École de technologie supérieure

THÈME 0
Langages et ordinateurs

INF125 Introduction à la programmation
Sylvie Ratté et Hugues Saulnier

1. Ordinateurs et programmation

Le principe de l'exécution d'un programme est simple: le code binaire (appelé aussi code objet) est chargé en mémoire pour ensuite être exécuté instruction par instruction.

A. Indépendance ou dépendance machine

Mais plus personne aujourd'hui (ou presque) ne programme directement en binaire car la langue binaire est inintelligible pour un être humain car elle ne se compose que de 0 et de 1. Pour vous en donner une idée voici un fragment de code pour une machine dite de "Von Neumann" dont le design remonte à 1946; les ordinateurs modernes y ressemblent tellement que l'on dit souvent que nos ordinateurs possèdent une architecture Von Neumann. Voici le fragment:

```
00000010101111001010
00000010111111001000
00000011001110101000
```

Ce bout de code binaire additionne deux nombres situés aux emplacement 10 et 11 en mémoire (en binaire cela donne 1010 et 1011) puis range le résultat de cette addition à l'emplacement 12 (en binaire 1100). Le reste du fragment constitue la transcription binaire de l'instruction à effectuer.

Les langages d'assemblage constituent une variante du langage binaire en ce sens que des noms et des symboles viennent remplacer certaines séquences de bits. On peut ainsi écrire (notez que l'on peut désormais écrire les nombres en base 10) notre petit programme comme suit:

```
LOAD 10, A
LOAD 11, B
ADD  12
```

Cependant, les langages d'assemblage sont toujours de très bas niveau: chacune de leur instruction correspond (en général) à une seule instruction en langage machine. Lorsqu'on utilise un langage tel que le C, on obtient un langage de haut niveau dans lequel non seulement il est possible d'écrire le fragment précédent en une seule instruction (d = a+b;) mais l'on en arrive à devenir indépendant de la machine sur laquelle le programme sera appelé à être exécutée.

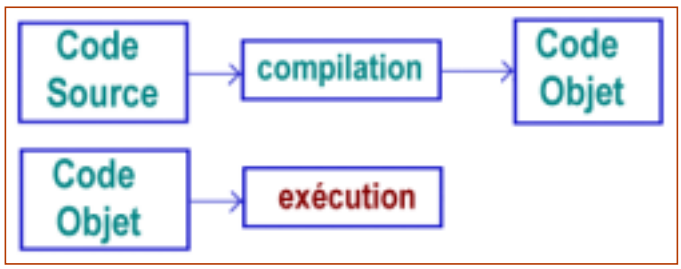
Évidemment, puisque la machine ne comprend que le code binaire, il faut bien qu'il existe un mécanisme pour traduire le programme exprimé dans le langage choisi dans la langue comprise par une machine spécifique. Ce processus se nomme "compilation" et dans le cas de la traduction des programmes écrits en langages d'assemblage, on nomme ce processus "assemblage".

Ainsi donc, plus le langage est " élevé ", plus le programme " traducteur " est complexe.

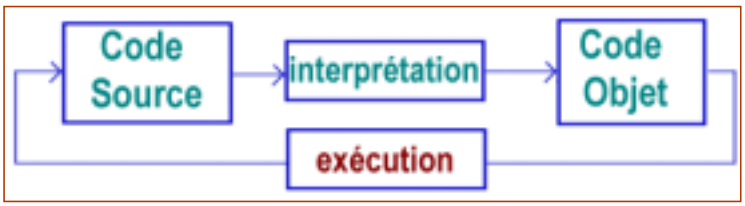
B Compilateur, interpréteur et développement

Il existe également des traducteurs que l'on nomme "interpréteurs" parce qu'il effectue, en fait, deux tâches: il transforme non pas tout un programme (comme le fait un compilateur) mais une instruction du code source (ex. une instruction Basic) en plusieurs instructions en langage machine pour ensuite les faire exécuter immédiatement par la machine. L'interpréteur passe ensuite à l'instruction suivante du programme source.

Exécution d'un programme compilé



Exécution d'un programme interprété



Environnements de développement (EDD)

Un EDD ou IDE (Integrated Development Environment) incorpore un éditeur, un compilateur, un dévermineur et un gestionnaire de fichiers. Ces quatre programmes rendent le développement plus aisé.

2. Langages de programmation

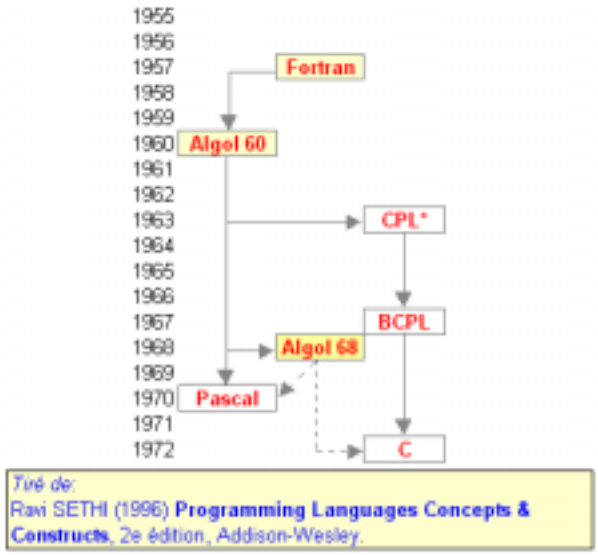
On peut suggérer une première classification historique des langages de programmation en suggérant quatre familles:

- Langages dits "scientifiques":
  le premier "Fortran" (Backus et al. 1957) … C, C++.
- Langages dits "algorithmiques" ou "algébriques":
  le premier " Algol" (Perlis & Samelson 1958) …Algol60, Algol68, Pascal, Modula, Ada, Basic.
- Langages dits "commercials":
  le premier " Cobol" (1955-56)...RPG
- Langages dits "symboliques":
  le premier "Lisp" (McCarthy 1958)… Snobol, Prolog

Mais ces familles suggèrent en fait que les langages sont dédiés à résoudre des problèmes dans des domaines particuliers. De nos jours, on parle plutôt de "paradigmes de programmation" et l'on discerne ainsi les quatre familles suivantes:

Langages impératifs:
  Langages centrés sur "l'action". Le programme est vu comme une séquence d'actions à faire: Fortran, C, Pascal, ...
Langages fonctionnels:
  Langages centrés sur le traitement symbolique des données: Lisp, ML, Scheme...
Langages objets:
  Langages centrés sur la notion d'objet: Simula (Nygaard & Dahl 1961-67), Smalltalk, C++, Java,...
Langages logiques:
  Langages centrés sur les raisonnements logiques: Prolog (Colmerauer & Roussel 1972)

Diagramme des "influences" historiques (les langages impératifs)



Les diagrammes suivants sont également disponibles:

- Les langages fonctionnels
- Les langages objets
- Tous les langages