WIKIPÉDIA

Programmation impérative

En informatique, la **programmation impérative** est un paradigme de programmation qui décrit les opérations en séquences d'instructions exécutées par l'ordinateur pour modifier l'état du programme. Ce type de programmation est le plus répandu parmi l'ensemble des langages de programmation existants, et se différencie de la programmation déclarative (dont la programmation logique ou encore la programmation fonctionnelle sont des sous-ensembles).

Sommaire

Langages impératifs et processeurs

La quasi-totalité des processeurs qui équipent les ordinateurs sont de nature impérative : ils sont faits pour exécuter une suite d'instructions élémentaires, codées sous forme d'*opcodes* (pour *operation codes*). L'ensemble des *opcodes* forme le langage machine spécifique à l'architecture du processeur. L'état du programme à un instant donné est défini par le contenu de la mémoire centrale à cet instant.

Les langages de plus haut niveau utilisent des variables et des opérations plus complexes, mais suivent le même paradigme. Les recettes de cuisine et les vérifications de processus industriel sont deux exemples de concepts familiers qui s'apparentent à de la programmation impérative ; de ce point de vue, chaque étape est une instruction, et le monde

physique constitue l'état modifiable. Puisque les idées de base de la programmation impérative sont à la fois conceptuellement familières et directement intégrées dans l'architecture des microprocesseurs, la grande majorité des langages de programmation est impérative.

Instructions de la base impérative

La plupart des langages de haut niveau comporte cinq types d'instructions principales :

1. la séquence d'instructions
2. l'assignation ou affectation
3. l'instruction conditionnelle
4. la boucle
5. les branchements

Séquence d'instructions

Une *séquence d'instructions*, (ou bloc d'instruction) désigne le fait de faire exécuter par la machine une instruction, puis une autre, etc., en séquence. Par exemple

$$\text{ouvrirConnexion; envoyerMessage; fermerConnexion;}$$

est une séquence d'instructions. Cette construction se distingue du fait d'exécuter en parallèle des instructions.

Instructions d'assignation

Les *instructions d'assignation*, en général, effectuent une opération sur l'information en mémoire et y enregistrent le résultat pour un usage ultérieur. Les langages de haut niveau permettent de plus l'évaluation d'expressions complexes qui peuvent consister en une combinaison d'opérations arithmétiques et d'évaluations de fonctions et l'assignation du résultat en mémoire. Par exemple:

$$x \leftarrow 2 + 3;$$

assigne la *valeur* $2 + 3$, donc 5, à la variable de nom x.

Instructions conditionnelles

Les *instructions conditionnelles* permettent à un bloc d'instructions de n'être exécuté que si une condition prédéterminée est réalisée. Dans le cas contraire, les instructions sont ignorées et la séquence d'exécution continue à partir de l'instruction qui suit immédiatement la fin du bloc. Par exemple

$$\textit{si}\ \text{connexionOuverte}\ \textit{alors}\ \text{envoyerMessage;}$$

n'enverra le message que si la connexion est ouverte.

Instructions de bouclage

Les *instructions de bouclage* servent à répéter une suite d'instructions un nombre prédéfini de fois (voir Boucle_for), ou jusqu'à ce qu'une certaine condition soit réalisée. Par exemple

$$\textit{tantque}\ \text{connexionNonOuverte}\ \textit{alors}\ \text{attendreUnPeu;}$$

bouclera jusqu'à ce que la connexion soit ouverte.

Il se trouve que ces quatre constructions permettent de faire tous les programmes informatiques possibles, elles permettent de faire un système Turing-complet.

Branchements sans condition

Les *branchements sans condition* permettent à la séquence d'exécution d'être transférée à un autre endroit du programme. Cela inclut le saut, appelé « goto » (*go to*, /ɡəʊ tuː/, « aller à ») dans de nombreux langages, et les sous-programmes, ou appels de procédures. Les instructions de bouclage peuvent être vues comme la combinaison d'un branchement conditionnel et d'un saut. Les appels à une fonction ou une procédure (donc un Sous-programme) correspondent à un saut, complété du passage de paramètres, avec un saut en retour.

Historique

Langage machine

Les langages impératifs les plus anciens sont les langages machine des premiers ordinateurs. Dans ces langages, le jeu d'instructions est minimal, ce qui rend la mise en œuvre matérielle plus simple — on maîtrise directement ce qui se passe en mémoire —, mais gêne la création de programmes complexes.

A-0

Le premier compilateur – un programme destiné à vérifier un programme au préalable et à le traduire en langage machine – dénommé A-0, fut écrit en 1951 par Grace Murray Hopper.

Fortran

Fortran, développé par John Backus chez IBM à partir de 1954, fut le premier langage de programmation capable de réduire les obstacles présentés par le langage machine dans la création de programmes complexes. Fortran était un langage compilé, qui autorisait entre autres l'utilisation de variables nommées, d'expressions complexes, et de sous-programmes. Après plusieurs révisions du langage, Fortran est toujours utilisé dans le milieu scientifique pour la qualité de ses bibliothèques numériques et sa grande rapidité, ce qui en fait le langage informatique ayant eu la plus grande longévité.

Algol

Les deux décennies suivantes virent l'apparition de plusieurs autres langages de haut niveau importants. ALGOL, développé en 1958 par un consortium américano-européen pour concurrencer FORTRAN, qui était un langage propriétaire, fut l'ancêtre de nombreux langages de programmation d'aujourd'hui.

COBOL

COBOL (1960) est un langage pour la programmation des applications de gestion développé avec plusieurs objectifs : d'une part avoir un langage standardisé, avec des sources portables sur des matériels différents, d'autre part avoir des sources lisibles et vérifiables par des non-spécialistes de l'informatique. Dans cet objectif, il été défini avec une syntaxe proche de l'anglais. Le langage a ensuite évolué pour intégrer la programmation structurée (COBOL 85), et la programmation orientée objet (2000). Le parc énorme d'applications COBOL existantes dans les grandes entreprises assure sa longévité.

BASIC

Le langage BASIC (1963) a été conçu comme une version simplifiée de FORTRAN à but éducatif, destinée aux débutants et interactive. Sa simplicité et le fait que BASIC soit interprété facilitaient grandement la mise au point des programmes, ce qui lui conféra rapidement une grande popularité, malgré la pauvreté de ses constructions.

Malheureusement, cette pauvreté même devait mener à une quantité de programmes non structurés et donc difficilement maintenables. Après un article de Edsger Dijkstra dénonçant les ravages de BASIC, la réputation de BASIC comme langage pour l'enseignement de la programmation déclina, au profit de Pascal.

Pascal

Dans les années 1970, le Pascal fut développé par Niklaus Wirth, dans le but d'enseigner la programmation structurée et modulaire. Pascal dérivait d'une proposition faite par N. Wirth (et refusée) pour l'évolution du langage ALGOL. Il combine les constructions de base de la programmation structurée (boucles tant-que, répéter-jusqu'à et boucle avec compteur), la possibilité de définir ses propres types de donnée, dans un ensemble élégant (servi par un grand nombre de types prédéfinis : ensemble, énumérations, intervalle), qui lui assura un succès durable comme langage d'initiation (en remplacement de BASIC). Par la suite, Niklaus Wirth fut à l'origine de Modula-2, Modula-3, et d'Oberon, les successeurs de Pascal.

C

À la même époque, Dennis Ritchie créa le langage C aux laboratoires Bell, pour le développement du système Unix. La puissance du C, permettant grâce aux pointeurs de travailler à un niveau proche de la machine, ainsi qu'un accès complet aux primitives du système, lui assura un succès qui ne s'est jamais démenti depuis.

Une des causes du succès du langage C par rapport aux autres langages procéduraux de la même génération vient de son mode de distribution : les universités américaines pouvaient acheter une licence au prix de 300 dollars pour toute l'université et tous ses étudiants. [réf. nécessaire]

Ada

En 1974, le Département de la Défense des États-Unis cherchait un langage dont le cahier des charges mettait l'accent sur la sûreté d'exécution, pour tous ses besoins futurs. Le choix se porta sur Ada, langage créé par Jean Ichbiah chez CII-Honeywell Bull, dont la spécification ne fut complétée qu'en 1983. Le langage a connu plusieurs révisions, la dernière en date remontant à 2012.

Smalltalk

Dans les années 1980, devant les problèmes que posaient la complexité grandissante des programmes, il y eut un rapide gain d'intérêt pour la programmation orientée objet. Smalltalk-80, conçu à l'origine par Alan Kay en 1969, fut présenté en 1980 par le Palo Alto Research Center de la compagnie Xerox (États-Unis).

C++ et Objective C

À partir des concepts objet, Bjarne Stroustrup, chercheur aux Bell Labs, conçut en 1985 une extension orientée objet de C nommée C++. Parallèlement, une extension à C moins ambitieuse, mais inspirée de Smalltalk avait vu le jour, Objective C. Le succès d'Objective C, notamment utilisé pour le développement sur les stations NeXT et Mac OS X, est resté faible par rapport à C++.

Perl, Tcl, Python, PHP, Java, JavaScript

Dans les décennies 1980 et 1990, de nouveaux langages impératifs interprétés ou semi-interprétés doivent leur succès au développement de scripts pour des pages web dynamiques et les applications client-serveur. On peut citer dans ces catégories Perl (Larry Wall, 1987), Tcl (John Ousterhout, 1988), Python (Guido van Rossum, 1990), PHP (Rasmus Lerdorf, 1994), Java (Sun Microsystems, 1995), JavaScript (Brendan Eich, Netscape Navigator, 1995).

Langages fonctionnels et langages de programmation logique

Les langages de programmation impératifs doivent être distingués d'autres types de langages, les langages fonctionnels et les langages de programmation logique. Les langages fonctionnels, tels que Haskell ou ML, ne sont pas des suites d'instructions et ne s'appuient pas sur l'idée d'état global, mais au contraire tendent à s'extraire de ce modèle pour se placer à un niveau plus conceptuel (qui a ses fondations dans le lambda-calcul). Les langages de programmation logiques, tels que Prolog, se concentrent sur ce qui doit être calculé, et non comment le calcul doit être effectué.

Annexes

Liens externes

- Un synopsis de l'histoire des langages de programmation (http://www.levenez.com/lang/)
- Un cours en ligne de l'Université Paris XIII (http://www-lipn.univ-paris13.fr/~recanati/docs/L1-ProgC/Partie1.pdf)

Sur les autres projets Wikimedia :

Algorithmique impérative, sur Wikibooks