés**, dans lesquels l'ensemble du code source est transformé en binaire par un programme appelé **compilateur** (C, C++, Fortran, Pascal, …).
- Les **langages interprétés**, dans lesquels chaque ligne du code source est transformée en binaire par un programme appelé interpréteur (Python, JavaScript, PHP, …).
- Les **langages semi-interprétés**, dans lesquels le code source est e transformée dans un langage intermédiaire (bytecode), qui sera traité par l'interpréteur (Java, C#, …).

On voit que les compilateurs et les interpréteurs utilisent comme données le code source d'autres programmes. D'une manière plus générale, les programmes en interaction agissent sur des données dont certaines sont d'autres programmes.

La machine de Turing


Alan Turing (1912-1954)

En 1936, le mathématicien anglais **Alan Turing** a publié un article dans lequel il donne la description d'une machine abstraite (machine de Turing) qui effectue des calculs de manière mécanique, sans intervention de l'homme, excepté pour l'entrée des données et la lecture des résultats. Il décrit ensuite une machine universelle (machine de Turing universelle) qui peut simuler n'importe quelle autre machine de Turing. Le programme de la machine particulière devient la donnée de la machine universelle.

Une machine de Turing est un modèle abstrait de programme, c'est-à-dire un objet mathématique.

Une machine de Turing comporte les éléments suivants :

- **un ruban infini divisé en cases consécutives**. Chaque case contient un symbole d'un alphabet fini donné ;
- **une tête de lecture/écriture** qui peut lire et écrire les symboles sur le ruban, et se déplacer vers la gauche ou vers la droite du ruban ;
- **un registre d'état** qui mémorise l'état courant de la machine de Turing ;
- **une table d'actions** (ou table de transition) qui indique à la machine quel symbole écrire sur le ruban, comment déplacer la tête de lecture (vers la droite ou vers la gauche), et quel est le nouvel état, en fonction du symbole lu sur le ruban et de l'état courant de la machine. Si aucune action n'existe pour une combinaison donnée alors la machine s'arrête.

La première machine à calculer mécanique avait été conçue en 1642 par **Blaise Pascal**. Les premiers métiers à tisser programmables (avec des cartes perforées) ont été fabriqués en 1801 par **Joseph Marie Jacquard**. En 1834, **Charles Babbage** a imaginé et construit la première machine à calculer utilisant des cartes perforées. La construction de calculateurs électronique a débuté dans les années 1930 (**Djon Atanasoff** et **Clifford Berry** pour l'ABC, **Konrad Zuse** pour la série Z1, Z2, Z3, **J. Presper Eckert et John Mauchly** pour l'ENIAC en 1943 et l'EDVAC en 1946).


Alonzo Church (1903-1995)

L'EDVAC a été conçu sur l'architecture de **John Von Neumann** qui en a attribué la paternité à Turing. Les données sont enregistrées en mémoire comme le décrivait Turing dans sa machine universelle et cette architecture est toujours d'actualité dans nos ordinateurs modernes !

Le mathématicien américain **Alonzo Church** et Alan Turing ont démontré que tous les problèmes résolubles par un algorithme sont résolubles par une machine de Turing universelle.

Calculabilité

La **Thèse de Church-Turin**g énonce que ***Si un problème est tel qu'il existe un algorithme le résolvant, alors il existe une machine de Turing qui résout le problème***.

Cette thèse mène à la notion de calculabilité :

- **On dit qu'un problème est calculable s'il existe une machine de Turing le résolvant.**
- **Un nombre est calculable** s'il existe un programme qui permet d'obtenir un à un la suite des chiffres de son écriture décimale, qui peut être infinie. Certains nombres sont calculables, d'autres ne le sont pas …
- **Une fonction f est calculable** si on peut calculer $f(x)$ avec une machine de Turing (en un nombre fini d'étapes pour tout x donné). Certaines fonctions sont calculables, d'autres ne le sont pas …

Décidabilité

Un problème qui peut se ramener à une fonction calculable peut être résolu. Un problème est décidable si et seulement s'il existe un algorithme qui le résout.

Le mathématicien autrichien **Kurt Gödel** a démontré en 1931 qu'il existe des propriétés mathématiques indécidables. Le mathématicien allemand **David Hilbert** a posé en 1932 la question en logique mathématique qu'on appelle le problème de la décision, en allemand « **Entscheidungsproblem** » : peut-on déterminer par un algorithme si un énoncé est vrai ou faux ? Alan Turing a démontré en 1936 que tous les problèmes n'étaient pas décidables à l'aide d'un contre-exemple : le problème de l'arrêt, qui est un problème indécidable.

Le problème de l'arrêt

Supposons qu'il existe un algorithme H qui décide :

$$\begin{cases} \mathrm{H(A)} \rightarrow 1 \text{ si A s'arrête} \\ \mathrm{H(A)} \rightarrow 0 \text{ si A ne s'arrête pas} \end{cases}$$

Si H existe alors $\overline{H}$ existe.

Considérons $\overline{H}(\overline{H})$.

- Si $\overline{H}$ s'arrête alors $H(\overline{H}) = 1$ et donc $\overline{H}(\overline{H})$ crée une boucle infinie.
- Si $\overline{H}$ ne s'arrête pas alors $H(\overline{H}) = 0$ et donc $\overline{H}(\overline{H})$ s'arrête.

Dans chaque cas, il y une contradiction, donc H ne peut pas exister. Le problème de l'arrêt est indécidable.