Histoire de l'informatique

Chapitre 4

Histoire des langages de programmation

Pablo Rauzy <pr@up8.edu>
pablo.rauzy.name/teaching/hi

Histoire des langages de programmation

Introduction

- Les toutes premières choses qu'on peut rétrospectivement appeler "langages de programmation" sont les cartes perforées.
- De manière générale, les premières formes de langages de programmation étaient très contraintes par des restrictions matérielles.

- Dans les années 40 naissent quelques langages notoires :
  - Plankalkül, conçu par Konrad Zuse entre 1943 et 1945, mais jamais implémenté.
  - Le jeu d'instruction de la SSEM, première machine à architecture de von Neumann.

Plankalkül

- Plankalkül est considéré par son créateur comme le premier langage de programmation “haut niveau”.
- Il inclut déjà :
  - assignations,
  - sous-programme,
  - itération,
  - condition,
  - tableau,
  - enregistrements,
  - exceptions,
  - ...
- Il est très en avance sur son temps mais reste très largement inconnu (en partie parce que Zuse est en Europe dans un contexte historique qui permet difficilement à ses innovations de voyager).

- La *Small-Scale Experimental Machine* est construite à l'université Victoria de Manchester par Frederic C. Williams, Tom Kilburn, et Geoff Tootill.
- Elle sert avant tout de banc de test pour les *tube de Williams*, une forme primitive de mémoire.
- Elle a le rôle de "proof of concept" qui lancera le projet *Manchester Mark I* dont on a déjà parlé.
- Elle propose un jeu d'instructions sur 3 bits (en écriture moderne) :
  - `JMP S` (saut absolu),
  - `JRP S` (saut relatif),
  - `LDN S` (charge + négation),
  - `STO S` (stockage),
  - `SUB S` (soustraction),
  - `CMP` (saut d'une instruction selon le signe de la valeur dans l'accumulateur),
  - `STP` (stop).

Les premiers langages modernes

- Les premiers langages modernes apparaissent dans les années 1950 et 1960.
- On peut notamment retenir :
  - FORTRAN (FORmula TRANslator) en 1954,
  - LISP (LISt Processor) en 1958,
  - ALGOL (ALGorithmic Oriented Language) en 1958,
  - COBOL (COmmon Business Oriented Language) 1959,
  - BASIC (Beginner All-purpose Symbolic Instruction Code) en 1964.

- FORTRAN a été créé par John Backus chez IBM en 1954.
- À l'origine il a pour but de produire des cartes perforées, et sa syntaxe est extrêmement contraintes par cet aspect.
- Ses versions plus récentes et libérées de cette contrainte sont encore utilisées aujourd'hui, notamment pour le calcul scientifique.

- LISP a été créé par John McCarthy au MIT en 1958, en s'inspirant du λ-calcul.
- À la base, LISP a été publié dans un article intitulé "*Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I*".
- Dans cet article McCarthy montre qu'avec quelques opérateurs simple et une notations pour les fonctions, il est possible de construire un langage algorithmique Turing-complet.
- Il fait cela en utilisant les quelques opérateurs qu'il introduit pour construire en LISP une fonction `eval` qui est capable d'évaluer les programmes LISP.
- Steve Russel implémente cette fonction `eval` en code machine sur un IBM 704 et crée ainsi le premier interpréteur LISP.

Apports

- LISP est le premier langage où le code est représenté directement dans une structure de données standard du langage (on appelle aujourd'hui cette propriété l'*homoiconicité*).
- La syntaxe `if-then-else` a été inventée par McCarthy, mais il inclue dans LISP la forme `cond`, plus générale.
- LISP a eu une grosse influence sur les langages qui l'ont suivi, notamment sur Smalltalk (un des premiers langages orienté objet).
- LISP a introduit le concept de *glanneur de cellule automatique*.
- LISP a été très influent dans les débuts de l'intelligence artificielle.

Généalogie

- LISP a eu énormément de descendant et on parle aujourd'hui de Lisp comme d'une famille de langages.
- Les plus utilisés sont Common Lisp, Scheme/Racket, Clojure, Emacs Lisp.
- Mais Scheme est lui même une famille de langages...

- Dans un certains sens on peut aussi voir Lisp comme l'ancêtre de tout les langages fonctionnels.

- La première description d'ALGOL date de 1958, mais est stabilisé en 1960 par John Backus et Peter Naur.
- Il apporte deux innovations principales :
  - une structure en blocs imbriqués des programmes, sans avoir forcément besoin de nommer les blocs, et
  - la notion de portée qui permet de rendre certaines variables locales et apporte de la sûreté.
- C'est aussi le premier langage dont la syntaxe est décrite dans le métalangage BNF, aujourd'hui très utilisé.
- Son succès est principalement universitaire.
- Il a une influence considérable sur les futurs générations de langages de programmation.

- COBOL est développé en 1959 par un comité réunissant des gens de chez IBM, RCA, et Sylvania Electric Products.
- COBOL est conçu en vu d'être un langage commun pour la programmation de logiciel de gestion.
- Les spécifications sont largement inspirées du langage FLOW-MATIC de Grace Hopper.
- La complexité du langage en font aujourd'hui un sujet de rigolade, mais parfois jaune, car il existe toujours des applications de gestion écrites avec et qu'il est nécessaire de maintenir (notamment dans le milieu bancaire).

Exemple

```
000100 IDENTIFICATION DIVISION.
000200 PROGRAM-ID. SALUTTOUS.
000300 DATE-WRITTEN. 21/05/05 19:04.
000400 AUTHOR UNKNOWN.
000500 ENVIRONMENT DIVISION.
000600 CONFIGURATION SECTION.
000700 SOURCE-COMPUTER. RM-COBOL.
000800 OBJECT-COMPUTER. RM-COBOL.
000900
001000 DATA DIVISION.
001100 FILE SECTION.
001200
100000 PROCEDURE DIVISION.
100100
100200 DEBUT.
100300 DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
100400 DISPLAY "BONJOUR !" LINE 15 POSITION 10.
100500 STOP RUN.
```

- BASIC a été conçu en 1964 par John G. Kemeny et Thomas E. Kurtz au Dartmouth College pour permettre aux étudiants des filières autre que scientifiques d'utiliser des ordinateurs.
- Il se veut simple et interactif :
  - ses sept instructions devaient pouvoir être enseignées en une demi-journée,
  - les programmes devaient pouvoir s'exécuter jusqu'à la première erreur,
  - les messages d'erreurs devaient être compréhensible,
  - ne pas nécessiter de connaissance du matériel ou du système.

L'apparition des paradigmes fondamentaux

- La période des années 1960 et 1970 a vu un véritable foisonnement de langages de programmation.
- La plupart des paradigmes des principaux langages sont inventés durant cette période :
  - les prémisses de la programmation orientée objet avec Simula,
  - la programmation système avec C,
  - les environnements de développement intégrés avec Smalltalk,
  - la programmation logique avec Prolog,
  - la programmation généraliste fonctionnelle avec ML.
  - certains langages spécifiques à leurs domaines comme SQL.
- C'est à cette période qu'a pris place le grand débat sur la programmation structurée ("*GOTO statement considered harmful*").

Simula

- Simula I (SIMple Universal LAnguage) a été créé en 1962 par Ole-Johan Dahl et Kristen Nygaard à partir d'Algol 60.
- Il évolue en 1967 en Simula 67 et intègre le premier modèle de classe de Hoare, ce qui en fait le premier langage à classe et le parent des Smalltalk, C++, Java, Eiffel, etc.
- Son but est d'offrir un support pour la programmation de simulation à évènements discrets.
- Il intègre déjà les concepts principaux qui permettrons la programmation orientée objet :
  - les classes,
  - `new`,
  - `this`,
  - héritage,
  - restriction d'accès,
  - ramasse-miettes,
  - appel de méthode avec la notation `.`,
  - fonctions virtuelles,
  - ...

- Le langage C est inventé en 1972 chez Bell par Dennis Ritchie pour la réécriture d'UNIX.
- Son influence a été considérable sur les langages de haut niveau qui lui ont succédé (même si C est souvent qualifié de bas niveau ou alors d'"assembleur portable").
- Le but C étant d'être proche de la machine, la norme qui le défini laisse volontairement certains comportement "défini par l'implémentation", voire "non spécifié" ou "indéfini".
- C'est un des langages les plus utilisés jusqu'à aujourd'hui.
- La dernière version du standard date de 2011 et continue à faire de C un langage moderne.

Smalltalk

- Smalltalk a été créé en 1972. Il est inspiré par les langages Lisp et Simula.
- Il a été conçu par Alan Kay, Dan Ingals, Ted Kaehler, Adele Goldberg au Palo Alto Research Center de Xerox.
- Les principaux concepts de Smalltalk sont :
  - tout est objet,
  - tout est modifiable,
  - typage dynamique,
  - exécution dans une machine virtuelle (compilation vers du bytecode),
  - tout se fait via le paradigme d'envoi de message (pas de structure de contrôle).
- Il a eu une grande influence sur Java, Ruby, Objective-C, Self, Python, ...
- Il est toujours utilisé aujourd'hui, bien qu'en moindre mesure (Pharo, Squeak).

Prolog

- Prolog (PROgrammation LOGique) a été créé par Alain Colmerauer et Philippe Roussel vers 1972.
- Le but était de créer un langage de programmation où seraient définies les règles logiques attendues d'une solution et de laisser le compilateur la transformer en séquence d'instructions.
- Ça a été très utilisé en intelligence artificielle notamment pour le traitement de la langue naturelle.
- Cependant son influence a au final plutôt consisté en la création de moteur d'inférence dans des bibliothèques pour d'autres langages.

ML

- ML (Meta Language) est inventé par Robin Milner en 1973 à l'université d'Édimbourg.
- Il est construit sur un typage statique fort et polymorphe au-dessus de Lisp, pour le système de preuves formelles LCF (le typage de Lisp permettait de "prouver" des assertions fausses).
- Les langages fonctionnels qui dérivent de ML apportent une grande sûreté lors de l'exécution de programme, entre autre par leur typage qui permet d'attraper les bugs à la compilation.
- La variante la plus populaire de ML à l'heure actuelle est sans doute OCaml, un langage fonctionnel pragmatique (permettant notamment de générer du code rapide).

SQL

- SQL est inventé en 1974 chez IBM.
- Il fait suite à un article de 1970 de Edgar Frank Codd qui publie "*A Relational Model of Data for Large Shared Data Banks*".
- SQL est un langage spécifique à un domaine, il sert à manipuler des bases de données relationnelles.

Consolidation

- Dans les années 1980, on assiste plus à une consolidation des langages de programmations et des techniques de développement qu'à de réelles nouveautés.
- Certains paradigmes fusionnent (OO et système dans C++ par exemple).
- Les langages élaborés pendant la décennie précédente continuent d'être normalisés.
- La programmation modulaire et la programmation objet s'impose comme pratique majoritaire.
- En partie grâce à l'architecture RISC, de gros progrès sont fait dans les techniques de compilation.

- La décennie des années 1980 voit tout de mêmes quelques innovations, mais qui resteront longtemps confinées au milieu académique.
- Par exemple les langages d'assistants de preuve, servant aussi bien à faire des preuves mathématiques que des preuves de propriétés de programmes :
  - Coq,
  - Isabelle,
  - ACL2.
- On peut aussi parler des langages synchrones, destinés à l'embarqué temps-réel :
  - Esterel,
  - Lustre.

Internet et langages de scripts

- La décennie suivante, les années 1990, correspond à l'essor d'Internet et l'arrivée de nouvelles méthodes de développement rapide.
- On voit notamment naître les IDE.
- Aussi de nombreux langages de script, qui n'innovent pas vraiment mais mettent en pratique et combinent d'anciennes idées dans le but d'augmenter la productivité des programmeurs :
  - Python en 1991,
  - Ruby en 1993,
  - Lua en 1993,
  - JavaScript en 1995,
  - PHP en 1995.
- On peut également noter :
  - Haskell en 1990,
  - Common Lisp en 1994,
  - Java en 1995.

Tendances actuelles

- Depuis les années 2000, la tendance est plus à la sûreté et la sécurité.
- On cherche à ce que les compilateurs apportent des garanties fortes de sûreté à l'exécution, voire de sécurité.
- Certains paradigmes se prêtant mieux à ces objectifs refont leur apparitions sur le devant de la scène.
- En particulier depuis quelques années, la programmation fonctionnelle sort du milieu académique et fait son entrée dans l'industrie.
  - Par exemple Rust développé depuis 2010 par Mozilla intègre énormément d'aspect des langages fonctionnels.
  - Ou encore F*, développé depuis quelques années chez Microsoft, qui reprend OCaml mais lui ajoute des types dépendants permettant de prouver des propriétés du programme.
- On peut aussi noter que certains paradigmes restés jusque là confidentiels, comme la programmation synchrone avec Esterel ou Lustre, sont aussi mis en pratique industriellement.