Présentation des langages de programmation

Wikipédia, Xavier Dufillon & Gaëtan Croquefer

28 mars 2017

Sommaire

Table des figures

1 Résumé

En informatique, un langage de programmation est une notation conventionnelle destinée à formuler des algorithmes et produire des programmes informatiques qui les appliquent. D'une manière similaire à une langue naturelle, un langage de programmation est composé d'un alphabet, d'un vocabulaire, de règles de grammaire et de significations.[1] [2]
Les langages de programmation permettent de décrire d'une part les structures des données qui seront manipulées par l'appareil informatique, et d'autre part d'indiquer comment sont effectuées les manipulations, selon quels algorithmes. Ils servent de moyens de communication par lesquels le programmeur communique avec l'ordinateur, mais aussi avec d'autres programmeurs ; les programmes étant d'ordinaire écrits, lus, compris et modifiés par une équipe de programmeurs.[3]
Un langage de programmation est mis en œuvre par un traducteur automatique : compilateur ou interpréteur. Un compilateur est un programme informatique qui transforme dans un premier temps un code source écrit dans un langage de programmation donné en un code cible qui pourra être directement exécuté par un ordinateur, à savoir un programme en langage machine ou en code intermédiaire[2], tandis que l'interpréteur réalise cette traduction « à la volée ».
Les langages de programmation offrent différentes possibilités d'abstraction, et une notation proche de l'algèbre, permettant de décrire de manière concise et facile à saisir les opérations de manipulation de données et l'évolution du déroulement du programme en fonction des situations. La possibilité d'écriture abstraite libère l'esprit du programmeur d'un travail superflu, notamment de prise en compte des spécificités du matériel informatique, et lui permet ainsi de se concentrer sur des problèmes plus avancés.[2]
Chaque langage de programmation supporte un ou plusieurs styles de programmation – paradigmes. Les notions propres au paradigme font partie du langage de programmation, permettant au programmeur d'exprimer dans le langage de programmation une solution qui a été imaginée selon ce paradigme. Les premiers langages de programmation ont été créés dans les années 1950. De nombreux concepts de l'informatique ont été lancés par un langage, avant d'être améliorés et étendus dans les langages suivants. La plupart du temps la conception d'un langage de programmation a été fortement influencée par l'expérience acquise avec les langages précédents.[4]

2 Définition

Un langage de programmation est construit à partir d'une grammaire formelle, qui inclut des symboles et des règles syntaxiques, auxquels on associe des règles sémantiques. Ces éléments sont plus ou moins complexes selon la capacité du langage. Les modes de fonctionnement et de définition de la complexité d'un langage de programmation sont généralement déterminés par leur appartenance à l'un des degrés de la Hiérarchie de Chomsky.
Sous un angle théorique, tout langage informatique peut être qualifié de langage de programmation s'il est Turing-complet c'est-à-dire qu'il permet de représenter toutes les fonctions calculables au sens de Turing et Church (en admettant néanmoins pour exception à la théorie que la mémoire des ordinateurs n'est pas un espace infini)[5].

- 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[2]. 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[6], 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 « // »[6].

- 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.

3 Utilisation

Un langage de programmation offre un cadre pour élaborer des algorithmes et exprimer des diagrammes de flux[7][6]. Il permet en particulier de décrire les structures des données qui seront manipulées par l'appareil informatique et quelles seront les manipulations. Un langage de programmation sert de moyen de communication avec l'ordinateur mais aussi entre programmeurs : les programmes étant d'ordinaire écrits, lus et modifiés par une équipe de programmeurs[3].
Un langage de programmation offre un ensemble de notions qui peuvent être utilisées comme primitives pour développer des algorithmes. Les programmeurs apprécient que le langage soit clair, simple, et unifié, qu'il y ait un minimum de notions qui peuvent être combinées selon des règles simples et régulières. Les qualités d'un langage de programmation influent sur la facilité avec laquelle les programmes pourront être écrits, testés, puis plus tard compris et modifiés[6].
Les langages de programmation offrent différentes possibilités d'abstraction, et une notation proche de l'algèbre, permettant de décrire de manière concise et facile à saisir les opérations de manipulation de données et l'évolution du déroulement du programme en fonction des situations. Cette possibilité d'écriture abstraite libère l'esprit du programmeur d'un travail superflu, et lui permet de se concentrer sur des problèmes plus avancés[2].

La facilité d'utilisation, la portabilité et la clarté sont des qualités appréciées des langages de programmation. La facilité d'utilisation, qui dépend de la syntaxe, du vocabulaire et des symboles, influence la lisibilité des programmes écrits dans ce langage et la durée d'apprentissage. La portabilité permet à un programme écrit pour être exécuté par une plateforme informatique donnée (un système d'exploitation) d'être transféré en vue d'être exécuté sur une autre plateforme[7].
Les programmeurs apprécient que la syntaxe permette d'exprimer la structure logique inhérente au programme. Un des soucis en programmation est d'éviter des pannes, qu'il soit possible de les détecter, les éviter et les rectifier ; ceci est rendu possible par des mécanismes internes des langages de programmation. Des vérifications implicites sont parfois effectuées en vue de déceler des problèmes[7].
Les programmeurs apprécient qu'un langage de programmation soit en ligne avec les bonnes pratiques de programmation et d'ingénierie, qu'il encourage la structuration du programme, facilite la maintenance des programmes et qu'il dissuade voire interdise les mauvaises pratiques[7]. L'utilisation de l'instruction goto, par exemple, qui existe depuis les premiers langages de programmation, est considérée comme une mauvaise pratique. Son utilisation est déconseillée, voire impossible dans les langages de programmation récents[8].
L'alignement sur les standards industriels, la possibilité d'utiliser des fonctionnalités écrites dans un autre langage de programmation et l'exécution simultanée de plusieurs threads sont des possibilités appréciées des langages de programmation[7].

3.1 Notions courantes

Un langage de programmation permet de décrire les structures des données qui seront manipulées par l'appareil informatique et quelles seront les manipulations. Il offre un ensemble de notions telles que les instructions, les variables, les types, et les procédures, qui peuvent être utilisées comme primitives pour développer des algorithmes[9].

- Une instruction :
  Un ordre donné à un ordinateur[10].

- 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[9].

- 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[9]. 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[9].

- 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 et les arbres[11].

- 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[3]...
  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[3].

- 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[3].

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].

4 Paradigmes

Un paradigme est un style de programmation[12]. Chaque paradigme amène une technique différente de programmation ; une fois qu'une solution a été imaginée par le programmeur selon un certain paradigme, un langage de programmation qui suit ce paradigme permettra de l'exprimer[13]. Impératif, déclaratif, fonctionnel, logique, orienté objet, concurrent, visuel, événementiel, et basé web sont des paradigme de programmation[12]. 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[13]. 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[12]. Les langages de programmation contemporains de 2013 permettent typiquement d'adopter plusieurs styles de programmation[12]. Voir Fig.1 pour une vue d'ensemble des différents paradigmes existants.

4.1 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. L'abstraction est réalisée à l'aide de procédures auxquelles sont transmises des données. Il existe une procédure principale, qui est la première à être exécutée, et qui peut faire appel à d'autre procédures spécialisées pour effectuer certaines tâches (commande d'appareil, calculs, etc.). Le C, le Pascal, le Fortran et le COBOL sont des exemples de langage de programmation qui implémentent le paradigme impératif[13].

4.2 Déclaratif

Les deux paradigmes déclaratifs sont : fonctionnel et logique. En paradigme fonctionnel le programmeur décrit des fonctions mathématiques. En para-

digme logique il décrit des prédicats : une déclaration qui peut être vraie ou fausse[12]. Une machine abstraite effectue ensuite les opérations nécessaires pour calculer le résultat de chaque fonction et chaque prédicat. Dans ce paradigme une variable contient le résultat d'un calcul et ne peut pas être modifiée par assignation[12].

4.2.1 Fonctionnel

Le paradigme fonctionnel est basé sur l'idée d'évaluer une formule, et d'utiliser le résultat pour autre chose, selon le modèle du lambda-calcul. Tous les traitements sont faits en évaluant des expressions et en faisant appel à des fonctions, et l'exécution étape par étape n'est pas possible dans le paradigme fonctionnel. Le résultat d'un calcul sert de matière première pour le calcul suivant, et ainsi de suite, jusqu'à ce que toutes les fonctions aient produit un résultat[13]. Le paradigme fonctionnel a été introduit par le langage Lisp à la fin des années 1960. En 2013 des langages tels que Ruby et Scala supportent plusieurs styles dont le paradigme fonctionnel[12].

4.2.2 Logique

Le paradigme logique est basé sur l'idée de répondre à une question par des recherches sur un ensemble, en utilisant des axiomes, des demandes et des règles de déduction. L'exécution d'un programme est une cascade de recherches de données dans un ensemble, en faisant usage de règles de déduction. Les données obtenues, et associées à un autre ensemble de règles 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[12].

4.3 Orienté objet

Le paradigme orienté objet 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[12]. Le paradigme orienté objet permet d'associer fortement les données avec les procédures[13]. 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[12]. 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, C++ , Swift et Java sont des langages de programmation en paradigme orienté objet[13].

4.4 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[12]. 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 processeurs multicore et permettent de créer et manipuler des threads[12]. Plus récemment, on a vu apparaître des langages plus modernes intégralement orientés vers la gestion de la concurrence, comme le langage Go.

4.5 Visuel

Dans la grande majorité des langages de programmation, le code source est un texte. Ce qui rend difficile d'exprimer des objets en deux dimensions[12]. 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 le langage Smalltalk, dans le but de faciliter la programmation des interfaces graphiques[12].

4.6 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[12]. 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[12]. 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[12].

4.7 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[12]. 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[12].

- Mise en œuvre
  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 :

- 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.

- 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[14].

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[15].

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[16].
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[17].

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[17] qui sert alors d'intermédiaire entre les possibilités offertes par la plateforme informatique et les constructions propre au langage de programmation[18].
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[19]. Les runtime servent également à mettre en œuvre certaines fonctionnalités avancées des langages de programmation telles que le ramasse-miettes, ou la réflexion[17].
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[20].

4.8 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[17].

- Un ramasse-miettes
  Un mécanisme qui supprime les variables inutilisées et libère l'espace mémoire qui leur avait été réservé[21].

- 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[22].

- 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[23].

- 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[24].

- La liaison tardive
  Le procédé de liaison (anglais 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[25], elle est dite tardive lorsque l'opération de liaison est effectuée très tard, juste avant que l'emplacement concerné ne soit utilisé[26].

- 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[27].

5 Historique

Bien que la notion de programme apparaisse progressivement au cours de la deuxième moitié du XIXe siècle, les premiers langages de programmation n'apparaissent qu'autour de 1950. Chacun pouvant créer son propre langage, il est impossible de déterminer le nombre total de langages existant à l'heure actuelle. Pour un aperçu des languages les plus courants de nos jours, voir Fig.2 ou Fig.3.

6 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.

6.1 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.

6.2 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.

6.3 Pour rendre la programmation plus difficile

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. Par exemple, le Piet permet de programmer à l'aide d'images matricielles.
On peut également citer le Brainfuck qui est un langage minimaliste et Turing-complet (8 instructions seulement). Il est prévu pour tourner sur une machine de Turing avec un compilateur de seulement 171 octets. Un extrême dans la difficulté à mettre en œuvre est sans doute le Malbolge, créé en 1998, si difficile à comprendre quand il est arrivé qu'il a fallu deux ans au premier programme Malbolge pour apparaître.

6.4 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

6.4.1 Langages synchrones

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

6.4.2 Langages à vocation pédagogique

Les pseudo-codes (comme le Langage K) ont généralement un but uniquement pédagogique.
Logo est un langage fonctionnel simple à apprendre.
Dans les années 1990, c'est le langage BASIC qui 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.

6.4.3 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

6.4.4 Langages pour la statistique

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

6.4.5 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...

6.4.6 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.

6.4.7 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.

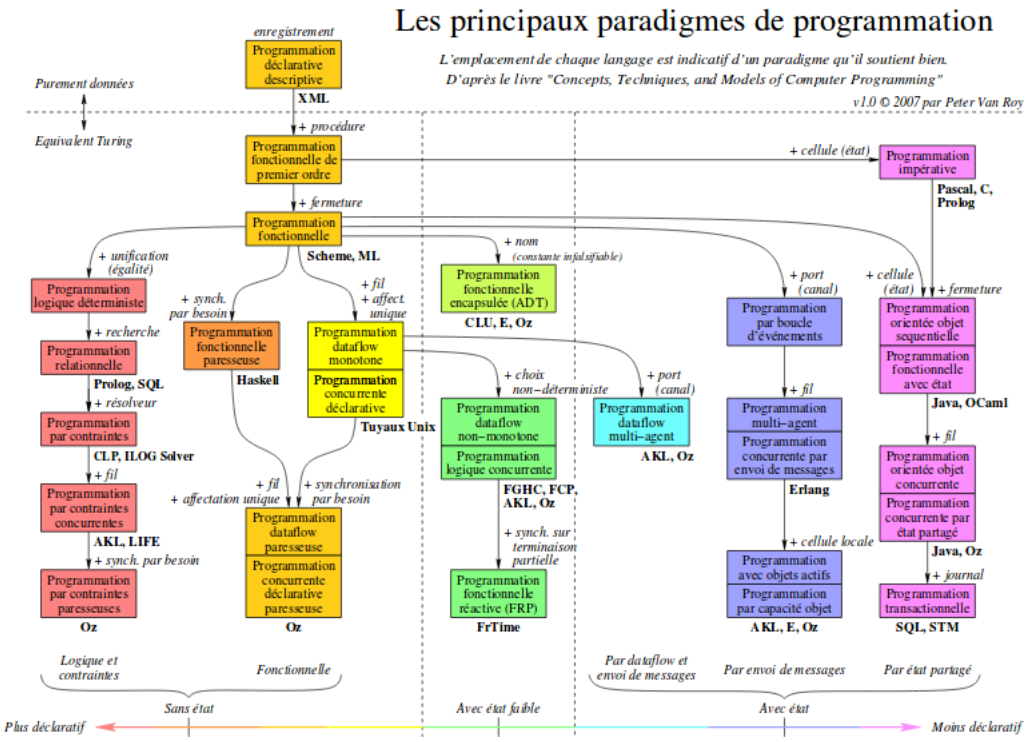

FIGURE 1 – Les différents paradigmes de programmation

Source[28]

Classement				
2016	2015	Langage	Utilisation	Changements
1	2	Java	21,465%	+5,94%
2	1	C	16,036%	-0,67%
3	4	C++	6,914%	+0,21%
4	5	C#	4,707%	-0,34%
5	8	Python	3,854%	+1,24%
6	6	PHP	2,706%	-1,08%
7	16	Visual Basic .NET	2,582%	+1,51%
8	7	JavaScript	2,565%	-0,71%
9	14	Assembleur	2,095%	+0,92%
10	15	Ruby	2,045%	+0,92%
11	9	Perl	1,841%	-0,42%
12	20	Delphi/Object Pascal	1,786%	+0,95%
13	17	Visual Basic	1,684%	+0,61%
14	25	Swift	1,363%	+0,62%
15	11	MATLAB	1,228%	-0,16%
16	30	Pascal	1,194%	+0,52%
17	82	Groovy	1,182%	+1,07%
18	3	Objective-C	1,074%	-5,88%
19	18	R	1,054%	+0,1%
20	10	PL/SQL	1,016%	-1%
...				

FIGURE 2 – Classement des langages de programmation

Source[29]

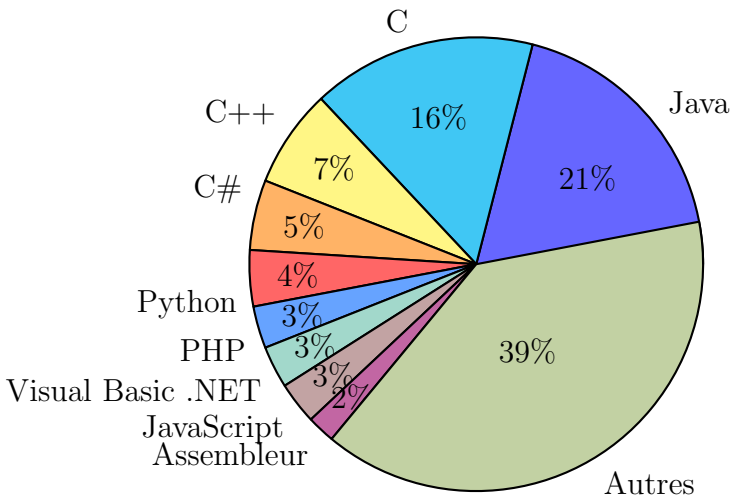

FIGURE 3 – Utilisation des principaux langages de programmation

Références

[1] Maurizio Gabbrielli et Simone Martini. *Programming Languages : Principles and Paradigms*. Springer, 2010.

[2] Kenneth C. Louden et Kenneth A. Lambert. *Programming Languages : Principles and Practices*. Cengage Learning, 2011.

[3] William Sims Bainbridge. *Berkshire Encyclopedia of Human-computer Interaction, vol. 2*. Berkshire Publishing Group LLC, 1994.

[4] David Anthony et William Findlay Watt. *Programming Language Design Concepts*. John Wiley & Sons, 2004.

[5] *Premiers pas vers une ontologie générale des programmes informatiques*, 2007.

[6] Seema Kedar. *Programming Paradigms And Methodology*. Technical Publications, 2002.

[7] I. T. L. Education Solutions Limited. *Introduction To Information Technology*. Pearson Education India, 2005.

[8] J. Barlow et A. R. Barnett. *Computing for Scientists : Principles of Programming with Fortran 90 and C++*. John Wiley and Sons, 1998.

[9] Seema Kedar et Sanjay Thakare. *Principles of Programming Languages*. Technical Publications, 2009.

[10] Ada programming. Technical report.

[11] Krishnamurthy. *Data Structures Using C*. Tata McGraw-Hill Education, 2008.

[12] Arvind Kumar Bansal. *Introduction to Programming Languages*. CRC Press, 2013.

[13] Jana. *Java And Object-Oriented Programming Paradigm*. PHI Learning Pvt. Ltd., 2008.

[14] D.A. Godse et A.P. Godse. *Computer Architecture & Organisation*. Technical Publications, 2007.

[15] Dhamdhere. *Systems Programming and Operating Systems*. Tata McGraw-Hill Education, 2011.

[16] Tmh. *Computer Science Vii (Tn)*. Tata McGraw-Hill Education, 2007.

[17] Michael L. Scott. *Programming Language Pragmatics*. Morgan Kaufmann, 2009.

[18] Matthew Adams Ian Griffiths. *Net Windows Forms in a Nutshell*. O'Reilly Media, Inc., 2003.

[19] Raghavan. *Prin Of Compiler Design*. Tata McGraw-Hill Education, 2010.

[20] Diomidis Spinellis et Georgios Gousios. *Beautiful Architecture : Leading Thinkers Reveal the Hidden Beauty in Software Design*. O'Reilly Media Inc., 2009.

[21] Chris Smith. *Programming F#*. O'Reilly Media, Inc., 2009.

[22] Jeff Friesen et Geoff Friesen. *Java 2 by Example*. Que Publishing, 2002.

[23] Harold Davis. *Visual Basic 6 Secrets*. Que Publishing, 1998.

[24] Ken Jones et Jeffrey Hobbs Brent B. Welch. *Practical Programming in Tcl and Tk*. Prentice Hall Professional, 2003.

[25] Sibsankar Haldar et Alex A. Aravind. Operating systems,pearson education india. 2010.

[26] Amit Singh. *Mac OS X Internals : A Systems Approach*. Addison-Wesley Professional, 2007.

[27] Kishori Sharan. Harnessing java 7 : A comprehensive approach to learning java 7. 2011.

[28] Une taxonomie des principaux paradigmes de programmation. `https://www.info.ucl.ac.be/~pvr/paradigmes.html`.

[29] Tiobe index. `https://www.tiobe.com/tiobe-index/`.