Histoire de l'informatique

Tome 2 : Les langages informatiques

Par Dimitri PIANETA

Edition 2020

Préface

Ce tome 2 de l'histoire de l'informatique à pour thème les langages informatiques. J'ai voulu regroupé les notions importantes de la programmation informatique.

Faire un bilan des nombreux langages de programmation qui existe et qui a existé.

J'ai décomposé ce tome 2 en 5 chapitres. Dans un premier chapitre, j'ai effectué un historique des langages de programmation selon la littérature. Dans un deuxième chapitre, une liste non exhaustive des langages de programmation. Dans un troisième chapitre, les notions basent d'un langage et je n'ai pas voulu dans ce tome 2 de traiter la théorie des langages et les notions de compilation qui sont liés. Un quatrième chapitre, sur les personnages qui ont contribué aux langages de programmation cités en chapitre 1. Un dernier chapitre de vocabulaires avec certaines notions de l'informatique pour le développement et quelques explications de certains langages cités dans ce tome 2.

Je vous souhaite une bonne lecture de Tome 2.

Dimitri PIANETA

Table des matières

Chapitre 1 : Historique

1.1 Repère

Année	
1843	Ada Lovelace est la première personne à avoir créé et décrit un programme informatique. En 843, elle construit un algorithme pour le calcul du nombre de Bernoulli utilisant le **Analytical Engine***.
1889	Le Hollerith tabulating machine a été inventé par Herman Hollerrith en 1889, permet de programmer un conteur et tabulation.
1956	Le premier langage de programmation est le **FORTRAN**. Ce langage a été présenté au public le 15 Octobre 1956. Il a été développé par John Backus et autres chez IBM.
1958	Le second plus ancien langage est le **LISP** qui a été développé par John McCarthy et a été utilisé en premier en 1958.
1959	**COBOL** a été développé en 1995 par Grace Hopper et Bob Bemer.
1964	L'origine du langage **BASIC** ont été développé par John Kemeny, Mary Keller et Thomas Kurtz et présentait au public le 1 Mai 1964.
1965	**Simula** est considéré comme le premier langage de programmation orienté objet (POO) et développait en 1965 par Ole-Johan Dahl et Kristen Nygaard.
1966	Martin Richard développait le langage de programmation **BCPL** en 1966, qui est devenu très populaire et très portable.
1966	Le langage de programmation **MUMPS** ont été développé par Neil Pappalardo à Massachusetts General Hospital en 1966.
1967	La capacité graphique se développe alors le langage **LOGO** qui a été crée par Seymour Papert en 1967.
1971	**Pascal** a été développé en 1971 par Niklaus Wirth.
1972	Dennis Ritchie et Brian Kernighan développe le langage de programmation **C** dans les laboratoires de Bell en 1972.
1972	Le langage de programmation **Prolog** a été développé par Alain Colmerauer et ces collègues en 1972 à l'Université de Marseille.
1972	**Smalltalk** était le second langage de programmation orientée objet et le premier vrai IDE, développait par Alan Kay et autres pour Xerox PARC en 1972.
1974	**SQL** est le langage de programmation des bases de données. Il a été développé par Edgar Codd en 1974 et encore beaucoup utilisé dans le monde du développement.
1975	Une variante de LISP, le langage de programmation **Scheme** a été dévéloppé en 1975 par Guy Steele et Gerry Sussman à MIT's Artificial Intelligence lab.
1975	Le **Altair BASIC** a été développé par Bill Gates, Paul Allen et Monte Davidoff et ont été produit le 2 janvier 1975. Ce qui a permit de créer l'ordinateur Altair.
1979	Le C++ a été développé en 1979 par Bjarne Stroustrup. Originalement appelait le C with classes.
1979	Oracle s'inspire de SQL pour créer une première version commerciale de SQL en 1979.
1979	Le département de Défense américaine ont développé le langage de programmation **Ada**, original DoD-1 et nommé après à l'hommage de Ada Lovelace en Mai 1979.
1984	**FoxPro** est un langage de programmation pour développé des applications de base de données et proposé par Fox Software en 1984.
1984	Cleve Moler début à développé le langage de programmation MATLAB en 1970 et proposé au public par des packages MATLAB en 1984.
1987	L'open source a crée le langage de programmation **Perl** qui a été développé par Larry

	Wall était introduit en 1987. C'est très commun d'utiliser ce langage pour créer des scripts CGI et pour les applications web.
1988	Développé en mi-1980 par Brad Cox et Tom Love, le **Objective-C** est officialisé par licence NeXT en 1988.
1990	Tim Bernes-Lee développe le **HTML** en 1990. Le HTML est le langage le plus populaire et le plus utiliser dans le monde.
1990	**Haskell**, un langage de programmation polyvalent.
1990	Les ingénieurs APPLE ont développé le langage de programmation Dylan dans les années 1990. **Dylan** était désigné rassemblant à la syntaxe du langage de programmation ALGOL.
1991	Le développement de **Python** a commencé en 1989 par Guido van Rossum et présentait au public en 1991.
1991	**Visual Basic** était développé par Alan Cooper et proposé en Mai 1991.
1993	**Lua** était crée en 1993 par les ingénieurs de Pontifical Catholic University of Rio De Janeiro, au Brésil.
1993	**R** est un langage de programmation créé par Robert Gentteman et Ross Ihoko et présentait en 1993.
1994	Le concept de **CSS** était commencé par Hakon Wium Lie en 1994. W3C introduit la spécification pour CSS en 1996.
1995	**Java** est développée par James Gosling et autres développeurs de Sun Microsystems et était présenté au public en 1995.
1995	Le langage de programmation orientée objet **Ruby** était développé par Yukihiro Matsumoto était présenté au public en 1995.
1995	L'expérimentation, du muti-paradigme du langage de programmation **Curry** était présentée par Michale Hanus, Herbert Kuchen et Juan Jose Moreno-Navarro en 1995.
1995	Le **PHP** était développé par Rasmus Lerdorf qui a commencé en 1994 et sortie le 8 Juin 1995.
1995	Originalement appelait **LiveScript** quand la sortie en Novembre 1995, **Javascript** était développé par Brendan Eich et renommait en Décembre 1995.
1996	Présentation en 1996, de **OCaml** est un langage orientée-objet de **CAML**.
1998	**XML** est un langage Markup avec les spécifications qui sont données par W3C et depuis le 10 février 1998.
1999	Le langage de programmation **D** a commencé d'être conçu en Décembre 1999. D est un langage de haut niveau comparé à C++.
2000	Le **C#** est basé des langages JAVA et C++ qui a été développé par Microsoft et présentait en Juin 2000. C# devenu un standard ISO en 2003.
2003	La programmation orientée objet voit introduit le **SCALA** en 2003.
2005	Don Syme développait le langage de programmation **F#** et Microsoft est le premier à le proposer en 2005.
2007	Le langage de programmation **GO** a été développé par Google au début 2007. Il y a complètement était présenter au public en 2009.
2007	Rich Hickey développe le langage **Clojure** et fini la première version en 2007.
2008	Présentation en 2008, **Nim** est un langage de programmation utilisant les limites de la mémoire.
2008	La programmation orientée objet en 2008 un nouveau langage le **Reai**.
2010	Le multi-paradigme **CoffreScript** qui permet de compilé le JAVASCRIPT.
2011	Google développait en open Source web le langage **DART** qui a été présenté en octobre 2011.
2012	**Julia** était développée par Jeff Bezanson, Alan Edelman, Stefan Karpinki et Vital B. Shah et présentait en 2012.
2014	**Babel** est un langage généraliste développé en 2014 et qui permet de gérer la

	consommation de la Batterie et les ressources de l'appareil.
2014	Création de Apple et présentait le 2 Juin 2014, le langage de programmation **Swift** créer un programme et une application pour iOS, macOS, Apple Watch et AppleTV.
2015	Graydon Hoare commençait à développer le langage RUST en 2010. Après une contribution de 100 personnes, il a officialisé la version 1.0.0 apha par Mozilla le 9 Janvier 2015.

1.2 Autres classements

Dans les années 60 :

1951 - Langage assembleur
1952 - Autocode
1954 - IPL (précurseur de LISP)
1955 - FLOW –MATIC (conduit à COBOL)
1957 - FORTRAN (Premier compilateur)
1958 - LISP
1958 - ALGOL 58
1959 - FALT (précurseur de COBOL)
1959 – COBOL
1959 – RPG
1962 – APL
1962 – Simula
1962 – SNOBOL
1963 – CPL (précurseur de C)
1964 - Speakeasy (environnement de calcul)
1964 – BASIC
1964 – PL/I
1966 – JBOSS
1967 – BCPL (précurseur de C)

Dans les années 70 :

1968 – LOGO
1969 – B (précurseur de C)
1970 – Forth
1972 – C
1972 – SmallTalk
1972 – Prolog
1973 – ML
1978 – SQL

Dans les années 80 :

1980 – C++
1983 – Ada
1984 – Lisp Commun
1984 – Matlab

1984 – dBaseIII, dBaseIIIPlus
1985 – Eiffel
1985 – Objectif-C
1986 – LabView (langage de programmation visuel)
1986 – Erlang
1987 – Perl
1988 - TCL
1988 – Wolfrom Language
1989 – FL (Baclus)

Dans les années 90 :

1990 – Haskell
1991 – Python
1991 – Visual Basic
1993 – Lua
1993 – R
1993 - CLOS (partie de Lisp ANSI)
1995 – Ruby
1995 – Ada 95
1995 – JAVA
1995 – Delphi (Object Pascal)
1995 – Javascript
1995 – PHP
1997 – Rebol

Dans les années 2000 :

2000 – ActionScript
2001 – C#
2001 – D
2002 – Scratch
2003 – Goovy
2003 – Scala
2005 – F#
2006 – PowerShell
2007 – Clojure
2009 – Aller
2010 – Rust
2011 – Dart
2011 – Kotlin
2011 – Red
2011 – Elicir
2012 – Julia
2014 – Swift
2016 – Bague

Chapitre 2 : Les langages

A

- **A+**. 2001 Proche de APL.
- **A#**. Orienté objet, langage de programmation fonctionnel, maintenant remplacé par Aldor.
- **ABAP**, Advanced Business Application Programming. 1983. Langage de programmation proche de Cobol pour les serveurs d'applications Web de SAP.
- **ABC**. Précurseur de Python.
- **Action!** Langage de conception de compilateur, comme Micro-SPL.
- **ActionScript**. 2004. Version de ECMAScript pour Flash.
- **Actor**. 1986. Langage de programmation et aussi concept pour une façon de concevoir un langage (orienté acteur).
- **Actum**. 2009 par Microsoft. Experimental, concurrent sur le modèle d'acteurs. Interne à la firme.
- **Ada**. 1983. Nommé d'après Ada Lovelace, développé pour le département de la défense américain.
- **Afnix**. 1998. Anciennement Aleph. Langage fonctionnel.
- **Agena**. 2009. Inspiré d'ALGOL et C.
- **Aldor**. 1985. IBM. Pour le traitement mathématique.
- **Aleph**. Voir Afnix.
- **Algae**. Langage interpreté pour l'analyse numérique.
- **Algo**. Langage de programmation algébrique.
- **ALGOL**, ALGOrithmic Language. 1958. Suivi de d'ALGOL 60, d'ALGOL W (Wirth) puis d'ALGOL 68. A inspiré Pascal.
- **Alma-0**. Modula 2, langage impératif, augmenté de fonctions de programmation logique.
- **Alphard**. 1974. Nom de l'étoile la plus brillante dans Hydra. Pascal-like, non implémenté.
- **Altran**. 1968. Variante de Fortran.
- **AmigaE**. 1993. Wouter van Oortmerssen. Langage inspiré par Ada, C++, Lisp.
- **AMPL**, A Mathematical Programming Language. 1985 par Brian Kernighan et autres. Langage de modélisation pour la programmation mathématique.
- **ANI**. 2010. Implicitement parallèle. Le projet semble abandonné.
- Anubis. 2000. Langage fonctionnel, non ML.
- **ApeScript**. 2005. Dyanmique, interprété, C-like.
- **APL**. A Programming Langage. 1962. Par Kenneth E. Iverson.
- AppleScript. 1993. Langage de script proche de l'anglais.
- **APT**. Automatically Programmed Tool. Langage de haut niveau pour les machines controlées numériquement.
- **Arduino**. Une version du langage wiring pour le controleur open source Arduino.
- **Argos**. Langage synchrone.
- **ARS++.** Abstraction plus Reference plus Synthesis. Nouvelle approche de programmation donnée dans le nom.
- **Asm.js**. Sous-ensemble de JavaScript qui s'exécute plus rapidemment. Il est implémenté par Mozilla.
- **AspectJ**. Implementation Java de la programmation orientée Aspect. Développement Assembly.
- **ATLAS**. Plusieurs langages mineurs portent ce nom.
- **Autocode**. 1952. Il y a plusieurs versions de ce langage primaire historique.
- **AutoIt**. Langage d'automatisme. A l'origine pour des scripts d'applications Windows, plus général maintenant.
- **Avail**. 2014. Proche du langage naturel en anglais, fonctionne sur une machine virtuelle.

- **Averest**. Langage Synchrone, remplacé par Quartz.
- **AWK**, Alfred **A**ho, Peter **W**einberger, and Brian **K**ernighan. 1977. Voir aussi gawk, nawk, mawk. Interprét, pour le traitement de chaîne, l'extraction de données.
- **Axiom**. Système d'algèbre automatisée, en fait un ensemble d'outils utilisant le langage A#.

B

- **B**. 1969. Dennis Ritchie et Ken Thompson. Dérivé de BCPL, dont il raccourci le nom, prédécesseur du langage C.
- **Bash**. Bourne-Again shell. 1989. Un interpréteur de ligne de commande pour succéder à Bourne shell.
- **BASIC**. Beginner's All-purpose Symbolic Instruction Code. 1964. John G. Kemeny et Thomas E. Kurtz. Destiné à permettre aux étudiants du Darmouth Collège d'utiliser un ordinateur, il s'est popularisé avec les micro-ordinateurs.
- **BAL**. Langage d'assemblage pour IBM 360.
- **BCPL**, Basic Combined Programming Language. 1966. Par Martin Richards à Cambridge. Successeur de CPL, inspiré de BASIC, a inspiré B lequel a lui-même à inspiré C.
- **BeanShell**. 2000. Scripting Java-like.
- **BETA**. Orienté objet dans la tradition Simula, C-like.
- **Bigwig**. Descendant de MAWL, pour réaliser des services web.
- **Bistro**. 1999. Proche de Smaltalk et Java.
- **BLISS**. 1970. Langage système par Carnegie Mellon, supplanté par C.
- **Blockly**. 2012 par Google. Langage graphique, on déplace des blocs pour créer une application.
- **Boo**. 2004. Similaire à Python sur .NET.
- **Bosque**. 2019. Par Microsoft. Un autre C-like conçu pour explorer de nouveau concepts dans l'expression des programmes.
- **Bourne shell**. 1977. Langage de commandes pour Unix.
- **Bournegol**. 1977. Portage d'Algol réalisé avec des macros C, utilisé pour écrire Bourne shell. Le nom pourrait être apocryphe.
- **BPEL**. Web Services Business Process Execution Langage. 2003. Langage standardisé par OASIS pour exprimer les processus professionnels dans les services Web.

C

- **C**. 1972. Conçu par Dennis Ritchie pour écrire le code du système d'exploitation Unix.
- **C--.** 1997. Langage intermédiaire portable destiné aux compilateurs. A la différence de LLVM, une interface au runtime pour ajouter des traitements tel un garbage collector.
- **C++**. 1983. Par Bjarne Stroustrup. S'appelait C with Classes jusqu'en 1983. Le standard est C++ 98 auquel a succédé C++ 11 en 2011.
- **C#**. 2000. Par Microsoft comme alternative à Java et dérivé aussi de C++. C'est un langage impératif et OO complet.
- **C Shell**. 1978. C-like pour scripts en ligne de commande sur Unix. Son successeur est tcsh.
- Caché ObjectScript. 1997. Langage procédural avec des fonctions de base de données. Compatible avec MUMPS.
- **Caml**, Categorical Abstract Machine Langage. 1985. Dérivé de ML, prédécesseur de OCaml.
- Cayenne. fonctionnel, proche de Haskell avec des aspects Java et les valeurs de retour peuvent dépendre de composants externes.
- **Cecil**. 1992. Proche de Modula et Objective C. (Chercher sur le site).
- **Cedar**. 1983. Palo Alto. Successeur de Mesa et de Pascal.

- **Ceylon**. 2012. Créé par Red Hat pour permettre d'écrire collectivement des programmes et utiliser des données structurées. Il ressemble à JavaScript avec des classes et interfaces mais fonctionne sur la JVM ou Node.js. Note: Ceylon (Ceylan) = thé, Java = café.
- **CFScript**. Partie JavaScript de ColdFusion. Voir aussi CFML.
- **Cg**. C for Graphics. C-like par NVidia et Microsoft pour les cartes graphiques.
- **Chapel**, Cascade High Productivity Language. 2009 par Cray, fabricant de superordinateurs. Programmation parallèle, C-like.
- **Charity**. 1992. Langage fonctionnel et catégorique.
- **CHILL**. CCITT High Level Language. 1980. Langage de télécommunications. Chill 96 est orienté objet et générique.
- **CHR**. 1991. Constraint Handling Rules. Utilisé dans l'intelligence artificielle.
- **ChucK**. 2004. Langage multimedia et concurrent pour la synthèse audio, et autres tâches musicales.
- **Cilk**. 1994. Multi-thread et concurrent basé sur C.
- **Clarion**. 2011. Doté de commandes de base de donnée, automatise la réalise d'applications de rapport.
- **Clay**. 2011. Essai de langage générique.
- **Clean**. 1987. Similaire à Haskell.
- **Clipper**. 1985. Compilateur pour dBASE III qui s'est doté de fonctions de C et Pascal.
- **CLIPS**. C Langage Integrated Production System. Voir Cool.
- **Clojure**. 2007. Lisp-like, compilé en bytecode pour la JVM.
- **CLOS**. Voir Common Lisp.
- **CLU**. CLUster. 1975. Par le MIT. A apporté des concepts qui ont inspiré Python et Ruby.
- **Cobol**. COmmon Business Oriented Langage. 1959. Inspiré par Flow-matic, Fortran. Les standards ANSI sont Cobol 58, 74, 85et 2002 oriented objet.
- **Code**. Computationally-Oriented Display Environment. Système de programmation visuel et parallèle.
- **CoffeeScript**. 2009. Il se compile en JavaScript et offre une syntaxe plus lisible style Python.
- **ColdFusion**. 2001. Combinaison de CFScript et CFML compatible JavaScript, utilisé pour le Web dynamique.
- **COMAL**. Common Algorithmic Language. 1973. Inspiré de BASIC.
- **CIL**. Common Intermediate Langage. Bytecode pour .NET.
- **Common Lisp**. 1984. Dialect of Lisp, ANSI standard.
- **Component Pascal**. Voir Oberon.
- **COMIT**. 1957. Premier langage de traitement de listes ou de chaînes de caractères.
- **Cool**. Classroom Object Oriented Language. 1996. Designed to instruct compiler building.
- **Coral66**. Computer On-line Real-time Applications Language. 1964. Basé sur Algol 60 et Fortran, était utilisé par l'administration britannique.
- **COWSEL**, COntrolled Working SpacE Language. 1964. Renommé POP-1, suivi de POP-2.
- **CPL**, Combined Programming Language. 1963. Prédécesseur de BCPL et lui-même inspiré de Algol 60.
- **Crack**. 2009. Langage de scripting conçu pour avoir la vitesse d'un programme compilé. C-like, LLVM.
- **Crystal**. 2015. Ruby-like, compilé.
- **Csh**. Voir C Shell.
- **Curl**. CURLy bracket. 1998. Langage de données comme HTML et de programmation, OO, réflexif pour construire des applications Web. (Ne pas confondre avec cURL.
- **Curry**. Nom d'un mathématicien. 1996. Fonctionnel et logique, basé sur Haskell.
- **Cyclone**. 2006. Dialecte de C par ATT, conçu pour être plus sur, éviter les fuites de mémoire et problèmes de pointeurs.

D

- **D**. 2000. Par Walter Bright. Une nouvelle version de C à objets, tableaux dynamiques et garbage collector.
- **Databus**. Voir PL/B.
- **DarkBASIC**. 1999. Langage de création de jeux commercial. Compile en C++ avec une extension BASIC.
- **Dart**. 2011. Par Google. Un langage fonctionnant dans le navigateur ou sur le serveur pour remplacer JavaScript. Ajoute classes, interfaces et mixins.
- **DCL**. DIGITAL Command Langage. ~1977. Scripting sur ordinateurs Digital.
- **Deca**. 2011. Langage de haut-niveau pour la programmation système. Utilise LLVM.
- **Delphi**. 1995. Version de Pascal créée par Borland, actuellement maintenue par Embarcadero.
- **DiBOL**, Digital's Business Oriented Language. 1970. Inspiré de BASIC et COBOL pour le traitement de l'information.
- **DisCo**. DIStributed CO-operation. 1992. Langage de spécification pour des systèmes réactifs.
- **Dotty**. 2014. Nouvelle version simplifiée de Scala.
- **DRAKON**. 1996. Langage visuel par le programme spatial russe, exprime la connaissance permettant d'accomplir un but.
- **Dylan**. DYNamic LANguage. 1992 par Apple. Dérivé de Scheme. Totalement orienté objet, il avait été créé pour le Newton.

E

- **E**. 1997. Voir aussi AmigaE. Dérivé de Joule, pour le traitement distribué et persistant.
- **Ease**. 1991. Inspiré par Csp and Linda. Les *contexts* sont des structures et types parallèles construits dynamiquement.
- **ECMAScript**. 1997. Le standard officiel pour JS.
- Edinburgh IMP. Voir IMP.
- **Eiffel**. 1986. Par Bertrand Meyer. Langage conçu pour la sécurité.
- **Elan**. 1974. Pour apprendre et enseigner la programmation systématique en remplacement de BASIC.
- **elastiC**. 1999. C-like, OO, portable, interprété.
- **Elixir**. 2012. Fonctionnel et concurrent, compatible avec la machine virtuelle Erlang (BEAM), mais avec une syntaxe Ruby-like. Un programme elixir peut accéder à son code source et le tester.
- **Elm**. 2012. Programmation réactive fonctionnelle, compile en HTML, CSS et JavaScript.
- **Emacs Lisp**. Scripting pour l'éditeur de code.
- **EGL**. 2008. Enterprise Generation Language, par IBM. Basé sur Cross System Product crée en 1981. Langage de très haut niveau qui est compilé en d'autres langages comme COBOL, Java, etc.
- **Epigram**. 2004. Concurrent, fonctionnel.
- **Erlang**. 1986, open source en 1998. ERicsson LANguage et aussi du nom de Agner Krarup Erlang. Inspiré par Prolog, Smalltalk, CSP. Fonctionnel, concurrent avec un runtime et une machine virtuelle (BEAM). Le modèle à acteurs résoud la plupart des problèmes de concurrence.
- **Escapade**. 1997. Programmation coté serveur pour accès aux bases de données et produire des pages web.
- **Esterel**. 1980. Par l'INRIA. Pour le développement de systèmes réactifs synchrones complexes, avec parallélisme et préemption.

- **Euclid**. 1970+. Par le Xerox PARC lab. Pascal-like impératif pour des programmes vérifiables. Son successeur est Mesa.
- **Euphoria**. 1993. Langage de script interprété.
- **Euler**. 1966. Niklaus Wirth et Helmut Weber. Successeur de Algol 60. Typé dynamiquement.
- **Exec**. Voir Rexx.

F

- **F**. Sous-ensemble de Fortran 77 avec modules et accès au système de fichiers.
- **F3**, Form Follow Function. 2005. Nom original de JavaFX Script, mais forké (relancé) sous ce nom.
- **F#**. 2005. Microsoft. Fonctionnel, OO, inspiré par OCaml, Haskell et autres langages fonctionnels.
- **Fabric**. 2010, Cornell. Dérivé de Java, distribué, il incorpore des dispositifs de sécurité pour la conservation et l'utilisation de l'information.
- **Factor**. 2003. Basé sur une pile comme Forth
- **Fantom**. 2005. C-like s'exécutant sur JVM et .NET avec un bibliothèque commune. Syntaxe évolutive, concurrence, mixins.
- **Felix**. Inspiré par C++ et ML.
- **Flow-Matic**. 1955 par Grace Hopper. Premier langage utilisant des mots anglais dans des instructions.
- **Focal**. FOrmula CALculator. 1968. Interprété pour PDP-8.
- **FOCUS**. 1975. Construit des requêtes de base de données.
- **FOIL**. 1967. Apprentissage par ordinateur. Un autre langage de ce nom pour générer de la musique est apparu en 1979.
- **Forth**, FOuRTH. 1973. Par Charles H. Moore. Utilise une pile. Sert à commander des machines incluant le démarrage d'ordinateurs.
- **Fortran**. 1957. FORmula TRANSlator. Les standards sont Fortran II (58), IV (61), 66, 77 (Procedural), 90, 95, 2003 (Orienté objet). Langage de calculs scientifiquess. Autres dialectes: S-Fortran, SFtran, QuickTran, LTRTran, HPF, Co-Array Fortran.
- **Fortress**. 2007. Conçu par Sun pour de hautes performances. Présenté comme un remplaçant de Fortran, d'où le nom.
- **FP**, Function Programming. 1977. John Backus. Créé pour mettre en oeuvre la programmation fonctionnelle.

G

- **G**. 1986. Langage de "dataflow" pour le système LabVIEW, graphique et parallèle (et fonctionnel). On programme visuellement en connectant des objets.
- **GAMS**, General Algebraic Modeling System. 1976-1987. Système de modélisation pour optimisation mathématique.
- **Go**. 2009. Langage par Google inspiré par C et Pascal. Il est concurrent avec un garbage collector, destiné surtout aux services web.
- **Gödel**. 1995. Prolog-like.
- **Gosu**. Dérivé de Java et fonctionnant sur machine virtuelle il facilite l'extension de types.
- **GPSS**, General Purpose Simulation System. 1972. Un système est construit par transactions passées entre services.
- **Grap**. Par Brian Kernighan et Jon Bentley à Bell Labs. Pour les graphiques de composition.
- **Groovy**. 2003. Langage de scripts OO pour Java.

H

- **Hack**. 2014. Par Facebook. Version statiquement typée de PHP.
- **Halide**. 2012. Par le MIT, langage de traitement d'image à la syntaxe concise.
- **Hal/S**. 1968. Langage de programmation en temps réel pour l'aéronautique.
- **HAScript**, Host Access Script par IBM. Syntaxe XML pour interagir en ligne de commande, sur JVM.
- **Haskell**. 1990. langage fonctionnel. Haskell 98 suivra. En 2002 version d'un langage fonctionnel paresseux. Compilateur.
- **Haxe**. 2006. Compile en JS, C++, PHP.
- **Heron**. Java-like, OO et fonctionnel.
- **HLA**, High Level Assembly. Assembleur avec constructs des langages de haut niveau.
- **Hobbes**. 2017. Morgan Stanley (Banque). Orienté vers le pattern-matching et le parsing, l'interpréteur JIT peut s'intégrer à un programme C++.
- **Hugo**. 1995. Pour des fictions interactives.
- **HyperTalk**. 1987. Par Dan Winkler à Apple. Procédural composé de cartes à relier et assembler. Hypernext et Supercard sont des outils inspiré par Hypercard.

I

- **IAL**, International Algebraic Language**.** 1958. Renommé Algol.
- **ICI**. 1988. C-like interprété avec garbage collector et modèle de données dynamique pour le scripting.
- **Icon**. 1977-79. C et Pascal-like, pour le traitement de chaînes, est orienté buts. Suivi par Unicon.
- **IDL**, Interactive Data Language. 1977. Une langage descriptif inspiré de Fortran et C utilisé dans les sciences.
- **IMP**. 1970. Algol-like. Système, syntaxe extensible. Au contraire d'Edinburgh IMP, sa syntaxe s'écarte de celle d'ALGOL.
- **Inform**. 1993. Langage et système de conception de fiction interactive. Suivi par Inform 6 (1996) et Inform 7 (2006) basé sur sur le langage naturel.
- **INTERCAL**. 1972. Pour l'histoire, un langage parodique pour se moquer de la prolifération d'étranges constructs dans les langages.
- **Io**. 2002. Basé sur des prototypes, inspiré par Smalltalk.
- **IPL**, Information Processing Langage. 1956. Le premier dans le traitement de listes, mais remplacé par Lisp.
- **ISWIM**, If you See What I Mean. 1966. Non implémenté, il a inspiré les langages fonctionnels.

J

- **J**. 1990. Langage mathématique et d'analyse de données, dérivé de APL.
- **JADE**. 1996. Pascal-like, dédié à l'utilisation de base de données comme Delphi.
- **Jal**, Just Another Language. 2003. Langage Pascal-like pour micro-controleurs.
- **Janus**. 1982. Par Caltech. Calcul réversible.
- **Janus**. 1990. Concurrent, contraint avec des arguments à deux aspects, d'où le nom. Prédécesseur de Toontalk.
- **Java**. 1995. James Gosling and Sun. Fonctionnant sur machine virtuelle et donc portable, il est dérivé de C avec objets. Chaque classe est stockée dans un fichier.
- **JavaFX Script**. 2005. Scripting pour l'interface JavaFX. Abandonné par Oracle, mais forké (relancé) sous le nom de Visage.

- **JavaScript**. 1995. Par Brendan Eich. Dynamique, C-like, inspiré par Self pour les prototypes. Scripting pour le navigateur, GUI, documents, ou sur le serveur.
- **JCL**, Job Control Language. Pour mainframes IBM.
- **Jif**. 2001. Cornell. Java avec des contrôles sur l'accès aux informations.
- **Join Java**. 2000. Version de Java étendue avec les joint-pattern.
- **Joss**, JOHNNIAC Open Shop System. 1963. Pour le temps partagé, prédécesseur of MUMPS.
- **Joule**. 1996. Concurrent et distribué, précurseur de E.
- **JOVIAL**. Jules Own Version of the International Algorithmic Langage. 1960. ALGOL-like pour les systèmes embarqués (IAL était le premier nom d'ALGOL).
- **Joy**. 2001. Fonctionnel.
- **JScript**. 1996. Dialecte de ECMAScript par Microsoft. Similaire à JavaScript mais sans le mot Java pour des raison de nom de marque. Abandonné depuis IE 10.
- **Julia**. 2010. Pour la programmation scientifique, très rapide sur LLVM. Parallèle, distribué. Un programme peut modifier son propre code.

K

- **K**. 1993. Propriétaire, pour le traitement de chaînes, dérivé d'APL. Kona est un interpréteur open source.
- **Kaleidoscope**. 1990. Langage impératif, OO, à contraintes. A évolué de Smalltalk-like à ALGOL-like.
- **Korn shell**. 1983. Scripting en ligne de commande compatible avec Bourne.
- **Kotlin**. 2012. Par JetBrains. Langage statiquement type pour la JVM ou JavaScript. Essaie de combiner toutes les théories sur les langages.

L

- **LabView**. 1986. Langage visuel par National Instruments, destiné à la commande d'appareils.
- **Ladder Logic**. Langage visuel pour les circuits logiques programmables utilisés dans le contrôle industriel.
- **Lagoona**. Experimental, pour la programmation orientée composants communiquants par message.
- **Lava**. 2001. OO, interprété. Veut construire un programme à partir d'une arborescence plutpot qu'un éditeur de texte.
- **Leda**. 1994. Son but est de combiner le style impératif, fonctionnel et logique.
- **Lfyre**. 2005. Extensible.
- **Limbo**. 1995. Par Rob Pike & Bell Labs. Langage concurrent (basé sur CSP), pour les applications distribuées sur l'OS Inferno. Successeur de Alef et Newsqueak.
- **LINC 4DL**. Prédécesseur de EAE et AB Suite, deux générateurs de code de Unisys.
- **Lingo**. Plusieurs langages ont ce nom: Macromedia Lingo, Lingo Allegro, Linn Lingo, Lindo Lingo.
- **Lisaac**. 2003. Langage OO basé sur des prototypes, pour la construction de système d'exploitation.
- **Lisp**, LISt Processing. 1958 par John McCarthy. Extensible, composé d'arborescences et de parenthèses, a influencé de nombreux languages.
- **LLJS**, Low Level JavaScript. 2012. Par Mozilla, dialecte typé de JavaScript plus proche de C, et compilé en JS. Remplacé par Asm.js.
- **LLVM bitcode**. 2004. Langage intermédiaire pour compilateurs et machines virtuelles.
- **Lobster**. 2013. Programmation de jeux en 3D avec OpenGL en arrière-plan.
- **Loci**. 2014. Proche de C++ en plus simple, avec un garbage collector, compile sur LLVM.

- **Logo**. 1966-68. Lisp sans parenthèses. Apprendre la programmation en déplaçant une souris graphique.
- **Lua**, lune en portugais. 1993. Langage de scripts à l'origine extension à C, devenu indépendant.
- **Lucid**. 1976. Modèle de programmation différent proche de la prog. réactive, où les instructions sont des équations dont les variables sont des processeurs interconnectés.
- **Lustre**. 1991. Pour des systèmes réactifs.
- **LYaPAS**. 1964. Par l'Académie des Sciences de Russie. Langage logique pour la représentation d'algorithmes synthétiques. Extension d'APL.

M

- **M**. 2008. Langage de modélisation de Microsoft pour la plateforme Oslo.
- **M#.** 2014. Par Microsoft, générateur de code similaire à JavaFX script, décrit un site et compile en C# et ASP.NET.
- **M**. Voir MUMPS.
- **MAD**. Voir IAL, ALGOL.
- **Mary**. 1970+. Similaire à ALGOL 68, orienté programmation de bas niveau.
- **Mathematica**. 1988. Langage de programmation qui utilise la notation algébrique pour les expressions.
- **MATLAB**.1975-78 par Cleve Moler. Le langage scientifique et mathématique a évolué vers des applications plus diverses.
- **Mercury**. 1995. Langage de programmation fonctionnel et logique. Porté sur C, Java, .NET.
- **Mesa**. 1970+. Palo Alto. Pascal-like, modulaire, a inspiré Modula-2 et Java. Remplacé par Cedar.
- **MetaL**. 2001. Générateur de code basé sur XML.
- **MIMIC**. 1964. Expression-orienté, simulation pour la conception industrielle.
- **Mirah**. 2011. Similaire à Ruby mais tourne sur la machine virtuelle Java et utilise son API. Peut servir à des applications Android.
- **Miranda**. 1985 par David Turner. Langage fonctionnel, a inspiré Haskell.
- **Miva Script**. 1996. Propriétaire, pour la création de site d'ecommerce.
- **Mixal**, Mix Assembly Language. Pour l'ordinateur Mix historique de Donald Knuth.
- **ML**. 1973. Université d'Edinburgh. Fonctionnel, inspiré par ISWIM.
- **Moby**. 2002. Experimental, pour combiner le foncitonnel avec la concurrence et l'OO.
- **Modula**. 1970+ par Niklaus Wirth. Pascal (du même auteur) avec modules.
- **Modula-2**. 1980 par Niklaus Wirth. Modula avec coroutines, se veut langage système et d'applications.
- **Modula-3**. 1989 par DEC et Olivetti. Modula 2 avec généricité, multithreading, exceptions, garbage collector. A influencé d'autres langages sans être lui-même adopté.
- **Mondrian**. Version OO de Haskell.
- **Mortran**. Dérivé de Fortran avec des différences syntaxiques.
- **Moto**. 1999. C-like imbriqué dans les documents tels HTML.
- **MSIL**. Voir CIL.
- **MUMPS**. 1967. Massachusetts General Hospital Utility Multi-Programming System. Langage orienté bases de données.

N

- **Napier88**. Du nom de John Napier. 1989. Langage persistant expérimental.
- **Neko**. 2005. Compile en bytecode pour sa propre machine virtuelle.
- **Nemerle**. Du nom d'un personnage de fiction. 2003. Functionnel, OO et impératif. Sur .NET.

- **Nesl**. 1993. A Carnegie Mellon. Parallèle, fonctionnel et orienté tableaux.
- **NetRexx**. 1996. Par Mike Cowlishaw. Version du langage de script Rexx utilisé chez IBM, porté sur la JVM et premier à l'être.
- **Newspeak**. Du nom du langage imaginé par Orwell (novalangue en français). 2006. Doté de classes imbriquées.
- **Newsqueak**. 1989. Par Rob Pike chez Bell Labs, qui plus tard va créer Go, un autre langage concurrent. Dérivé de Squeak, il facilite la création de GUI. A inspiré Alef, Limbo et Go.
- **Ngl**, aNGeL (ange). 2004. Extension de J, doté d'une notation mathématique.
- **Nial**, Nested Interactive Array Language. 1981. Notation de programmation fonctionnelle pour tableau, appliqué à l'IA.
- **Nice**. 2003. OO avec des capacité étendues et un contrôle plus strict contre les erreurs.
- **Nickle**. 2001. Orienté numérique pour l'algorithmique.
- **Nim** (ex-Nimrod). 2010. Python-like pour la programmation système. Méta-programmation, OO, compile en C, JS ou binaire.
- **Nit**. 2009. Statiquement typé et orienté objets, proche de Ruby.
- **Noop**. 2009. Par Google. Langage expérimental dérivé de Java pour encourager les bonnes pratiques de programmation et décourager les mauvaises. Compile du bytecode pour la JVM.
- **Nu**. 2007. Lisp-like, OO et interprété.

O

- **o:Xml**. 2002. OO avec une syntaxe XML-like.
- **Oberon**, nom d'une lune d'Uranus. 1986 par by Niklaus Wirth. Réflexif et extensible, dérivé de Modula-2.
- **Objective-C**. 1983. C plus objets de Smalltalk, utilisé essentiellement sur les appareils d'Apple après avoir été popularisé sur les machines NeXT en 1988.
- **OCaml**, Objective Caml. 1996 par l'INRIA. Dérivé de ML, fonctionnel et impératif, sur une machine virtuelle. Etend Caml.
- **Objective Modula 2**. 2006. Combinaison de Objective-C, Smalltalk et Modula 2.
- **Obliq**. Dérivé d'Oberon pour du traitement distribué.
- **Occam**. 1983. (occam-π). Concurrent basé sur le principe CSP.
- **Octave**. 1988. Interprété pour le calcul numérique.
- **ooc**. 2009. Un dérivé de C, orienté objet, compile en C.
- **Opa**. 2011. Coté serveur ou client, compilé en JavaScript.
- **Opal**, OPtimized Applicative Language. Université de Berlin. Langage fonctionnel algébrique, introduit la notion de monades, alors appelées "commands".
- **OpenEdge ABL**. OpenEdge Advanced Business Language. 1984. Syntaxe proche de l'anglais et **OO**, avec des commandes de gestion de BD.
- **OPL**, Open Programming Langage. 1984. BASIC-like pour l'OS Symbian.
- **OPS5**, Official Production System 5. Basé sur des règles avec un moteur d'inférence, écrit en BLISS.
- **Orc**. 2004 par l'Université du Texas. Langage concurrent et distribué qui fonctionne entre sites. Peut s'utiliser pour des scripts Web.
- **Oz**. 1991. Multi-paradigme: impératif, fonctionnel, logique, à contraintes, OO, distribué et concurrent.

P

- **Pascal**, du nom du mathématicien français. 1970. Par Nicklaus Wirth. Sa syntaxe encourage la programmation structurée.
- **PBASIC**. 1992. Version de BASIC pour microcontroleurs.

- **Perl**. 1987 par Larry Wall. Interprété, dynamique, sa syntaxe obscure fait qu'on le surnomme "langage en lecture seule".
- **PHP**, Personal Home Page Hypertext Processor. 1995 by Rasmus Lerdof. PHP 5 en 2004. PHP 6 en 2007. Scripting coté serveur et générateur de pages.
- **Pico**. 1997. Minimaliste, pour apprendre les concepts de la programmation aux étudiants d'autres disciplines.
- **Picture**. 2015 par la MIT. Langage probabilistique pour la reconnaissance d'images.
- **Pike**. 1994. C-like, interprété, dynamique, OO avec des types de données avancées. Peut s'utiliser pour apprendre C.
- **PILOT**, Programmed Instruction, Learning, or Teaching. 1968. Premiers pas en apprentissage par ordinateur.
- **PL-11**. 1971. OO for PDP 11.
- **PL/0**. 1976. Par Niklaus Wirth, version simplifiée de Pascal pour l'éducation.
- **PL/B**, Programming Language for Business, anciennement DATABUS. 1970+. Alternative à COBOL, compilé en bytecode.
- **PL/C**. 1970+. Sous ensemble de PL/1 pour apprendre la programmation.
- **PL/I**. 1964 par IBM. Programming Langage One. Procédural pour le traitement numérique et industriel.
- **PL/M**, Programming Language for Microcomputers. 1972 par Gary Kildall. Langage de haut-niveau pour les microprocesseurs d'Intel.
- **Planner**. 1969. Pour ajouter le traitement logique à un langage procédural. Des sous-ensembles ont été implémentés.
- **Plankalkül**. Trad: système de planification formel. 1948 par Konrad Zuse.
- **POP-2**. 1970. succède à POP-1 et suivi de POP-11. Fonctionnel, inspiré de Lisp et ALGOL 60.
- **POV-Ray**. Langage graphique du logiciel de lancer de rayons.
- **Processing**. 2001. C-like, pour la création d'images et d'animations interactives.
- **Prograph**. 1983 par Acadia University. Langage visuel a icônes.
- **Prolog**. 1972 par Alain Colmerauer. Langage d'inférences logiques déclaratif.
- **Proteus**, PROcessor for TExt Easy to USe. 1988. Fonctionnel pour le traitement de texte.
- **P-TAC**. 1989. Langage parallèle.
- **Pure**. 2008. Langage fonctionnel interprété (par LLVM) basé sur la réécriture des termes.
- **Purescript**. 2011. Langage fonctionnel statiquement typé, compile en JavaScript.
- **Python**. 1991 par Guido van Rossum. Langage de scripts interprété ou compilé.

Q

- **Q**. 2003. Dérivé d'APL, traitement de tableaux, pour applications financières.
- **QuakeC**. Version de C pour le jeu Quake.
- **QPL**, Quantum Programming Languages. Ensemble de langages de programmations pour les ordinateurs quantum.
- **QML**, **Qt Modeling Language**. 2009. Langage déclaratif d'interface utilisateur, similaire à JavaFX, pour Qt.
- **Quorum**. 2012. Langage orienté objet et extensible qui se veut facile à lire pour les débutants. Compilé pour la JVM.

R

- **R**. 1993. Langage et environnement pour calculs et graphiques statistiques, dérivé du langage S, et proche de Scheme.
- **R++.** 1998 par Bell Labs. Version de C++ basé sur des règles.
- **Racket**. 1994. Lisp-like conçu pour être développé par le programmeur.

- **Ratfiv**. Jeu de mots sur Ratfor (for=four=4) et Raf five (5). 1980+.Version de Ratfor avec des fonctions de C.
- **Ratfor**. 1975 par Brian Kernighan. Prépocesseur pour Fortran.
- **rc**. 1989 par Bell Labs. Langage de commandes pour Plan9, porté ensuite sur Unix.
- **Rebol**, Relative Expression Based Object Language. 1997. Langage dynamique avec de nombreux types prédéfinis. La version 3.0 devient open source en 2012.
- **Red**. 2011. Similaire à Rebol, mais compilé et open source depuis le début.
- **Refal**, REcursive Functions Algorithmic Language. 1968. Fonctionnel, pattern-matching orienté buts. La structure de donnée de base est le but.
- **RPG**, Report Program Generator. 1959 par IBM. Outil de requêtes étendu en langage de programmation proche de la conduite par évènements. Les versions principales sont RPG II, RPG III, RPG/400, RPG IV.
- RPL, ROM-based Procedural Language. 1984 par HP. Langage de calculatrices similaire à Forth.
- **Rexx**, REstructured eXtended eXecutor. 1979 par Mike Cowlishaw. Conçu pour le scripting sur l'OS IBM puis porté sur d'autres plateformes.
- **RLaB**. 2000. Alternative à MATHLAB avec un syntaxe plus simple.
- **RSL**, Robot Scripting Langage. 2002 par Microsoft. Pour le jeu Robot Battle.
- **Ruby**. 1995 par Yukihiro Matsumoto. Suit un "principe de la moindre surprise", chaque chose doit être intuitive. Multi-paradigmes, orienté object pour le scripting, les applications en ligne.
- **Rust**, roux en vieil anglais. 2006. Langage concurrent par Mozilla Labs inspiré de C et LLJS et amélioré pour être plus sûr. Alternative à Go basée sur LLVM.

S

- **S**. 1976. Bell Laboratories, John Chamber. Langage statistique. Remplacé par R.
- **S-algol**, St Andrews Algol. 1979 par l'Université de St-Andrews (Ecosse). Version simplifiée et améliorée d'ALGOL-60.
- **Sail**, Stanford Artificial Intelligence Langage. 1970. Basé sur une mémoire associative d'enregistrements, des évènements et contextes.
- **SAM76**. 1970+. Langage de macros pour CP/M.
- **SAS**. 1972. Pour des rapports et analyses statistiques. Produit des documents HTML ou PDF.
- **SASL**, St. Andrews Standard Language. 1972. Implémentation de ISWIM.
- **Sather**, nom de la tour Jane Sather. 1990 par Berkeley. Basé sur Eiffel mais a évoluté et s'est doté d'un coté fonctionnel, classes, itérateurs.
- **Sawzall**. 2003. Par Rob Pike à Google pour gérer les données de fonctionnement de ses serveurs.
- **Scala**. 2003 par Martin Odersky. Pour écrire du code concis compatible Java. Implémente de nombreux nouveaux concepts.
- **Scheme**. 1975 par le MIT. Dialect de Lisp et ALGOL avec un design simple.
- **Scratch**. Langage éducatif conçu par le MIT consistant en blocs à assembler. Le même principe a été utilisé pour la librairie Java OpenBlocks.
- **Scriptol**. 2001. Orienté objets et conçu pour être intuitif et améliorer la productivité, il intègre la programmation réactive et impératives. Interprété ou compilé en JavaScript, C++, PHP.
- **Sed**, Stream EDitor. 1974 par Bell Labs. Traitement de texte.
- **Seed7**. 2005. Similaire à Pascal et ADA, syntaxe extensible.
- **Self**. 1993. OO avec prototypes comme Smalltalk. Utilise un JIT.
- **SETL**, SET Language. 1967-1969. Langage d'ensembles, a inspiré ABC, prédécesseur de Python, lui a transmis les tuples.

- **Short Code**. 1949. Précurseur des langages de programmation.
- **Simit**. 2016. Par le MIT pour remplacer Matlab et opérer sur des graphes ou des simulations physiques, similaire à Julia avec des structures de graphes.
- **Simula**. 1962. Sur-ensemble d'ALGOL. Simula 67 introduit les classes et l'héritage ainsi que les méthodes virtuelles et coroutines.
- **SISAL**. Streams and Iteration in a Single Assignment Langage. 1983. Pascal-Like, fonctionnel, pour le calcul numérique.
- **Slip**, Symmetric LIst Processor. Traitement de liste pour Fortran et autres langages.
- **Smalltalk**. 1972 par Alan Kay et autres. OO, dynamique et reflexif, a inspiré d'autres langages comme Objective-C.
- **SNOBOL**. 1962. Snobol 3 (1965), 4 (1966). Basé surtout sur la notion de pattern-matching. **SPITBOL** (SPeedy ImplemenTation of snobOL) est une version compilée de SNOBOL pour IBM 360.
- **SOAP**, Symbolic Optimal Assembly Program. 1957. Langage d'assemblage pour IBM 650.
- **Snowball**, imitation de SNOBOL. 2001. Traitement de chaînes et linguistique, compile en C ou Java.
- **SPARK**. 1983. ADA-like, pour des systèmes de sécurité.
- **SP/k**. 1974. Sous-ensemble de PL/I, utilisé pour l'enseignement.
- **SPL**, Shakespeare Programming Language. 1993. Humoristique.
- **Squeak**. 1996. Dialecte de Smalltalk.
- **Squirrel**. 2003. C-like pour scripts embarqués dans un projet C ou C++.
- **SR**, Synchronizing Resources. Ancien langage concurrent.
- **S/SL**, Syntax/Semantic Language. 1980. Université de Toronto. Pour générateurs de code.
- **Standard ML**. 1990. Dérivé de ML, fonctionnel, avec inférence de type.
- **Subtext**. 2001. Expérimental, visuel.
- **SuperCollider**. 1996. Interprété, OO pour synthèse audio en temps réel et composition algorithmique.
- **SuperX++.** 2001. Langage XML.
- **Swift**. 2014. Par Apple pour ses systèmes d'exploitation dans le but de remplacer Objective-C par un langage plus sûr et plus rapide. C'est aussi ne nom d'un autre langage.
- **Synergy/DBL**. Langage de l'environnement de développement Synergy/DE.

T

- **T**. 1980+. Une version de Scheme.
- **TACL**, Tandem Advanced Command Langage. 1974. Langage de scripting utilisé par Hewlett-Packard sur des serveurs.
- **TACPOL**, Tactical Procedure Oriented Language. Avant 1977. Implémentation de PL/I, était utilisée par l'US army.
- **TADS**, Text Adventure Development System. 1988. Un langage pour réaliser des jeux.
- **TAL**, Transaction Application Langage. Langage système, croisement entre C et Pascal utilisé sur les ordinateurs Tandem.
- **Tcl**, Tool Command Langage. 1988 par John Ousterhout. Tk est un toolkit graphique associé.
- **TELCOMP**. 1965. Dérivé de JOSS, langage conversationnel utilisé sur les ordinateurs PDP jusqu'en 1974. A influencé Mumps.
- **Tempo**. Déclaratif et logique, concurrent.
- **Titanium**. 2005. Dialecte de Java parallèle pour calcul scientific.
- **TI-BASIC**. 1996. Langage pour calculatrices de Texas Instrument, proche de BASIC.
- **TOM**. 1990+. OO avec des classes dynamiques qui évoluent.
- **TRAC**, Text Reckoning And Compiling. 1960+. Orienté macros pour traitement de texte.
- **Transcript**.

- **TTCN-3**, Testing and Test Control Notation. Pour le contrôle de système de communication.
- **Turing**. 1982. Proche de Pascal, dérivé de Euclid.
- **TUTOR**. 1965. Langage de programmation de CAI.
- **TypeScript**. 2012. Sur-ensemble de JavaScript par Microsoft, avec des types statiques, des classes et des modules. Compilé en JavaScript. Open source sous licence Apache.
- **TXL**, Turing eXtender Language. 1988. Dérivé de Turing ci-dessus.

U

- **Ubercode**. 2005. Commercial, croisement entre Eiffel et BASIC.
- **UNCOL**, Universal Computer Oriented Language. 1958 par Melvin E. Conway. Premier concept d'un langage intermédiaire pour une machine virtuelle.
- **Unicon**. Unified Extended Dialect of Icon. 1996. Dérivé de Icon avec OO, accès au système.
- **UnrealScript**. 1998. Scripting pour le moteur de jeux Unreal.
- **UrbiScript**. 2003. Langage de programmation de robots.
- **UML**, Unified Modeling Langage. 1994 par Rational Software. Langage de programmation visuel, standard ISO.

V

- **Verilog HDL**, Verilog Hardware Description Language. 1990. Un langage de description de matériel.
- **VHDL**, VHSIC Hardware Description Language. 1980+.
- **VDS**. Visual DialogScript. 1995. Interprété pour réaliser des interfaces sur Windows.
- **Visual Basic**. 1991 par Microsoft. Version améliorée et OO de BASIC.
- **Visual Basic.NET**. 2001. Successeur de Visual Basic 6.0, fonctionne sur .NET.
- **VBScript**, Visual Basic Script Edition.1996 par Microsoft. Version allégée et interprétée de Visual Basic pour Windows.
- **VTL, VTL-2**, Very Tiny Language. 1976. Langage minimal stocké dans la ROM de moins d'1 K octet du Altair 680B et 8800.

W

- **Water**. Pour le prototype de services Web XML.
- **Whitespace**. 2003. En réalité une plaisanterie, un langage de programmation exotique, avec un vrai interpréteur.
- **Winbatch**. 1991. Langage de scripts pour Windows.
- **Wiring**. 2003. Plateforme de développement et langage dérivé de C adapté à la programmation de composants électroniques.
- **WLanguage**. 1992. Langage de l'outil de développement Windev influencé par BASIC et Pascal.
- **Wolfram**. 2013. Basé vers le traitement de la connaissance, il réunit plusieurs paradigmes pour obtenir la plus grande flexibilité dans le traitement automatique.
- **Wyvern**, nom d'une créature mythique. 2014 par Carnegie Mellon. Interprété et compilé pour des applications sécurisées.

X

- **X10**. 2004. Par IBM pour le projet PERCS. Vise la performance sur les grands sites avec le parallélisme structuré.

- **XOTcl**, Extended Object Tcl. Version de TCL orientée objet avec mixins.
- **XPL**. 1968. Dérivé de PL/I, pour l'écriture de compilateurs.
- **XL**, eXtensible Language. 2000. Implémente la programmation par concept. Sa syntaxe est reconfigurable par programme.
- **Xtend**. 2011 par la fondation Eclipse pour faciliter Java, il apporte des améliorations, comme la suppression des point-virgules, un switch puissant comme en Scriptol. Produit du code Java

Y

- **YAFL**. 1990+. Une version de Modula-2.
- **Yorick**. 1996. Langage interprété pour le calcul scientifique et la simulation.

Z

- **Z** Notation. 1977. Spécification visuelle de programmes comme UML.
- **Zig**. 2016. Langage humoristique conçu comme une parodie de Rust.
- **ZPL**, Z-level Programming Language. 1993. Parallèle pour le calcul scientifique et technique.
- **ZOPL**, Version Z, Our Programming Language. 1970+. Proche de C et Pascal, pour mainframes.

Langages à balises et formats de données

- **CFML**, ColdFusion Markup Langage. 1995 par Adobe. Langage de script pour les applications Web fonctionnant sur JVM et .NET.
- **EmotionML**. 2013. Un dialecte XML pour représenter les émotions, par le W3C.
- **HTML**, HyperText Markup Langage. 1991 par Tim Berners-Lee. Basé sur SGML.
- **JSON Patch**. Standard proposé par l'IETF pour une série d'actions sur un document JSON.
- **PostScript**. 1982 par Adobe. Langage de graphismes vectoriel, souvant appliqué à l'impression de documents.
- **Protocol Buffers**. 2008 par Google. Format de sérialisation de document en fichier de texte, similaire à JSON. FlatBuffer est une version rapide binaire.
- **RDF**, Resource Description Framework. 1999 par le W3C. Format de stockage de l'information avec méta-données.
- **SGML**, Standard Generalized Markup Language. 1969 par IBM. Précurseur d'XML pour le stockage d'information lisible par l'homme.
- **SVG**, Scalable Vector Graphic. 2001 par le W3C. Format vectoriel basé sur XML pour les graphismes 2D, supporté par les navigateurs.
- **Tex**. Format de texte.
- **XAML**. eXtensible Application Markup Langage.
- **XBL**. eXtensible Bindings Langage. Pour créer des composants de langages XML.
- **Xforms.** Interface utilisateur interactive graphique pour le Web.
- **XHTML**. XML HTML.
- **XML**. eXtensible Markup Langage.
- **XUL**. XML-based User interface Langage.

Langages de requêtes et de bases de données

- **Andl**. 2015. Un nouveau langage de requête pour base de données différent de SQL. Il veut stocker plus de détails dans la base et moins dans le langage de requêtes. Implémenté sur PostgreSQL.
- **AQL**, Aerospike Query Language. 2012. Langage simple et plus développé que SQL pour la BD Aerospike.
- **Aubit-4GL**. Voir Informix.
- **D**. 1994. Langage relationel abstrait, implémenté dans D4 réalisé en C#. Tutorial D est une version d'enseignement.
- **Dataflex**. 1980. langage de programmation de base de données.
- **dBase**. 1979. Premier langage de programmation de base de données sur micro-ordinateur (Apple II et IBM-PC).
- **GraphQL**. 2015. Créé par Facebook pour simplifier les requêtes à la place de SQL. Elle prenne la forme d'un objet JavaScript.
- **Hypertalk**. 1987. Langage de fiches pour Apple.
- **Informix-4GL**. 4GL signifie langage de 4ième génération spécialisé. Informix est spécialisé en base de données et rapport.
- **pl/SQL**. Extension à SQL.
- **SQL**, Structured Query Langage. 1987 par IBM. Langage de requêtes le plus utilisé.
- **Visual Foxpro**. 1984. Dérivé de dBase. Possédé par Microsoft, remplacé par LightSwitch.

What *Is* Code?

A few of those 1,700 languages are still with us.■24 Cobol, for example, a legendary and much-hated, extre

■24 There are a few very big, popular programming languages—and many hundreds of others. Here's an abridged list of important ones, adapted from Wikipedia (which is actually good when it comes to computer things), with some of the most interesting languages noted.

Source : Bloomberg Businessweek june 15 -June 28, 2015

y verbose language that was intimately linked to the "year 2000" problem. As computer ▶

67

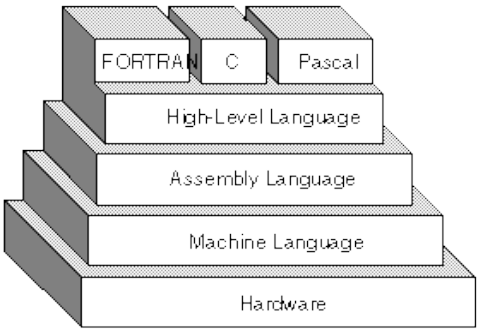
Source : Bloomberg Businessweek june 15 -June 28, 2015

Chapitre3 : Les notions et paradigmes de la programmation

3.1 Définition

Un **langage de programmation** est un vocabulaire et un ensemble de règles grammaticales pour ordonner à un ordinateur ou à un dispositif informatique d'effectuer des tâches spécifiques. Le terme langage de programmation fait généralement référence à des langages de haut niveau, tels que BASIC, C, C ++, COBOL, Java, FORTRAN, Ada et Pascal.

Chaque langage de programmation a un ensemble unique de mots-clés (mots qu'il comprend) et une syntaxe spéciale pour organiser les instructions du programme.



Langage de Haut niveau : Les langages de programmation de haut niveau, bien que simples par rapport aux langages humains, sont plus complexes que les langages que l'ordinateur comprend réellement, appelés langages machine. Chaque type de CPU a son propre langage machine.

Entre les langages machine et les langages de haut niveau se trouvent des langages appelés langages d'assemblage. Les langages d'assemblage sont similaires aux langages machine, mais ils sont beaucoup plus faciles à programmer car ils permettent à un programmeur de substituer des noms aux nombres. Les langages machine se composent uniquement de nombres.

Au-dessus des langues de haut niveau, il y a des langues appelées langues de quatrième génération (généralement abrégées 4GL). Les 4GL sont très éloignés des langages machine et représentent la classe des langages informatiques les plus proches des langages humains.

Conversion en langage machine :
Quelle que soit la langue que vous utilisez, vous devez éventuellement convertir votre programme en langage machine afin que l'ordinateur puisse le comprendre. Il y a deux façons de faire ça:

1) Compilez le programme.
2) Interprétez le programme.

3.2 Règles et vocabulaires

Les *règles de syntaxe*

> Définies par une grammaire formelle, elles régissent les différentes manières dont les éléments du langage peuvent être combinés pour obtenir des programmes[2]. La ponctuation (par exemple l'apposition d'un symbole **;** en fin de ligne d'instruction d'un programme) relève de la syntaxe.

Le *vocabulaire*

Parmi les éléments du langage, le vocabulaire représente l'ensemble des instructions construites d'après des *symboles*. L'instruction peut être mnémotechnique ou uniquement symbolique comme quand elle est représentée par des *symboles d'opérations* tels que des opérateurs arithmétiques (« + » et « - ») ou booléens (&& pour le et logique par exemple). On parle aussi parfois de *mot clé* pour désigner une instruction (par abus de langage car le concept de mot clé ne recouvre pas celui des symboles qui font pourtant eux aussi partie du *vocabulaire*).

La *sémantique*

Les règles de *sémantique* définissent le sens de chacune des phrases qui peuvent être construites dans le langage, en particulier quels seront les effets de la phrase lors de l'exécution du programme. La science l'étudiant est la sémantique des langages de programmation.

L'*alphabet*

L'alphabet des langages de programmation est basé sur les normes courantes comme ASCII, qui comporte les lettres de A à Z sans accent, des chiffres et des symboles, ou Unicode pour la plupart des langages modernes (dans lesquels l'utilisation se limite en général aux chaînes de caractères littérales et aux commentaires, avec quelques exceptions notables comme C♯ qui autorisent également les identifiants unicode).

La plupart des langages de programmation peuvent prévoir des éléments de structure complémentaires, des méthodes procédurales, et des définitions temporaires et variables et des identifiants :

Les *commentaires*

Les *commentaires* sont des textes qui ne seront pas traduits. Ils peuvent être ajoutés dans les programmes pour y laisser des explications. Les commentaires sont délimités par des marques qui diffèrent d'un langage de programmation à l'autre tel que « -- », « /* » ou « // ».

Les *identifiants*

Les éléments constitutifs du programme, tels que les *variables*, les *procédures*, ou les *types* servent à organiser le programme et son fonctionnement. On peut ainsi, par exemple, diviser un programme en fonctions, ou lui donner une structure par objets : ces éléments de structure sont définis par des identifiants ou des procédures par *mot clé* selon le langage.

Une *instruction*

Un ordre donné à un ordinateur.

Une *variable*

Un nom utilisé dans un programme pour faire référence à une donnée manipulée par programme.

Une *constante*

Un nom utilisé pour faire référence à une valeur permanente.

Une expression *littérale*

Une valeur mentionnée en toutes lettres dans le programme.

Un *type*

Chaque donnée a une classification, celle-ci influe sur la plage de valeurs possibles, les opérations qui peuvent être effectuées, et la représentation de la donnée sous forme de bits. Chaque langage de programmation offre une gamme de types primitifs, incorporés dans le langage. Certains langages offrent la possibilité de créer des nouveaux types.

Les types de données primitifs courants sont les nombres entiers, les nombres réels, le booléen, les chaînes de caractères et les pointeurs.

Plus précisément, le type booléen est un type qui n'a que deux valeurs, vrai et faux, tandis que le type pointeur : une référence à une donnée, qui se trouve quelque part en mémoire.

Une structure de données

Une manière caractéristique d'organiser un ensemble de données en mémoire, qui influe sur les algorithmes utilisés pour les manipuler. Les structures courantes sont les tableaux, les enregistrements, les listes, les piles, les files et les arbres.

Une *déclaration*

Une phrase de programme qui sert à renseigner au traducteur (compilateur, interpréteur...) les noms et les caractéristiques des éléments du programme tels que des variables, des procédures, de types...

Des vérifications sont effectuées au moment de la compilation, ou au moment de l'exécution du programme, pour assurer que les opérations du programme sont possibles avec les types de données qui sont utilisés. Dans un langage fortement typé, chaque élément du programme a un type unique, connu et vérifié au moment de la compilation, ce qui permet de déceler des erreurs avant d'exécuter le programme.

Les *procédures, fonctions, méthodes*

Divers langages de programmation offrent la possibilité d'isoler un fragment de programme, et d'en faire une opération générale, paramétrable, susceptible d'être utilisée de façon répétée. Ces fragments sont appelés procédures, fonctions ou méthodes, selon le paradigme.

Les *modules*

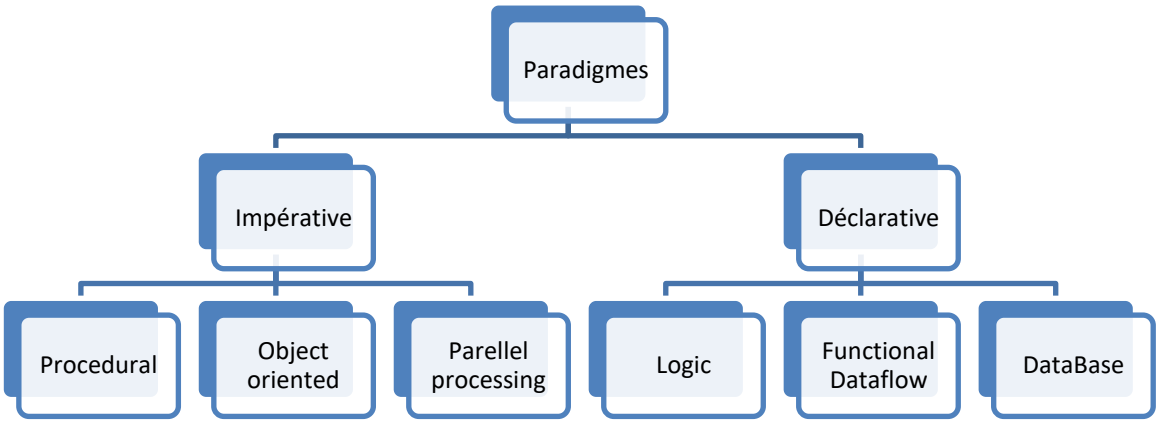
Les langages de programmation peuvent également offrir la possibilité de découper un programme en plusieurs pièces appelées modules, chacune ayant un rôle déterminé, puis de combiner les pièces.

Les notions de procédure et de module sont destinées à faciliter la création de programmes complexes et volumineux en assistant la prise en charge de cette complexité. Ces fonctions permettent en particulier la modularité et l'abstraction.

3.3 Paradigmes

Un *paradigme* est une façon d'approcher la programmation. Chaque paradigme amène sa philosophie de la programmation ; une fois qu'une solution a été imaginée par un programmeur selon un certain paradigme, un langage de programmation qui suit ce paradigme permettra de l'exprimer. Impératif, déclaratif, fonctionnel, logique, orienté objet, concurrent, visuel, événementiel, et basé web sont des paradigmes de programmation. Chaque langage de programmation reflète un ou plusieurs paradigmes, apportant un ensemble de notions qui peuvent être utilisées pour exprimer une solution à un problème de programmation. Au cours de l'histoire, les scientifiques et les programmeurs ont identifié les avantages et les limitations d'un style de programmation et apporté de nouveaux styles. La plupart des langages de programmation contemporains de 2013 permettent d'adopter plusieurs paradigmes de programmation à condition que ceux-ci soient compatibles.

On retrouve dans la littérature ce schéma :



Impératif (ou procédural)

Le paradigme *impératif* ou *procédural* est basé sur le principe de l'exécution étape par étape des instructions tout comme on réalise une recette de cuisine. Il est basé sur le principe de la machine de Von Neumann. Un ensemble d'instructions de contrôle de flux d'exécution permet de contrôler l'ordre dans lequel sont exécutées les instructions qui décrivent les étapes. Le C, le Pascal, le Fortran et le COBOL sont des exemples de langage de programmation qui implémentent le paradigme impératif.

Les langages impératifs sont classés par les programmations suivantes :

- Programmation structurée

- Programmation procédurale : on peut citer le C et Pascal.
- Programmation orientée objet : on peut citer Java, Eiffel et C++.

Déclaratif

Il y a essentiellement deux paradigmes déclaratifs ; ce sont le paradigme fonctionnel et le paradigme logique. En paradigme fonctionnel le programme décrit des fonctions mathématiques. En paradigme logique il décrit des prédicats : c'est-à-dire des déclarations qui, une fois instanciées, peuvent être vraies ou fausses ou ne pas recevoir de valeur de vérité (quand l'évaluation du prédicat ne se termine pas). Dans un modèle d'implantation, une machine abstraite effectue les opérations nécessaires pour calculer le résultat de chaque fonction ou chaque prédicat. Dans ces paradigmes une variable n'est pas modifiée par affectation. Une des caractéristiques principales est la transparence référentielle, qui fait qu'une expression peut être remplacée par son résultat sans changer le comportement du programme.

Les langages déclaratifs sont classés par les programmations suivantes :

- Programmation descriptive
- Programmation fonctionnelle
- Programmation logique
- Programmation logique
- Programmation par contraintes

Fonctionnel

Le paradigme *fonctionnel* a pour principe l'évaluation de formules, afin d'utiliser le résultat pour d'autres calculs ; il s'appuie sur la récursivité et il a pour modèle le lambda-calcul, plus précisément la réduction en forme normale de tête. Tous les calculs évaluent des expressions ou font appel à des fonctions. Pour simplifier, le résultat d'un calcul sert pour le calcul ou les calculs qui ont besoin de son résultat jusqu'à ce que la fonction qui produit le résultat du programme ait été évaluée. Le paradigme fonctionnel a été introduit par les langages Lisp et ISWIM ainsi qu'en ce qui concerne les fonctions récursives par Algol 60, dans les années 1960. Des langages tels que Ruby et Scala supportent plusieurs paradigmes dont le paradigme fonctionnel, tandis qu'Haskell ne supporte que le paradigme fonctionnel et OCaml privilégie le paradigme fonctionnel qu'il partage avec le paradigme objet et une petite dose d'impératif.

Logique

Le paradigme *logique* vise à répondre à une question par des recherches dans un ensemble, en utilisant des axiomes, des requêtes et des règles de déduction. L'exécution d'un programme est une cascade de recherches de faits dans un ensemble, en invoquant des règles de déduction. Les données obtenues, peuvent être associées à un autre ensemble de règles et peuvent alors être utilisées dans le cadre d'une autre recherche. L'exécution du programme se fait par évaluation, le système effectue une recherche de toutes les affirmations qui, par déduction, correspondent à au moins un élément de l'ensemble. Le programmeur exprime les règles, et le système pilote le processus[13]. Le paradigme logique a été introduit par le langage Prolog en 1970.

Orienté objet

Le paradigme programmation orienté objet (POO) est destiné à faciliter le découpage d'un grand programme en plusieurs modules isolés les uns des autres. Il introduit les notions d'objet et d'héritage. Un objet contient les variables et les fonctions en rapport avec un sujet. Les variables peuvent être *privées*, c'est-à-dire qu'elles peuvent être manipulées uniquement par l'objet qui les contient. Un objet contient implicitement les variables et les fonctions de ses ancêtres, et cet *héritage* aide à réutiliser du code. Le paradigme orienté objet permet d'associer fortement les données avec les procédures. Il a été introduit par le langage Simula dans les années 1960, et est devenu populaire dans les années 1980, quand l'augmentation de la puissance de calcul des ordinateurs a permis d'exécuter des grands programmes. Divers langages de programmation ont été enrichis en vue de permettre la programmation orientée objet ; c'est le cas de C++ (dérivé du langage C), Simula, Smalltalk, Swift et Java sont des langages de programmation en paradigme orienté objet.

Le POO peut être rangé en sous-division suivante :

- Programmation orientée prototype
- Programmation orientée classe
- Programmation orientée composant

Concurrent

En paradigme concurrent un programme peut effectuer plusieurs tâches en même temps. Ce paradigme introduit les notions de thread, d'attente active et d'appel de fonction à distance. Ces notions ont été introduites dans les années 1980 lorsque, à la suite de l'évolution technologique, un ordinateur est devenu une machine comportant plusieurs processeurs et capable d'effectuer plusieurs tâches simultanément. Les langages de programmation contemporains de 2013 tels que C++ et Java sont adaptés aux microprocesseurs multi-cœur et permettent de créer et manipuler des threads. Plus récemment, on a vu apparaître des langages intégralement orientés vers la gestion de la concurrence, comme le langage Go.

Visuel

Dans la grande majorité des langages de programmation, le code source est un texte, ce qui rend difficile l'expression des objets bidimensionnels. Un langage de programmation tel que Delphi ou C# permet de manipuler des objets par glisser-déposer et le dessin ainsi obtenu est ensuite traduit en une représentation textuelle orientée objet et événementielle. Le paradigme visuel a été introduit à la fin des années 1980 par Alan Kay dans le langage Smalltalk, dans le but de faciliter la programmation des interfaces graphiques.

Événementiel

Alors qu'un programme interactif pose une question et effectue des actions en fonction de la réponse, en style événementiel le programme n'attend rien et est exécuté lorsque quelque chose s'est passé. Par exemple, l'utilisateur déplace la souris ou presse sur un bouton. Dans ce paradigme, la programmation consiste à décrire les actions à prendre en réponse aux événements. Et une action peut en cascade déclencher une autre action correspondant à un autre événement. Le paradigme événementiel a été introduit par le langage Simula dans les années 1970. Il est devenu populaire à la suite de l'avènement des interfaces graphiques et des applications web.

Basé web

Avec l'avènement de l'Internet dans les années 1990, les données, les images ainsi que le code s'échangent entre ordinateurs. Si un résultat est demandé à un ordinateur, celui-ci peut exécuter le programme nécessaire, et envoyer le résultat. Il peut également envoyer le code nécessaire à l'ordinateur client pour qu'il calcule le résultat lui-même. Le programme est rarement traduit en langage machine, mais plutôt interprété ou traduit en une forme intermédiaire, le bytecode, qui sera exécuté par une machine virtuelle, ou traduit en langage machine au moment de l'exécution (*just-in-time*). Java, PHP et Javascript sont des langages de programmation basée web.

3.4 Taxinomie [1]des langages

Un langage de programmation est un langage artificiel, compréhensible par une machine de Turing et servant à la ***création et à l'implantation d'algorithme****.*

Un programme exécute des processus qui appliquent des algorithmes à des données. *Un algorithme étant un procédé consistant, pour obtenir un résultat, à appliquer un nombre fini d'opérations dans un ordre déterminé à partir d'opérandes (de données) finies et s'exécutant en un temps fini.*

Cette classification permet de se rendre compte de la grande **diversité des approches choisies** par les concepteurs des langages de programmation.

Dépendance au processeur

Langage **machine** (assembleur), langage **évolué** (Java), langage de **liaison**/de script/macro-langage/inter-application (Javascript)
ou
Langage de **bas niveau**, de **haut niveau**

Structuration

Langage **impératif** (Pascal), langage **fonctionnel** (LISP), langage **objet** (Java)

Déterminisme

Langage **déclaratif** (non déterministe, non procédural) (Prolog), **procédural** (C)

Gestion des processus

Langage à traitement **parallèle** (Ada), **séquentiel**

Langages de bas niveau : Ces langages, proches de la machine, "collent" au jeu d'instructions de l'ordinateur et sont propres à un processeur donné.

Exemple : langage machine, langage assembleur (dépend du type du processeur)

Langages de haut niveau : ces langages, proches de l'homme, permettent de gérer la complexité des problèmes traités grâce à la structuration :

- enregistrements regroupant plusieurs données,
- modules ou procédures regroupant plusieurs instructions élémentaires,
- etc...

[1] Science des classifications.

Un petit classement non exhaustif des langages de programmation :

- ✓ **Langage orientée objet** : Java, Delphi, C#
- ✓ **Langage procédural** : Pascal, C, Fortran
- ✓ **Hybride orientée objet** : C++, ADA 95
- ✓ **Langage basé objet** : Smalltalk, Eiffel
- ✓ **Langage fonctionnel** : FP, MLC CAMLIGHT, OCAML, LISP(modernisé en dialecte scheme)
- ✓ **Langage logique** : Prolog, Programmation par contrainte
- ✓ **Langage à balise** : SGML, HTML, XML
- ✓ **Langage de Script** : Perl, CGI
- ✓ **Langage base de donnée**s : SQL

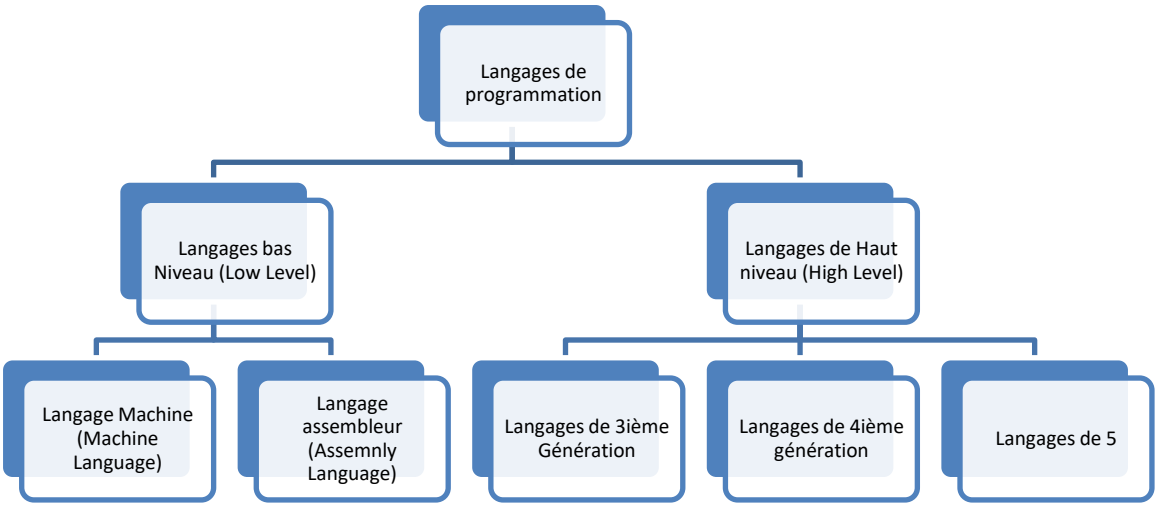
Les langages de programmation sont classés en génération :

- **1ère génération** : langage machine, notation directement compréhensible par la machine.

  Ce langage "colle" au jeu d'instructions de l'ordinateur. C'est ce qu'on appelle également langage de bas niveau.

- **2ème génération** : langage d'assemblage, les instructions du processeur et les divers registres disponibles sont représentés par des mots clés mnémotechniques
- **3ème génération** : langages de haut niveau, langages procéduraux et structurés
  - besoins de gros programmes de calcul numérique ==> FORTRAN
  - besoin d'un langage permettant d'exprimer et de structurer aisément les constructions algorithmiques : Algol 60, Algol 68, Pascal (1971), Modula-2 (1983), C (1972).
  - besoins de simulation du monde réel : Simula-67, Smalltalk (1972, 1980), Eiffel, C++ (1986), Java
- **4ème génération** : générateur de programme, langage de programmation permettant par exemple l'interrogation de base de données avec une syntaxe proche du langage humain.
- **5ème génération :** les langages à programmation logique prétendent représenter cette nouvelle génération, mais l'utilisation est marginale.

Je peux faire le diagramme suivant des générations des langages informatiques :

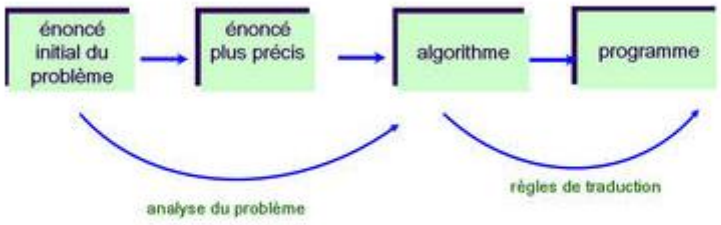


3.4 Mise en œuvre

Avant de programmer, il faut suivre une technique suivante :

- *Phase d'analyse* : chercher par quel moyen on pourra obtenir les résultats souhaités à partir des données dont on dispose.
- *Phase de programmation* : exprimer dans un langage donné (Pascal, Java, ...) le résultat de la phase précédente.

Si l'analyse a été bien menée, il s'agira d'une simple transcription



Démarche pour la programmation

Qu'est ce qu'un programme ?

Un programme traduit la marche à suivre pour expliquer le traitement à effectuer sur les données.

Il s'agit d'une succession d'instructions exécutables par l'ordinateur.

Les instructions du programme sont exécutées les unes après les autres, le plus souvent dans l'ordre séquentiel dans lequel elles sont données dans le programme.

Qu'est ce qu'un ordinateur ?

C'est à l'exécutant que l'on va fournir un programme

Un ordinateur ne manipule que du binaire : pour écrire des programmes, on se sert d'une notation, appelée langage de programmation.

En résumé programmer signifie :

1. concevoir la marche à suivre pour cela il faut rédiger un ensemble de directives pour résoudre un problème
2. soumettre cette marche à suivre à un ordinateur : Pour cela il faut connaître la machine et connaître le langage de programmation à utiliser

La soumission se fait sous forme de programme.

- L'utilisation d'un langage est rendue possible par un traducteur automatique. Un programme qui prend un texte écrit dans ce langage pour en faire quelque chose, en général soit :

Traduction d'un programme :

L'utilisation d'un langage de haut niveau nécessite de traduire les programmes écrits dans ce langage dans un langage machine.

Puisque le programme est combinaison d'instructions destinées à l'ordinateur, il faut que le programme parvienne à l'ordinateur sous une forme exploitable par ce dernier.

Il existe deux approches :

1. lancer un programme de traduction simultanée, appelé interprète, qui traduit et exécute au fur et à mesure les instructions du programme à exécuter.

   Exemple : Basic, Matlab, Perl

2. analyser l'ensemble du programme et le traduire d'avance en un programme en langage machine, qui est ensuite directement exécutable. C'est la compilation.

   Exemple : C, C++, Cobol, Fortran, Pascal

Remarque : Cas du langage Java

Java fait partie des langages à approche hybride : un programme en Java est analysé et compilé, mais pas dans le langage de la machine physique.

Pour des raisons de portabilité, le compilateur Java traduit le programme dans un langage intermédiaire, appelé le byte-code.

Le byte-code est un pseudo-assembleur qui s'adresse à un processeur virtuel appelé machine virtuelle Java : JVM (Java Virtual Machine). Ce concept sera expliqué plus loin.

Fondamental :

Pour traduire un langage, il faut tenir compte des trois niveaux :

- Niveau lexical : concerne le vocabulaire du langage, les règles d'écriture des mots du langage (identificateurs de variables ou fonctions), les mots clés et les caractères spéciaux.
- Niveau syntaxique : spécifie la manière de construire des programmes dans ce langage, autrement dit les règles grammaticales propres au langage ;
- Niveau sémantique : spécifie la signification de la notation.

A chaque langage de programmation correspond donc un interprète et/ou un compilateur

D'une manière générale, un programme est :

- = fichier texte (écrit à l'aide d'un éditeur de texte)
- = fichier source (contient les lignes du programme en langage de haut niveau, appelé code source)

Un *compilateur*

Un programme qui traduit le texte dans un langage qui permettra son exécution, tel le langage machine, le bytecode ou le langage assembleur.

Le compilateur transforme le code source en code objet (fichier objet).

Il crée ensuite un fichier exécutable qui sera chargé en mémoire ou stocké sur un disque dur.

Le nouveau fichier exécutable sera autonome.

Le compilateur doit correspondre au langage que vous avez choisi d'utiliser. Il existe un compilateur par langage de programmation et par plate-forme.

Remarque : Avantages / Inconvénients

- Avantage : on peut diffuser uniquement le fichier exécutable
- Inconvénient à chaque modification du code source, il faut recompiler le programme, en plus il faut un exécutable par type de plate-forme

Un *interpréteur*

Un programme qui exécute les instructions demandées. Il joue le même rôle qu'une machine qui reconnaîtrait ce langage.

Langage machine

Chaque appareil informatique a un ensemble d'instructions qui peuvent être utilisées pour effectuer des opérations. Les instructions permettent d'effectuer des calculs arithmétiques ou logiques, déplacer ou copier des données, ou bifurquer vers l'exécution d'autres instructions. Ces instructions sont enregistrées sous forme de séquences de bits, où chaque séquence correspond au code de

l'opération à effectuer et aux opérandes, c'est-à-dire aux données concernées ; c'est le langage machine.

La traduction s'effectue en plusieurs étapes. En premier lieu, le traducteur effectue une analyse lexicale où il identifie les éléments du langage utilisés dans le programme. Dans l'étape suivante, l'analyse syntaxique, le traducteur construit un diagramme en arbre qui reflète la manière dont les éléments du langage ont été combinés dans le programme, pour former des instructions. Puis, lors de l'analyse sémantique, le traducteur détermine s'il est possible de réaliser l'opération, et les instructions qui seront nécessaires dans le langage cible.

Dans le langage de programmation assembleur, des mots aide-mémoire (mnémonique) sont utilisés pour référer aux instructions de la machine. Les instructions diffèrent en fonction des constructeurs et il en va de même pour les mnémoniques. Un programme *assembleur* traduit chaque mnémonique en la séquence de bits correspondante.

Les langages de programmation fonctionnent souvent à l'aide d'un *runtime*.

Un runtime

Un *runtime* (traduction : *exécuteur*) est un ensemble de bibliothèques logicielles qui mettent en œuvre le langage de programmation, permettant d'effectuer des opérations simples telles que copier des données, voire les opérations beaucoup plus complexes.

Lors de la traduction d'un programme vers le langage machine, les opérations simples sont traduites en les instructions correspondantes en langage machine tandis que les opérations complexes sont traduites en des utilisations des fonctions du *runtime*. Dans certains langages de programmation, la totalité des instructions sont traduites en des utilisations du *runtime* qui sert alors d'intermédiaire entre les possibilités offertes par la plateforme informatique et les constructions propre au langage de programmation.

Chaque langage de programmation a une manière *conventionnelle* de traduire l'exécution de procédures ou de fonctions, de placer les variables en mémoire et de transmettre des paramètres. Ces conventions sont appliquées par le *runtime*. Les *runtime* servent également à mettre en œuvre certaines fonctionnalités avancées des langages de programmation telles que le ramasse-miettes, ou la réflexion.

Les langages de programmation sont couramment auto-implémentés, c'est-à-dire que le compilateur pour ce langage de programmation est mis en œuvre dans le langage lui-même. Exemple : un compilateur pour le langage Pascal peut être écrit en langage Pascal.

Fonctionnalités avancées

Les fonctionnalités avancées telles que le ramasse-miettes (anglais *garbage collector*), la manipulation des exceptions, des événements, ou des threads, ainsi que la liaison tardive et la réflexion sont mises en œuvre par les *runtime* des langages de programmation.

Un ramasse-miettes (garbage collector)

Un mécanisme qui supprime les variables inutilisées et libère l'espace mémoire qui leur avait été réservé.

Une exception

Un fait inattendu, souvent accidentel, entraîne l'échec du déroulement normal du programme, et ce fait *exceptionnel* doit être pris en charge par le programme avant de pouvoir continuer. Certains langages de programmation permettent de provoquer délibérément l'arrêt du déroulement normal du programme.

Un événement

Une procédure qui va être exécutée lorsqu'une condition particulière est rencontrée. Les événements sont notamment utilisés pour mettre en œuvre les interfaces graphiques.

Un thread

Une suite d'instructions en train d'être exécutée. Les langages de programmation qui manipulent les *threads* permettent d'effectuer plusieurs tâches simultanément. Cette possibilité d'exécution simultanées, offerte par les systèmes d'exploitation, est également offerte en *allégé* par les *runtime* des langages de programmation.

La liaison tardive

Le procédé de *liaison* (anglais *late binding* ou *dynamic binding*) consiste à associer chaque identifiant d'un programme avec l'emplacement de mémoire concerné. Cette opération peut être effectuée lors de la traduction du programme, au cours de l'exécution du programme ou juste avant, elle est dite *tardive* lorsque l'opération de liaison est effectuée très tard, juste avant que l'emplacement concerné ne soit utilisé.

La réflexion

La possibilité pour un programme d'obtenir des informations concernant ses propres caractéristiques. Des instructions du langage de programmation permettent à un programme d'obtenir des informations sur lui-même, et de les manipuler comme des données.

Une monade :

Une structure permettant de manipuler des traits impératifs dans des langages fonctionnels purs.

3.5 Utilisations

On peut aussi classer les langages de programmation en fonction de leur utilisation car beaucoup de langages sont spécialisés à une application ou à un domaine particulier.

Langages pour pages Web dynamiques

Ce type de langage est utilisé pour une plus grande interaction entre un client et un serveur.

Du côté du serveur Web, cela permet de produire des pages dont le contenu est généré à chaque affichage. Ces langages sont par ailleurs souvent couplés avec un langage pour communiquer avec des bases de données (exemples : PHP, LiveCode).

Côté client (en général le navigateur web), ces langages offrent la possibilité de réagir à certaines actions de l'utilisateur sans avoir à questionner le serveur. Par exemple, le JavaScript d'une page Web peut réagir aux saisies de l'utilisateur dans un formulaire (et vérifier le format des données).

Certains langages permettent de développer à la fois les aspects client et serveur. C'est le cas d'Ocsigen, de Hop, de Dart ou bien encore du Server-Side JavaScript.

Langages de programmation théorique

On désigne parfois par langage de programmation théorique les systèmes formels utilisés pour décrire de façon théorique le fonctionnement des ordinateurs. Ils ne servent pas à développer des applications mais à représenter des modèles et démontrer certaines de leurs propriétés.

On peut citer la machine de Turing et le λ-calcul de Church, qui datent tous les deux des années 1930, et donc antérieurs à l'invention de l'ordinateur. Le λ-calcul a par la suite servi de base théorique à la famille des langages de programmation fonctionnelle. Dans les années 1980, Robin Milner a mis au point le π-calcul pour modéliser les systèmes concurrents.

Langages exotiques

Les langages exotiques ont pour but de créer des grammaires complètes et fonctionnelles mais dans un paradigme éloigné des conventions. Beaucoup sont d'ailleurs considérés comme des blagues.

Ces langages sont généralement difficiles à mettre en pratique et donc rarement utilisés.

Langages spécialisés

ABEL, langage pour la programmation électronique des PLD

CDuce, langage fonctionnel d'ordre supérieur pour la manipulation de documents au format XML.

Forme de Backus-Naur (BNF), formalisation des langages de programmation

PROMELA, langage de spécification de systèmes asynchrones

VRML, description de scènes en trois dimensions

Langages synchrones

Langages de programmation synchrones pour les systèmes réactifs : Esterel, Lustre.

Langages à vocation pédagogique

Les pseudo-codes ont généralement un but uniquement pédagogique.

Logo est un langage fonctionnel simple à apprendre.

Dans les années 1990, le langage BASIC était souvent conseillé pour débuter. Il avait cependant la réputation de favoriser la prise de mauvaises habitudes de programmation.

Le Processing est un langage simplifié qui s'appuie sur Java. Il permet un développement d'applications fenêtrées sur tout type d'ordinateur équipé de Java.

L'Arduino est un langage simplifié s'appuyant sur C/C++. Il permet un développement simple de projets électroniques à partir de carte Arduino (AVR).

L'ArduinoEDU est un langage encore plus simple, en français, pour les grands débutants s'appuyant sur le langage C/C++/Arduino. Il permet un développement très simple de projets électroniques à partir de cartes Arduino (AVR).

Flowgorithm est un outil de création et modification graphique de programmes informatiques sous forme d'Algorigramme.

Langages pour l'électronique numérique

Verilog, VHDL : langages de description matérielle, permettant de synthétiser de l'électronique numérique (descriptions de portes logiques) et d'en simuler le fonctionnement

SystemC, langage de description matérielle de plus haut niveau que les précédents et permettant une simulation plus rapide

Langages pour la statistique

R, SAS et xLispStat sont à la fois un langage de statistiques et un logiciel.

Langages de programmation de Commande Numérique (C.N.)

Une machine-outil automatisée, ou Commande Numérique (C.N.), a besoin d'un langage de programmation pour réaliser les opérations de tournage ou de fraisage…

Langages de programmation des automates programmables industriels (API)

Sequential function chart, langage graphique, dérivé du grafcet (NB : le grafcet définit les spécifications de façon graphique).

Langage Ladder, langage graphique.

Langages de programmation audio

Nyquist est un langage de synthèse et d'analyse sonore. Pure Data est un logiciel de création musicale graphique qui repose sur un langage de programmation procédural.

Chapitre 4 : Les personnages

Paul Allen : son nom complet est Paul Gardner Allen. Il né le 21 janvier 1953 à Seattle (Washington) au USA et mort le 15 octobre 2018.

Co-fondateur de la société Microsoft.
Aide au développement des premiers produits Microsoft tels que MS-DOS.
Développement d'Altair BASIC pour le MITS Altair qui est ensuite devenu Microsoft BASIC.
A acheté ZDTV en 2001 et a changé son nom en TechTV.

Charles Babbage : Il est né le 26 Décembre 1791 à Londres et mort en Octobre 1871.

Il désigne le premier mécanisme d'ordinateur qui avait appelé le « Difference Engine » - une machine qui permettait de résoudre les équations polynômiales sans utiliser la multiplication ou la division. Il commençait à développer sa machine en 1822 et travaillait pendant 10 ans mais son invention n'a jamais été terminé.

Son analytical Engine a été développé durant les années 1830. Cette machine a permit de donner les premiers concepts de l'informatique par la programmation par mémoire, les opérations bases par instruction, les boucles de contrôle et le concept d'entrée et de sortie. Maintenant, on considère qui est le père de l'informatique.

Ces publications significatives sont :

- A Comparative View of the Various Institutions for the Assurance of Lives (1826).
- On the Economy of Machinery and Manufactures (1835).
- Table of the Logarithms of the Natural Numbers from 1 to 108000 (1841).
- Passages from the Life of a Philosopher (1864).

John Backus : est né le 3 Décembre 1924 à Philadelphie et mort le 17 Mars 2007.

C'est un informaticien américain, et prend la direction de l'invention du langage FORTRAN de haut niveau.

Il y a une des récompenses :

- Nommé IBM Fellow (1963).
- Récompensé par W.W. Prix McDowell (1967).
- Reçu la médaille nationale de la science (1975).
- Récompensé par le prix ACM Turing (1977).
- Membre de l'Académie américaine des arts et des sciences (1985).
- Diplômé honoris causa de l'Université Henri Poincaré (1989).
- Prix Draper (1993).
- Récompensé par le Computer History Museum Fellow Award (1997).
- L'astéroïde 6830 Johnbackus nommé en son honneur (2007).

Tim Berners-Lee : Il né le 8 juin 1955 à Londres.

Inventé le WWW (World Wide Web).
Hypertexte proposé.
Création du premier site Web.
Fondé le World Wide Web Consortium.

Ces honneurs et prix :

- En 2017, Berners-Lee a reçu le Turing Award, la plus prestigieuse distinction en informatique, avec un prix d'un million de dollars fourni par Google.
- L'un des six membres du World Wide Web Hall of Fame.
- A remporté le prix «Jeune innovateur de l'année» de la Fondation Kilby.
- Reçu également le Software System Award de l'Association for Computing Machinery.
- Récipiendaire d'un doctorat honorifique de l'Université d'Essex.
- Le Time Magazine a nommé Berners-Lee l'une des 100 personnalités les plus importantes du 20e siècle.
- A reçu un diplôme honorifique de l'Open University en tant que docteur de l'université.
- Élu membre de l'Académie américaine des arts et des sciences.
- A reçu le prix Fellow du Computer History Museum, pour ses contributions fondamentales au développement du World Wide Web.
- Nommé comme le premier lauréat du Millennium Technology Prize de la Finlande, pour avoir inventé le World Wide Web.
- Nommé au grade de Chevalier Commandeur de l'Ordre le plus excellent de l'Empire britannique (la deuxième classe la plus élevée de cet Ordre qui implique une chevalerie) par la reine Elizabeth II.
- Présenté avec un doctorat honorifique en sciences de l'Université de Lancaster.
- Nommé le plus grand britannique de 2004.
- A reçu le Golden Plate Award de l'Academy of Achievement.
- Reçu l'Ordre du Mérite, devenant l'un des 24 membres vivants autorisés à détenir l'honneur et à utiliser les post-nominaux «O.M». après leur nom.
- Récipiendaire du prix IEEE / RSE Wolfson James Clerk Maxwell 2008, pour «avoir conçu et développé le World Wide Web».
- Titulaire d'un doctorat honorifique de l'Université de Manchester.
- Titulaire d'un doctorat honoris causa de l'Universidad Politécnica de Madrid.
- Membre élu de l'Académie nationale des sciences des États-Unis.
- A reçu le prix Webby pour l'ensemble de ses réalisations.
- Titulaire d'un doctorat honoris causa de la Vrije Universiteit Amsterdam.
- L'un des trois premiers récipiendaires du prix Mikhail Gorbatchev pour "L'homme qui a changé le monde".
- Récompensé par un doctorat honorifique en sciences de l'Université Harvard.
- Intronisé au Temple de la renommée de l'IA de IEEE Intelligent Systems pour ses «contributions significatives au domaine de l'IA et des systèmes intelligents».
- Intronisé au Temple de la renommée de l'Internet par l'Internet Society.

Wen Tsing Chow : Il est née en 1918 à Taiyuan, Shanxi, China et mort en 2001.

Ses contributions a été le pionner de l'ordinateur entre les chinois et les américains et contribuait à la science des missiles. Il est aussi le pionner de l'utilisation de l'ordinateur avec les satellites, missiles et guide les vaisseaux spatiaux. Il a déposé un brevet pour utilisation de la mémoire PROM.

- Il a reçu en 2004, la récompense de l'Air Force Space et Missiles Pionners Award.

Edgar Codd : né le 23 août 1923 à ISLE of Portland en Angleterre et mort le 18 avril 2003.

Il a été informaticien anglais chez IBM et invente la théorie de SQL et donne le terme OLAP (online analytical processing) et crée les relations algébriques.

Ces publications significatives :

- A Relational Model of Data for Large Shared Data Bank. (1970).
- Relational Completeness of Data Base Sublanguages". Database Systems: 65-98. (1970).
- 1981 Turing Award Lecture - Relational Database: A Practical Foundation for Productivity. (1981).
- The Relational Model for Database Management (Version 2 ed.) (1990).
- Providing OLAP to User-Analysts: An IT Mandate. (1993).

Ces honneurs et prix :

- Prix Turing (1981).
- Intronisé en tant que membre de l'Association for Computing Machinery (1994).
- SIGMOD a renommé son prix le plus élevé en SIGMOD Edgar F. Codd Innovations Award, en son honneur (2004).

Alain Colmerauer : Née le 24 janvier 1941 à Carcassonne en France et mort le 12 mai 2017.

Il est un informaticien français et crée le système Q et développe TAUM-METEO qui est une machine de traduction. Il crée le langage Prolog.

Ces honneurs et prix :

- ACP Research Excellence Award (1991).
- Fellow of the American Association for Artificial Intelligence (1991).
- Made Chevalier de la Legion d'Honneur by the French government (1986).
- Michel Monpetit Award (1985).
- Conseil Regional of Provence, Alpes, and Côte d'Azur Award (1984).
- Apple France Award for the Prolog II implementation (1982).

Alan Cooper : né le 3 juin 1952 à San Francisco au Californie (USA).

Largement reconnu comme le «père de Visual Basic». A écrit "General Ledger" l'un des premiers logiciels d'entreprise écrits pour les micro-ordinateurs, puis à la création de Visual Basic. Rédaction

du langage de programmation d'entreprise CBASIC, l'un des premiers concurrents de Microsoft BASIC.

Publications significatives :

- The Inmates Are Running the Asylum: Why High-Tech Products Drive Us Crazy.
- How to Restore the Sanity, About Face: The Essentials of User Interface Design, VBX.

Ole-Johan Dahl : né le 12 octobre 1931 à Mandal en Norvége et mort le 29 juin 2002.

Il était informaticien norvégien qui est considéré comme un des pères de Simula et de la programmation orientée objet.

Il a produit l'initiative de la programmation orientée objet en 1960 à Norwegian Computer Center avec le langage SIMULA I et Simula 67.

Honneur et prix :

- Médaille IEEE John von Neumann (2002).
- Prix Turing (2001).

Brendan Eich : né en 1961 à Pittsburgh en Pennsylvanie.
Programmeur informatique et surtout connu comme le créateur du langage de programmation JavaScript. Directeur de la technologie chez Mozilla Corporation. A commencé à travailler chez Netscape Communications Corporation en avril 1995, travaillant sur JavaScript (initialement appelé Mocha, puis LiveScript) pour le navigateur Web Netscape Navigator.

Dov Frohman : il est né le 28 mars 1939 à Amsterdam. Il a été ingénieur d'électrique pour Israël, auteur. Il a formé le vice président de l'entreprise Intel et inventeur de EPROM.

Les victoires et honneurs :

- Intronisé au National Inventors Hall of Fame (2009).
- A reçu la médaille Edison de l'IEEE (2008).
- Récompensé du Prix d'Israël pour les sciences exactes (1991).
- Récipiendaire du prix IEEE Jack Morton (1986).

Bill Gates : son nom complet est William H. Gates et né le 28 octobre 1955 à Seattle (Washington) au Etats-Unis.

Co-fondateur et premier PDG de la société Microsoft.
Développement d'Altair BASIC pour le MITS Altair qui est ensuite devenu Microsoft BASIC.
Aide au développement des programmes antérieurs de Microsoft tels que MS-DOS.

Ces honneurs et prix :

- Médaille présidentielle de la liberté (2016).
- Récompensé par le Silver Buffalo Award par les Boy Scouts of America.
- Récipiendaire du prix Bower 2010 pour le leadership en affaires du Franklin Institute.
- Le magazine Time a nommé Bill Gates parmi les 100 personnes les plus influentes du 20e siècle et parmi les 100 personnes les plus influentes de 2004, 2005 et 2006.

• Bill et son épouse ont reçu l 'Ordre de l' aigle aztèque pour leur travail philanthropique en 2006.
• Fait chevalier honoraire Commandeur de l 'Ordre de l'Empire britannique (KBE) par la reine Elizabeth II en 2005.
• Honoré en tant que 12e membre émérite de la British Computer Society.
• Le plus jeune milliardaire autodidacte.
• Présenté sur le devant du magazine TIME en 1984.

Rober Gentleman : né 1959. Bioinformaticien et statisticien canadien. Directeur scientifique chez 23andMe (nommé en 2015). Co-créateur du langage de programmation R avec Ross Ihaka. Directeur principal chez Bioinformatics.

Publications significatives :

- Bioconductor Case Studies (Use R!) (2013).
- R Programming for Bioinformatics (2008).
- Bioinformatics and Computational Biology Solutions Using R and Bioconductor (2005).

Honneurs et prix :

- Fellow ISCB, Société internationale de biologie computationnelle (2014).
- Prix de calcul statistique et graphique (2010).
- Ross L. Prentice Professeur de biostatistique (2009).
- Prix Benjamin Franklin (2008).

James Gosling : né le 19 mai 1955 prés Calgary en Alberta au Canada.

Informaticien, mieux connu comme le père du langage de programmation Java. Entre 1984 et 2010, Gosling était avec Sun Microsystems. Embauché par Google le 28 mars 2011, et a depuis rejoint une startup appelée Liquid Robotics. Gosling est répertorié en tant que conseiller de la société Scala Lightbend (anciennement Typesafe), lancée en mai 2011.

Publications significatives :

- The Java Programming Language, Fourth Edition (2005).
- The Java Language Specification, Third Edition (2005).
- The Real-Time Specification for Java (2000).
- The Java Application Programming Interface (1996).
- The Java language Environment: A white paper (1996).
- The NeWS Book : An Introduction to the Network/Extensible Window System (1989).

Honneurs et prix :

- Fait Officier de l'Ordre du Canada (2007).
- Récompensé par le prix The Economist Innovation Award (2002).

Grace Hopper : Elle est née le 9 décembre 1906 à New York à l'USA et mort le 1 janvier 1992. Informaticienne américaine et officier. Créer les premiers programmes pour Harvard Mark I computer et fait partie de l'équipe de développement de COBOL.

Ces distinctions :

• Médaille présidentielle de la liberté (2016) (posthume).
• Lancement de l'USS Hopper (DDG-70) (1996), l'un des rares navires militaires américains nommé en l'honneur d'une femme.
• Médaille nationale de la technologie (1991).
• Prix du boursier du Computer History Museum (1987).
• La première femme à retourner au service actif après que la marine des États-Unis n'a pas été en mesure d'élaborer un plan de paie après 823 tentatives.
• Encore une fois rappelé de sa retraite pour aider à normaliser les langages informatiques de haut niveau utilisés par la Marine.
• En raison de ses réalisations, de son rang militaire et de ses réalisations sans précédent, elle est parfois appelée «Amazing Grace».

Robert Griesemer : Programmeur informatique qui travaille chez Google. L'un des premiers concepteurs du langage de programmation Go. Auparavant travaillé sur la génération de code pour le moteur JavaScript V8 de Google et Chubby. Il a travaillé sur le langage de programmation Sawzall, la machine virtuelle Java HotSpot et le système Strongtalk.

H**erman Hollerith** : né le 29 Février 1860 à Buffalo (USA) et mort le 17 Novembre 1929.

Alan Kay : Alan Curtis Kay est né le 17 mai 1940. Il crée des ordinateurs protables avex la sociétéré Xerox PARC.

Thomas Kurtz : Thomas Eugene Kurtz est née le 22 février 1928. Il était le co-développeur du langage de programmation BASIC de 1963 à 1964 avec John Kemeny à Dartmouth College.

Mary Keller : Sœur Mary Kenneth Keller née en 1914 et mort en 1985. Première femme américaine qui append l'informatique à l'université The University of Wisconsin-Madison en 1965. Elle a aussi étudié au Darmouth College où elle aide à développer le langage de programmation BASIC. Elle a fondé le Computer Science Departement Clarke College en Duque, Iowa.

John Kemeny : Son nom complet est John George Kemeny. Il est né le 31 mai 1926 à Budapest en Hongries et mort le 26 décembre 1992.

Il était un mathématicien hongrois, informaticien et enseignant. Il est le co-développeur de langage de programmation Basic en 1964 avec Thomas E. Kurtz et il a été le 13^{ième} président du Dartmouth College de 1970 à 1981 et parti des pionniers dans l'enseignement de l'informatique.

Brian Kernighan : son nom en entier est Brian Wilson Kernighan. Il né en 1942 à Toronto au Canada.

Informaticien Canadien qui conçoit le terme Unix en 1970. L'origine du terme était UNICS (Uniplexed Information and Computing Service, a play on Multics), qui a changé en UNIX. Il développe UNIX chez Bell Labs.

<u>Principaux travaux sont :</u>

- Software Tools (1976 with PJ Plauger).
- Software Tools in Pascal (1981 with PJ Plauger).
- The C Programming Language ('K&R') (1978, 1988 with Dennis M. Ritchie).
- The Elements of Programming Style (1974, 1978 with PJ Plauger).
- The Unix Programming Environment (1984 with Rob Pike).
- The AWK Programming Language (1988 with Al Aho and Peter J. Weinberger).

- The Practice of Programming (1999 with Rob Pike).
- AMPL: A Modeling Language for Mathematical Programming, 2nd Ed. (2003 with Robert Fourer and David Gay).
- D is for Digital: What a well-informed person should know about computers and communications (2011).

Rasmus Lerdorf : né le 22 novembre 1968 à Qeqertarsuaq au Groenland. Programmeur danois de citoyenneté canadienne mieux connu comme le créateur du langage de script PHP. Il est l'auteur des deux premières versions. . Contribution au serveur HTTP Apache et ajout de la clause LIMIT au SGBD MYSQL

Honneurs et prix :

Nommé au MIT Technology Review TR100 comme l'un des 100 meilleurs innovateurs au monde de moins de 35 ans (2003).

Håkon Wium Lie : né 1965 à Halden en Norvège

Pionnier du Web, activiste des standards et, depuis 2011, CTO (Chief Technology Officer) d'Opera Software. Mieux connu pour avoir créé le concept et aidé au développement de CSS (Cascading Style Sheets) en 1994.

Ada Lovelace : son nom de naissance Augusta Ada Byron le 10 décembre 1815 à Londres. Elle est devenue reine Augusta Ada et Comtesse de Lovelace. Elle est morte le 27 novembre 1852.

Elle écrit le premier programme sur un ordinateur et développe le premier algorithme sur une machine.

Elle a aidé Charles Babbage a utilisé son analytical engine.

John MCCarthy : est né le 4 septembre 192 à Boston à l'USA et mort le 24 octobre 2011. C'est un ingénieur informaticien et scientifique. Il conçoit le terme intelligence artificielle et créer le langage LISP qui influence le design de ALGOL.

Ces publications sont :

- Programs with Common Sense. In Proceedings of the Teddington Conference on the Mechanization of Thought Processes (1959).
- Recursive functions of symbolic expressions and their computation by machine (1960).
- A basis for a mathematical theory of computation. In Computer Programming and formal systems (1963).
- Situations, actions, and causal laws. Technical report, Stanford University (1963).
- Some philosophical problems from the standpoint of artificial intelligence (1969).
- Epistemological problems of artificial intelligence (1977).
- Circumscription: A form of non-monotonic reasoning (1980).

Les honneurs et prix :

- Prix Turing de l'Association for Computing Machinery (1971).
- Prix de Kyoto (1988).

• National Medal of Science (USA) en sciences mathématiques, statistiques et computationnelles (1990).
• Intronisé en tant que Fellow du Computer History Museum (1999).
• Médaille Benjamin Franklin en informatique et sciences cognitives du Franklin Institute (2003).
• Intronisé au Temple de la renommée de l'IA de IEEE Intelligent Systems, pour ses «contributions significatives au domaine de l'IA et des systèmes intelligents» (2011).

Marvin Minsky : Il né le 9 aout 1927 à New York au USA et mort le 26 janvier 2016.

Il était physicien cognitive américain pour IA, co-foundateur de Massachusetts Institute of Technology's qui est un laboratoire de IA. Il a écrit un livre sur la perception avec Saymour Papert alors il est devenu le fondateur du travail sur le réseau neurone artificiel.

Il a inventé le premier head-mounted graphic display en 1963 et le microscope confocal en 1957.

Principale publication :

- The Emotion Machine, Simon and Schuster (2006).
- The Society of Mind, Simon and Schuster (1987).
- Communication with Alien Intelligence, (1985).
- Perceptrons, with Seymour Papert, MIT Press (1969).

Honneurs et prix :

• Prix Turing (1969).
• Prix du Japon (1990).
• Prix IJCAI d'excellence en recherche (1991).
• Médaille Benjamin Franklin (2001).

Cleve Moler : né le 17 août 1939.

Programmeur informatique et mathématicien américain.
Inventeur de MATLAB, un progiciel de calcul numérique.
Président et scientifique en chef chez MathWorks.
Spécialiste en analyse numérique.
L'un des auteurs de LINPACK et EISPACK, bibliothèques Fortran pour le calcul numérique dans les années 1970.

Ces publications significatives sont :

- Numerical Computing with MATLAB (2008).
- Experiments in Computational Matrix Algebra (1988).
- Computer methods for mathematical computations (1977).

Ces honneurs et prix :

• Récipiendaire de la médaille IEEE John von Neumann (2014).
• Nommé récipiendaire du Computer Pioneer Award par l'IEEE Computer Society (2012).
• Nommé docteur honoris causa à l'Université technique du Danemark (2012).
• Élu à la National Academy of Engineering (1997).

Kristen Nygaard : Né le 27 aout 1926 en Norvège et mort le 10 aout 2002.

Il était informaticien norvégien, pionner en langage de programmation et politicien.

Avec Ole-Johan Dahl, ils ont développée SIMLUA I (1961-1965) et SIMULA-6, les premiers langages de programmation orienté-objet. Ces langages introduisait le concept objet, classes, héritage, la quantité virtuel et le multi-threaded en programmation à exécution et donne les fondations du POO ; programmation Orienté-Objet.

Ces prix :

- Prix A. M. Turing de l'ACM (2002).
- Médaille IEEE John von Neumann de l'Institute of Electrical and Electronic Engineers (2001).
- Fait Commandeur de l'Ordre royal norvégien de Saint-Olav par le roi de Norvège (2000).

Seymour Papert : Il est né le 29 février 128 à Pretoria au Sud Afrique et mort le 31 juillet 2016.

Il était un informaticien et professeur de mathématique à MIT et un des pionniers de intelligence artificielle. Il a inventé le langage de programmation LOGO.

Ces publications significatives :

- The Connected Family: Bridging the Digital Generation Gap (1996).
- The Children's Machine: Rethinking School in the Age of the Computer (1993).
- Mindstorms: Children, Computers, and Powerful Ideas (1980).
- Counter-free automata (1971).
- Perceptrons (1969).

Ces honneurs et prix :

- Prix Smithsonian de Computerworld (1997).
- Prix pour l'ensemble de ses réalisations de la Software Publishers Association (1994).
- Bourse internationale Marconi (1981).
- Bourse Guggenheim (1980).
- Appelé «le plus grand éducateur de mathématiques vivant» par Marvin Minsky.

Dennis Ritchie : son nom en entier –Dennis MacAlistair Ritchie. Il est né le 9 septembre 1941 à Bronxville au USA et mort le 12 octobre 2011.

Il a crée le langage de programmation B en 1969 avec Ken Thompson et le langage de programmation C en 1972 avec Brian Kernighan. Il aide à développé le système exploitation UNIX et Multics.

Ces honneurs et prix :

- Japan Prize for Information and Communications, with Ken Thompson (2011).
- National Medal of Technology, with Ken Thompson (1998).
- IEEE Richard W. Hamming Medal (1990).
- Elected to the National Academy of Engineering (1988).
- Turing Award (1983).

Guido van Rossum : né le 31 janvier 1956 au Pays-Bas.

- Programmeur informatique néerlandais qui est surtout connu comme l'auteur du langage de programmation Python.
- Dans la communauté Python, Van Rossum est connu sous le nom de BDFL (Benevolent Dictator For Life), ce qui signifie qu'il continue de superviser le processus de développement de Python, en prenant des décisions si nécessaire.
- A travaillé pour Google de 2005 à 2012 et travaille maintenant pour Dropbox.

Publications significatives :

- Recognized as a Distinguished Engineer by the Association for Computing Machinery (2006).
- NLUUG Award (2003).
- Award for the Advancement of Free Software (2001).

Guy Steele : Son nom complet est Guy Lewis Steele Jr., il est né le 2 octobre 1954 à Missouri à l'USA.

Souvent appelé "The Great Quux" Informaticien américain qui a joué un rôle important dans la conception et la documentation de plusieurs langages de programmation informatique.
Il a rejoint la société Thinking Machines, où il a aidé à définir et promouvoir une version parallèle de Lisp appelée * Lisp (Star Lisp).

En 2005, a commencé à diriger une équipe de chercheurs chez Sun développant un nouveau langage de programmation nommé Fortress, un langage haut performances conçu pour rendre Fortran obsolète.

Ces publications significatives :

- The Java Language Specification (2005).
- The High Performance Fortran Handbook (1994).
- Common Lisp the Language (1984).
- C: A Reference Manual (1984).
- The Hacker's Dictionary (1982).

Ces honneurs et prix :

- Prix d'excellence en programmation de la revue Dr. Dobb (2005).
- Nommé Sun Fellow (2003).
- Membre de l'AAAS (American Academy of Arts and Sciences) (2002).
- Membre de la NAE (National Academy of Engineering) (2001).
- Membre, ACM (1994).

Bjarne Stroustrup : Il est né le 30 décembre 1950 à Aarthus au Danemark.

Informaticien danois, connu pour la création et le développement du langage de programmation C ++.

Publications significatif :

- A Tour of C++ (2013).
- Addison-Wesley Professional; 1 edition (December 29, 2008).
- Programming: Principles and Practice Using C++ (2008).

- The Design and Evolution of C++ (the "D&E") (1994).
- The Annotated C++ Reference Manual (the "ARM") (with Margaret A. Ellis) (1990).
- The C++ Programming Language (1986).

Ces prix et honneurs :

- Prix ACM Grace Murray Hopper (1992).
- Prix de l'entrepreneur en informatique de l'IEEE Computer Society 2004 (2004).
- Prix William Procter pour la réalisation scientifique (2005).
- Prix d'excellence en programmation du Dr Dobb (2008).

Don Syme : Informaticien d'origine australienne. Chercheur principal chez Microsoft Research. Créateur du langage de programmation F#.

Ken Thompson : son nom en entier est Kenneth Lane Thompson et né le 4 février 1943 à New Orléans en Louisiane au USA.

Il aide à développer l'OS UNIX et Multics.

Il a crée le langage de programmation B avec Dennis Ritchie chez Bell labs en 1969.

Ces honneurs et prix :

- Prix du Japon pour l'information et la communication (avec Dennis Ritchie) (2011).
- Prix Tsutomu Kanai (1999).
- Médaille nationale de la technologie (avec Dennis Ritchie) (1998).
- Médaille IEEE Richard W. Hamming (1990).
- Élu, National Academy of Engineering (1988).
- Prix Turing (1983).

Larry Wall : Né le 27 septembre 1954 à Los Angeles en Californie (USA).

Développeur original de Perl.

Ces publications significatives :

- Programming Perl (1991).
- Perl Language Reference Manual (2010).

Niklaus Wirth : né le 15 février 1934 à Winterthur en Suisse.

Informaticien suisse qui design différents langages de programmation comme le Pascal et fait parti des pionniers du Sofware engineering, ALGO W, Euler, Modula, Modula-2, Oberon, Oberon-2, Oberon-07 et Oberon System.

Publications significatif:

- Theory and Techniques of Compiler Construction (1996).
- Algorithms + Data Structures = Programs (1975).
- PASCAL - User Manual and Report. (1974).

Prix et honneurs :

- Prix ACM Turing (1994).
- Intronisé en tant que Fellow de l'ACM (1994).

Chapitre 5 : Vocabulaires

Abstraction : L'action de se focaliser exclusivement sur les parties importantes du problème.

ALGO : est abréviation de Algorithmic language. ALGO est un langage de programmation scientifique.

Les majeures versions sont :

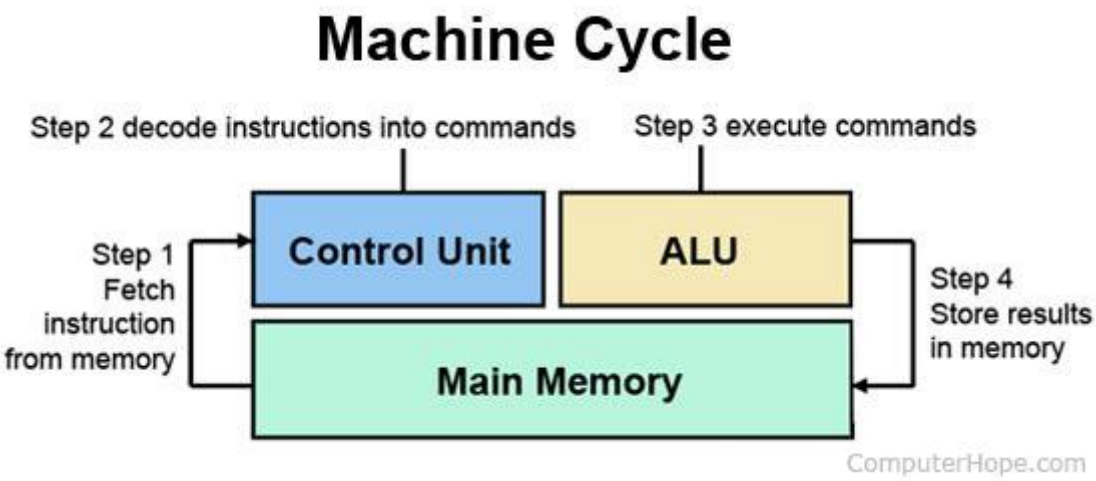
- **ALGOL 58** - introduced in 1958
- **ALGOL 60** - introduced in 1960
- **ALGOL 68** - revised in 1973

Algorithmes : L'action de trouver une solution à travers une règle composée par une série d'étapes.

Altair Basic : Altair BASIC est un langage BASIC développé par Bill Gates, Paul Allen et Monte Davidoff qui a été introduit pour la première fois et annoncé comme achevé le 2 janvier 1975. Altair BASIC a permis aux gens de créer des programmes pour les ordinateurs Altair.

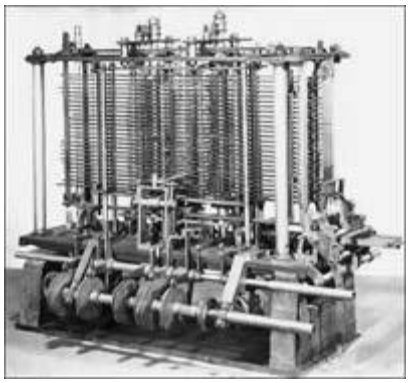
ALU : est abréviation de arithmetic logic unit. Le ALU est un circuit complexe numérique. C'est l'endroit que l'ordinateur fait les opérations binaires.

Le multiple Arithemtic Logic Units sont dans CPUs, GPUs, et FPUs. Dans le processeur, le ALU est divisé en AU et LU. Le AU performe en opération arithmétique et le LU en opérations logique.



Analytical engine (en français : machine analytique) : est une machine qui a été proposé en premier par Charles Babbage en 1837, qui est considéré pour être la première génération ordinateur. Le mécanisme est basé par le ALU* (Arithmetic logic unit) et permettait faire de la programmation basique par flow control*.

C'est programme utilise punch cards (inspiré par Jacquard Loom). C'est aussi l'intégration de la mémoire. Pour ces raisons, les historiens considèrent que c'est le premier concept de l'ordinateur.



Babel : Babel est un langage de programmation à usage général lancé pour la première fois en 2014. Il est conçu pour écrire des programmes destinés à préserver les ressources système et la durée de vie de la batterie sur les appareils cibles. Les programmes écrits dans Babel sont multiplateformes, récupérés et peuvent être rapidement prototypés à l'aide d'un débogueur REPL (lecture-évaluation-impression en boucle) intégré.

BASIC : L'origine de BASIC était développée par Dartmouth College par John Kemeny, Mary Keller et Thomas Kurtz et présentait le 1 mai 1964. BASIC est abréviation de Beginner's All-purpose Symbolic Instruction Code. Il a été très populaire dans les années 1970 à 1980. Aujourd'hui n'est plus utilisé dans les programmes développées mais reste comme un langage de la fondation de la programmation.

BCPL : est abréviation de Basic Combined Programming Language. BCPL était développé en 1966 par Martin Richards de The University of Cambridge. Ce langage est de hautes portabilités et successeur du langage de programmation CPL.

Besoins : (requirements) ou plus généralement tout ce qui précède, motive et/ou justifie le développement.

Boucles : Les boucles sont à la base d'un concept très utile en programmation : itération. L'itération permet d'exécuter de manière récursive une ou plusieurs opérations à tous les éléments qui font partie de la liste itérée. Les boucles permettent de parcourir un tableau ou une base de données par exemple.

Clojure : est une dérivée de LISP. Crée par Rich Hickey en 2007. Utilisent le mutithreat, la programmation fonctionnelle et compatible avec JAVA.

COBOL : est abréviation de Common Business Oriented Language. Ce langage a été crée le 1959 par Grace Hopper et Bob Bemer. Souvent utiliser pour les Gouvernements et les business.

CoffeScript : CoffeeScript est un langage de programmation multi-paradigme qui peut être converti en JavaScript lors de sa compilation. Cette capacité permet aux développeurs de créer des programmes JavaScript comme produit final, mais d'écrire les programmes dans un langage qui utilise une syntaxe plus pratique et plus robuste.

CoffeeScript facilite les styles de programmation impératifs et fonctionnels et peut être utilisé comme langage de script. Sa syntaxe intègre certains des éléments les plus populaires de Ruby, Haskell et Python. Par exemple, il utilise des espaces pour définir des blocs de code, un peu comme Python.

Compilation : la compilation d'un programme consiste à transformer toutes les instructions (ex : code source) en langage machine avant que le programme puisse être exécuté et, par conséquent, il sera nécessaire de refaire la compilation après des changements apportés au code source.

Construction : c'est-à-dire toutes les activités nécessaires à l'implémentation concrète de l'application.

Conventions : Il est de nature conventionnelle et arbitraire, car le fait d'utiliser certains mots et une certaine syntaxe est décidée par les créateurs du langage.

Curry : Curry est un langage de programmations expérimentales et multi-paradigme introduit par Michael Hanus, Herbert Kuchen et Juan Jose Moreno-Navarro en 1995. Son nom est un hommage au logicien Haskell Brooks Curry.

Le langage a été conçu avec l'intention de «combiner les caractéristiques les plus importantes des langages fonctionnels et logiques… [qui sont] les paradigmes de programmation déclaratifs les plus importants». Le code Curry est soit interprété, soit compilé dans les langages Prolog ou Java.

D : D est un langage de programmation informatique qui est un langage de niveau supérieur à C ++ qui a commencé le développement en décembre 1999. D a de nombreuses similitudes avec C et C ++ et est bien adapté pour créer des programmes de plusieurs millions de lignes à moyenne et grande échelle.

Dart : Dart est un langage de programmation Web open source développé par Google. Introduit en octobre 2011, Dart a été créé dans l'espoir qu'il remplacerait un jour JavaScript comme principal langage de programmation pour les applications Web. Il a une syntaxe similaire à celle du langage de programmation C et prend en charge les constructions orientées objet telles que les classes et l'héritage.

Décomposition : L'action de diviser un problème complexe ou un système en petites parties plus simples à gérer.

Déploiement : L'application est rendue disponible aux utilisateurs.

Dylan : Est un langage de programmation était développé dans les années 90 par une équipe ingénieur d'APPLE. C'est un langage qui utilise différente paradigme comme le fonctionnel et la programmation orientée objet (POO).

Evénement : On parle d'événement lorsque la logique de l'application prévoit l'exécution de certaines instructions qui sont liées à une modification de l'état actuel du programme. Les événements peuvent être liés à plusieurs aspects comme par exemples :

- Les événements liés aux *manipulations de l'utilisateur* : cliquer sur le bouton de la souris, sur la touche du clavier, déposer un objet après l'avoir glissé sur une cible, …
- Les événements liés *au temps* : la disparition d'un message après 10 secondes, le temps limite pour terminer un QCM, …
- Les événements liés aux *ressources internes ou externes à l'application* : l'établissement d'une connexion à une base de données, la fin d'un téléchargement d'une image, le démarrage de la lecture d'une vidéo, …

Evolution : Il évolue dans le temps, car de nouveaux mots ou de nouvelles possibilités syntaxiques peuvent s'ajouter ou disparaître du langage lorsqu'une nouvelle version est introduite.

Expressions régulières : Parmi les nombreuses difficultés qu'on peut rencontrer dans la programmation, les expressions régulières sont parmi les plus compliquées à maitriser. Néanmoins, elles sont fondamentales au fonctionnement de plusieurs concepts informatiques, y compris les langages de programmation eux-mêmes.

F# : Prononcé comme "eff sharp", F # est un open source multiplateforme. Le langage de programmation fonctionnel d'abord développé par Don Syme chez Microsoft apporte la programmation fonctionnelle à la plate-forme .NET de Microsoft. Vous trouverez ci-dessous un exemple de base d'impression de "HelloWorld!" à l'écran.

Fonctions : Les fonctions représentent une sorte de « programme dans le programme », car elles sont la première forme d'organisation du code. On utilise des fonctions pour regrouper des instructions et les appeler sur demande : chaque fois qu'on a besoin de ces instructions, il suffira d'appeler la fonction au lieu de répéter toutes les instructions.

Fortran : vient de Formula Translation et le premier langage de programmation et devenu ancien aujourd'hui. Le développement de Fortran avait commencé en 1954 par John Backus et les membres de l'entreprise IBM. Le manuel de référence de Fortran a été publié le 15 Octobre 1956 et le compilateur est apparu en 1957.

Un petit historique des versions de Fortran :

FORTRAN - 1954
FORTRAN II - 1958
FORTRAN III - 1958 (*Never released to public)*
FORTRAN IV - 1961
FORTRAN 66 - 1972
FORTRAN 77 - 1980
Fortran 90 - 1991
Fortran 95 - 1997
Fortran 2003 - Published as ISO/IEC in 2004.
Fortran 2008 - Published as ISO/IEC in September 2010.
Fortran 2018 - Formerly known as Fortran 2015, Fortran 2018 is still in development.

GO : Go, également connu sous le nom de golang, est un langage de programmation informatique dont le développement a commencé en 2007 chez Google, et il a été présenté au public en 2009.

Les trois principaux développeurs de Go chez Google étaient Robert Griesemer, Rob Pike et Ken Thompson. Leur objectif était de créer un langage, vaguement basé sur la syntaxe du C, qui éliminerait les «déchets superflus» de langages tels que C ++. En conséquence, Go évite de nombreuses fonctionnalités d'autres langages modernes, telles que la surcharge de méthodes et d'opérateurs, l'arithmétique des pointeurs et l'héritage de types.

Go n'est pas un langage de forme libre: ses conventions spécifient de nombreux détails de mise en forme, y compris la façon dont l'indentation et les espaces doivent être utilisés. Le langage exige qu'aucune de ses variables déclarées ou bibliothèques importées ne soit inutilisée, et toutes les instructions de retour sont obligatoires.

Haskell : Haskell est un langage de programmation introduit pour la première fois en 1990. Il s'agit d'un langage polyvalent nommé d'après Haskell Curry, un mathématicien américain célèbre pour ses contributions au domaine de la logique combinatoire.

Majeur évolution :

- Haskell 98, qui a été publié à la fin de 1997, comprenait une bibliothèque standard spéciale à des fins d'enseignement et un cadre pour les extensions futures.
- Haskell Prime, lancé en 2006, est un processus continu, formel et ouvert pour affiner la spécification du langage Haskell.
- Haskell 2010 a ajouté une fonctionnalité connue sous le nom de FFI (interface de fonction étrangère), qui permettait aux programmes Haskell d'utiliser les fonctionnalités d'autres langages de programmation.

Hollerith tabulating machine : elle est aussi appelé tabulating machine. C'est une machine électrique inventait par Herman Hollerith.

Implémentation : L'application est « traduite » dans une forme interprétable par le dispositif sur lequel est censée fonctionner. Cette étape correspond normalement à la programmation, c'est-à-dire à l'écriture du code et à la génération de tous les éléments nécessaires au fonctionnement (ex : éléments graphiques,…).

Interface utilisateur : en anglais GUI ou Graphic User Interface. Présente des éléments visibles sur l'interface et permet aux utilisateurs de fournir des inputs et/ou de recevoir les outputs du programme.

Java : Initialement connu sous le nom de chêne, Java est un langage de programmation orienté objet développé par James Gosling et d'autres collaborateurs de Sun Microsystems. Il a été présenté au public pour la première fois en 1995 et est largement utilisé pour créer des applications Internet et d'autres logiciels. Aujourd'hui, Java est maintenue et détenu par Oracle.

Quand utilisé sur Internet, Java permet de télécharger et d'utiliser des applets via un navigateur, permettant au navigateur d'exécuter une fonction ou une fonctionnalité qui n'est normalement pas disponible. Contrairement à JavaScript, les utilisateurs doivent télécharger ou installer l'applet ou le programme avant de pouvoir utiliser le programme Java.

L'exemple d'applet Java ci-dessous de Sun permet de tester si Java est installé sur votre ordinateur. Si Java est installé sur votre ordinateur, des informations supplémentaires sur la version Java installée et votre système d'exploitation s'affichent. Si rien ne s'affiche, Java n'est pas installé sur votre ordinateur ou le plug-in Java de votre navigateur est désactivé ou n'est pas installé.

Java est également utilisée comme langage de programmation pour de nombreux logiciels, jeux et modules complémentaires. Quelques exemples des programmes les plus largement utilisés écrits en Java ou qui utilisent Java incluent la suite Adobe Creative, Eclipse, Lotus Notes, Minecraft, OpenOffice, Runescape et Vuze.

Javascript : Développé par Brendan Eich et connu à l'origine sous le nom de LiveScript, le langage de programmation JavaScript a été renommé en 1995 et est maintenant une marque d'Oracle.

JavaScript est un langage de script interprété côté client qui permet à un concepteur Web d'insérer du code dans sa page Web. JavaScript est généralement placé dans un fichier HTML ou ASP et s'exécute directement à partir de la page Web et est aujourd'hui le langage de programmation le plus populaire. JavaScript peut effectuer des tâches plus avancées, telles que l'impression de l'heure et de la date, la création d'un calendrier ou d'autres tâches qui ne sont pas possibles en HTML.

Julia : Développé par Jeff Bezanson, Alan Edelman, Stefan Karpinski et Viral B. Shah, et lancé pour la première fois en 2012, Julia est un langage de programmation de haut niveau utilisé dans le calcul scientifique. Il peut être utilisé pour les calculs de statistiques et l'analyse de données, comme le langage de programmation R. L'une de ses caractéristiques les plus puissantes est la répartition multiple - un type de polymorphisme qui permet aux fonctions de se comporter différemment en fonction du type de données des arguments qu'elles reçoivent.

Interprétation : dans le cas des langages de programmation qui ne sont pas compilés, il est nécessaire qu'un interprète lise et traduise les instructions « en temps réels » (ou en anglais « on run time »).

Langage de bas niveau : peut se considérer plus proche du langage de la machine plutôt que du langage humain. Un langage de bas niveau est plus difficile à apprendre et à utiliser, mais il permet néanmoins plus de possibilités d'interaction avec le hardware de la machine.

Langage de haut niveau : au contraire, plus proche du langage des êtres humains et il est par conséquent plus facile à utiliser. Cependant, cette facilité limite les possibilités d'interagir avec la machine selon ce que le langage met à disposition.

LISP : est abréviation List Processor. Lisp un langage de haut niveau crée par Jon McCarthy en 1958. LISP a été le second plus vieux langage de programmations. Actuellement a été utilisé pour Intelligence artificielle (AI) pour la recherche en 1958 et de nos jours.

LISP a eu plusieurs versions et dérivés :

- **LISP 1** - Initial release of LISP.
- **LISP 1.5** - First widely used and distributed version of LISP.
- **Stanford LISP** - LISP developed at Stanford AI Lab for PDP-10 systems using TOPS-10 operating system.
- **MACLISP** - LISP for PDP-10 and Multics systems.
- **InterLisp** - Lisp for PDP-10 systems running the TENEX operating system.
- **Franz Lisp** - Based on MACLISP for DEC VAX minicomputer.

- **XLISP**
- **PSL** (**Portable Standard Lisp**)
- **ZetaLisp**
- **LeLisp**
- **Scheme**
- **CL** (**Common Lisp**) - Introduced in 1984, Common Lisp is a combination of previous Lisp dialects and a successor to MACLISP.
- **Dylan**
- **EuLisp**
- **SBCL** (**Steel Bank Common Lisp**)
- **ISLISP**
- **ANSI Common Lisp**
- **ACL2**
- **Clojure**
- **GOAL** (**Game Oriented Assembly Lisp**)
- **Arc**
- **LFE** (**Lisp Flavored Erlang**)
- **Nu**
- **Hy**

Lua : Lua est un langage de programmation créé en 1993 par un groupe d'ingénieurs de l'Université pontificale catholique de Rio De Janeiro, au Brésil. Il a été rédigé par nécessité, en raison des barrières commerciales qui existaient à l'époque, ce qui les empêchait financièrement d'acheter des logiciels personnalisés dans d'autres pays. Aujourd'hui, Lua est utilisé dans le monde entier pour de nombreuses applications, notamment les jeux informatiques.

Maintien : L'application est modifiée ou adaptée à des nécessités qui se présentent dans le temps.

Matlab : Abréviation de matrice de laboratoire, MATLAB est un progiciel de calcul et de visualisation, ainsi qu'un langage de programmation de quatrième génération, publié par MathWorks. Il effectue des manipulations matricielles, le traçage de fonctions, l'implémentation d'algorithmes et de nombreuses opérations mathématiques de haut niveau. Son package compagnon, Simulink, est utilisé pour la simulation graphique multi-domaine.

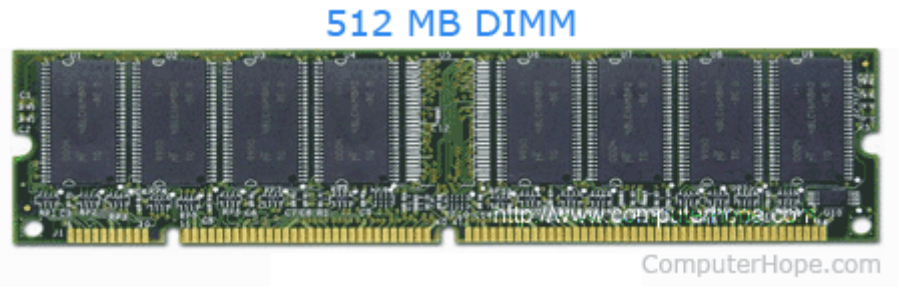
MATLAB a été initialement écrit à la fin des années 1970 par Cleve Moler, le président du département d'informatique de l'Université du Nouveau-Mexique. Moler voulait donner à ses étudiants un moyen d'utiliser des progiciels d'algèbre linéaire et de calcul matriciel sans avoir à apprendre FORTRAN. Aujourd'hui, il est principalement utilisé par des ingénieurs et des étudiants en algèbre linéaire, avec plus d'un million d'utilisateurs dans le monde.

Modula : Modula puis le successeur Modula-2 est un langage de programmation impérative développait en 1970 par Niklaus Wirth.

Mémoire : La mémoire informatique (en anglais computer memory) est un élément physique capable de mémoriser temporairement des données, on appelle RAM (random access memory), ou permanent, la mémoire ROM (read-only memory).

Qu'est son les types de mémoire ?

Prenons par exemple une mémoire de 512 MB DIMM. Ce module de mémoire est connecté à la mémoire slot sur la carte mère.



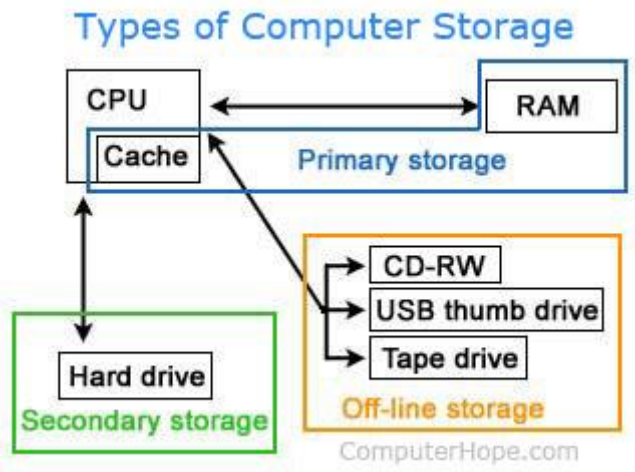
Mémoire Volatile vs. Non-volatile :

La mémoire doit être soit volatile ou non-volatile. La mémoire volatile est une mémoire qui pertes son contenu quand l'ordinateur ou appareil quand on lui coupe l'alimentation. La RAM est par exemple une mémoire volatile. Si ton ordinateur s'éteint ou redémarre donc tu auras perdu les informations.

La mémoire non-volatile a pour abréviation par NVSRAM est une mémoire qui garde les informations et les données après coupures de l'électricité. On peut citer par exemple EPROM.

Mémoire n'est pas un disque dur :

C'est très différent dans l'utilisation de la mémoire ordinateur par rapport à un disque dur. Cependant deux types de hard drive (mémoire dur) et mémoire de type RAM, on l'appel comme mémoire ou « primary memory » et un disque dur comme « storage » ou « secondary storage ».



Par exemple la RAM peut avoir une capacité de 1 GB à 16 GB et un disque dure plusieurs Giga-octets ou plusieurs Téraoctets.

Les différents types de mémoires sont :

- ROM est classée en trois catégories
    - PROM : ou Programmable ROM (programmable read-only memory) est une puce mémoire qui doit être programmé. Les informations qui sont programmées son permanent (on dit écrit) et ne peut être effacé facilement ou supprimer. PROM a été

développé en premier par Wen Tsing Chow en 1956. Par exemple la PROM est utiliser pour le BIOS de l'ordinateur à une certaine époque et maintenant remplacer par le EEPROM.

- EPROM : est abréviation de Erasable Programmable Read-Only Memory. EPROM est une puce mémoire non-volatile qui a été inventé par Dov Frohman en 1972 qui a été mise en place par INTEL que pour la lecture. Si on expose à la lumière ultraviolet, une EPROM peut-être programmer sinon impossible de réécrire ou accéder à la mémoire sauvegarder. Les puces EPROM sont cadencés par une horloge en Crystal de quartz.
- EEPROM : est abréviation de electrically erasable programmable read-only memory. Le EEPROM est une mémoire PROM qui est effaçable et reprogrammable électriquement. EEPROM a été développé par George Perlegos pour Intel en 1978.

- RAM est classée en six catégories :
  - EDO RAM : est abréviation de Extended Data Out. Ce type de mémoire a été développé en 1995 par Micron qui a été le premier ordinateur Pentium. EDO autorise le CPU à accès de la mémoire de 10 à 15 % plus rapide que le Past Page Memory.
  - SDRAM : est abréviation pour Synchronous DRAM. C'est une mémoire qui est synchrone avec horloge de l'ordinateur et de haute rapidité. Depuis 1993, est utilisé dans les ordinateurs.
  - DDR RAM : est abréviation de double data rate. Cette mémoire a été introduit en 1996 et elle a été remplacé la DDR2. Utilise une horloge et surtout utiliser pour les cartes de vidéo et mémoire ordinateur.
  - DDR2 RAM : est abréviation de double data rate two. Elle est la seconde génération de mémoire DDR de Septembre 2003. Elle est plus rapide que le DDR, une plus grande bande passante, moins consommation électrique.
  - DDR3 RAM : est abréviation de double data rate three. Le DDR3 est un type de DRAM de Juin 2007 qui succède à DDR2. DDR3 est une puce avec un bus d'horloge de 400 MHz jusqu'à 1066 MHz. Une plus grande capacité de 1 à 24 GB et consomme moins énergie de 30%. DDR3 RAM est constitué de 204 pins.
  - DDR4 RAM : est abréviation pour double data rate four. Elle a été introduite en Septembre 2014.

MUMPS : Ce langage de programmation a été développé en 1966 par Neil Pappalardo à Masashusetts General Hospital à Boston.

Nim : Initialement nommé Nimrod, Nim est un langage de programmation informatique introduit en 2008 qui impose un niveau élevé de sécurité des types de données. Nimrod est utilisé dans les projets de développement logiciel qui nécessitent des limites strictes sur la façon dont la mémoire d'un système est utilisée.

Oberon : est un langage de programmation écrit en 1986 par Niklaus Wirth.

Objectif-C : Objective-C est un langage de programmation orienté objet développé pour la première fois au milieu des années 1980 par les ingénieurs Brad Cox et Tom Love. Il est basé sur le langage de programmation C et utilise un système de passage de messages dérivé du langage de programmation Smalltalk. Objective-C a été licencié par NeXT en 1988 et est le langage de programmation utilisé dans la majorité des logiciels Apple aujourd'hui. Développeur initial de Perl.

Objets : On dit souvent en informatique qu'un langage est « orienté aux objets ». Un objet en programmation est un élément qui possède des caractéristiques (appelées propriétés) et qui peut normalement exécuter certaines fonctions (appelés méthodes).

OCaml : OCaml, également connu sous le nom d'Objective Caml (Object-Oriented Categorical Abstract Machine Language), est une version orientée objet du langage de programmation Caml. Depuis son introduction en 1996, Ocaml est devenu la principale implémentation de Caml. Il utilise un interpréteur interactif, mais inclut un compilateur pour créer des exécutables binaires. Il a influencé d'autres langages de programmation largement utilisés, notamment F # et Scala.

Opérateurs : Les opérateurs permettent de manipuler ou comparer des valeurs, notamment des variables. Parmi les opérateurs utilisés fréquemment dans tout langage de programmation on identifie :

- Les opérateurs *mathématiques* : addition, soustraction, multiplication, division, etc.
- Les opérateurs de *comparaison* : (égalité, différence, majeur ou mineur)
- Les opérateurs logiques (AND ou OR)

Pascal : Pascal est un langage de programmation de haut niveau développait par Niklaus Wirth en 1971 et qui donne le nom du Mathématicien Blaise Pascal.

PHP : Créé par Rasmus Lerdorf en 1994 et publié publiquement le 8 juin 1995, PHP, qui est l'abréviation de PHP: Hypertext Preprocessor, est un langage de script interprété côté serveur. Il a été conçu pour créer des pages Web dynamiques et des pages Web qui fonctionnent efficacement avec des bases de données.

Prolog : est abréviation de programing in logic ou en Français programmation et logique. Prolog a été développé par Alain Colmeraur et ces collègues de Marseille en 1972. Prolog est utilisé par IA.

Programmation : permet de résoudre un problème (ou un besoin) de manière automatisée grâce à l'application d'un algorithme. On utilise souvent cette équation pour définir un programme informatique : Programme = Algorithme + Données

Punch card : est aussi appelé Hollerith cards ou IBM cards. C'est une carte en papier avec des trous qui sont crées à la main ou avec une machine comme un ordinateur.

R : Le langage de programmation R est un langage de programmation informatique et un environnement logiciel pour le calcul statistique. Les statisticiens et les mineurs de données l'utilisent largement pour effectuer des analyses complexes.

Racket : Racket est un langage de programmation à usage général dérivé de Lisp. On l'appelle un «langage de programmation programmable» parce que son système de macros intégré permet à un programmeur de tout redéfinir sur le langage lui-même. Pour cette raison, Racket est souvent utilisé pour concevoir et implémenter des langages de programmation personnalisés pour les besoins spécifiques d'un projet. Racket a été développé à l'origine dans le cadre du projet PLT ("équipe de langage de programmation") par l'informaticien Matthias Felleisen en 1995. Son intention était de créer un langage qui aiderait les programmeurs novices à en apprendre davantage sur le potentiel des ordinateurs.

Reconnaissance de pattern : L'action d'identifier des similarités entre problèmes ou à l'intérieur du même problème donnée.

Reai : Reia est un langage de programmation à usage général et orienté objet introduit pour la première fois en 2008. Sa syntaxe est similaire à Ruby et il fonctionne sur la machine virtuelle Erlang. Ses principales utilisations sont l'informatique simultanée et distribuée et l'écriture de programmes hautement tolérants aux pannes. Il prend en charge plusieurs paradigmes de programmation, y compris la programmation déclarative, impérative et fonctionnelle.

Ruby : Ruby est un langage de programmation open source orienté objet développé par Yukihiro Matsumoto. La première version du langage (0,95) est sortie en 1995 et en 2011, la version 1.9.3 est sortie.

Ruby gagne en popularité et un framework appelé Ruby on Rails a contribué à augmenter son utilisation pour la programmation Web. Le langage Ruby est complètement orienté objet, en ce sens que tout est objet. Par exemple, même les types de données les plus élémentaires, comme les entiers, ont des méthodes et des variables d'instance. Cela offre une plus grande capacité à utiliser le

chaînage de méthodes, où de nombreuses lignes de code peuvent être consolidées en une seule. Par exemple, si vous souhaitez utiliser trois méthodes différentes sur une chaîne, une solution consiste à écrire plusieurs lignes de code, comme indiqué ci-dessous.

Rust : Rust est un langage de programmation multi-paradigme qui prend en charge les styles de programmation fonctionnels, impératifs, orientés objet et à action simultanée. Il a commencé comme un projet personnel de Graydon Hoare, employé de Mozilla, et en cinq ans, il est devenu un projet open source acceptant les contributions de centaines de bénévoles. Sa description officielle est la suivante: "fonctionne à une vitesse fulgurante, empêche presque tous les plantages et élimine les courses de données". La version 1.0.0 alpha a été publiée par Mozilla Research le 9 janvier 2015.

Scala : Scala est un langage de programmation introduit en 2003. Il est orienté objet, fortement typé statiquement et prend en charge un paradigme de programmation fonctionnel. Scala a été créé pour répondre à de nombreuses critiques du langage de programmation Java. En tant que tel, il a une syntaxe similaire à Java et se compile en bytecode Java. Cependant, il existe plusieurs différences clés inspirées des langages tels que Scheme et Haskell.

S**cheme** : est un langage de programmation qui est une variante de Lisp. Il était crée en 1975 par Guy Steele et Gerry Sussman a MITS's Artificial Intelligence Lab. Ce langage introduit la programmation fonctionnelle et algorithme récursive en particulier.

Simula : est un langage de programmation développer en 1965 par des ingénieurs développeurs de Norvégien Kristen Nygaard et Ole-Johan Dahl. Simula est basé sur le langage ALGOL 60 et suppose la simulation de système complexe. Simula est le premier langage de programmation a utilisé les objets, classes, sous-classes, et héritage (inheritance). Simula est considéré comme le premier langage de programmation orienté-objet.

Structure de contrôle : Les structures de contrôle permettent d'exécuter seulement certaines instructions d'un programme selon la vérification d'une ou plusieurs conditions.

Syntaxe : Il dispose d'une syntaxe, c'est-à-dire des règles d'associations de mots et de la ponctuation.

Tableaux (array) : Les tableaux sont des listes indexées d'éléments qui partagent normalement une certaine relation sémantique pour appartenir à la liste.

Test : tout ce qui se passe une fois que l'application/logiciel est disponible. L'application est contrôlée dans le but qu'elle fonctionne de la manière imaginée/souhaitée.

Variables : sont des éléments qui sont censés changer à un moment donné dans la logique de l'application.

- *Modifier la valeur associée à travers la logique de l'application* : c'est l'une des manières les plus utilisées pour garder une trace de l'évolution de l'application suite à son utilisation ;
- *Réutiliser dans les différents endroits du code source une valeur sans avoir à la répéter littéralement à chaque fois ;*
- *Déclaration d'une variable :* cela correspond à créer une variable, c'est-à-dire à insérer pour la première fois une variable dans la logique de l'application ;
- *Affectation/assignation d'une variable :* cela correspond à associer une valeur à la variable, ce qui fait normalement grâce à un opérateur d'affectation, par exemple le symbole =.

Selon le langage de programmation utilisé, il faut également définir à quel type de valeur/de données une variable va être associée. Parmi les types de données plus fréquentes, on identifie :

- ✓ Les *suites de caractères* (string) : elles sont utilisées pour représenter du texte (des mots, des phrases, etc.) ;
- ✓ Les *chiffres* (nombre entier, à virgule flottante, etc.) : ils sont utilisés surtout avec des opérateurs mathématiques ;
- ✓ Les *valeurs booléennes* (en anglais : boolean) : elles sont des valeurs dichotomiques (soit vrai, soit faux) ;
- ✓ Les *tableaux* (array) : ils sont utilisés pour créer des listes avec des indices qui permettent de récupérer la valeur associée ;
- ✓ Les *objets* : ils sont des conteneurs qui peuvent inclure souvent tout type de données, u compris de sous-objets, des variables (ex : des propriétés), ou des fonctions (ex : méthodes).

Visual Basic : VB ou Visual Basic est un langage de programmation développé par Microsoft avec l'aide d'Alan Cooper et a été publié pour la première fois en mai 1991. Visual Basic a été conçu avec le programmeur débutant à l'esprit et pour les programmeurs qui avaient besoin de développer des éléments visuels dans leurs programmes. Dans Visual Basic, les utilisateurs peuvent faire glisser et repositionner des éléments visuels tels que des fenêtres, des boutons et des formulaires, puis créer des événements et des déclencheurs pour ces éléments.

XML : Abréviation de langage de balisage extensible, XML est une spécification développée par le W3C à partir de la recommandation du 10 février 1998. XML est similaire au HTML en ce sens que XML utilise des balises pour baliser un document, permettant au navigateur d'interpréter et d'afficher les informations. Cependant, contrairement au HTML, le langage XML est illimité (extensible). Il permet aux balises de se définir et peut décrire le contenu au lieu d'afficher uniquement le contenu d'une page. En utilisant XML, d'autres langages tels que RSS et MathML ont été créés, même des outils comme XSLT ont été créés à l'aide de XML.

Références Biographiques

A. Sites web :

1. https://www.computerhope.com/
2. https://www.scriptol.fr

B. Livres et Magazines

1. [Coding] Magazine N°1 p52-61

C. Cours et articles

1. Les langages informatiques (les évolutions) de Michel Riguidel
2. Histoire et évolution des langages programmation par Denis Sureau
3. Programming Language History and Future de Jean E. Sammet