

Les langages de programmation

Qu’est-ce que c’est ?

Réalisé par :

François Maltais

Présenté à :

Godefroy Borduas

Dans le cadre du cours :

Outils de gestion et de soutien (420-C13-IN - Hiver 2023)

Cégep du Vieux Montréal

2023-02-07

Table des matières

[Contextualisation 4](#_Toc126660837)

[Principe 4](#_Toc126660838)

[Définition 5](#_Toc126660839)

[Les règles de syntaxe 5](#_Toc126660840)

[Le vocabulaire 5](#_Toc126660841)

[La sémantique 5](#_Toc126660842)

[L'alphabet 5](#_Toc126660843)

[Paradigmes 6](#_Toc126660844)

[Impératif/Procédural 6](#_Toc126660845)

[Orienté objet 7](#_Toc126660846)

[Concurrent 8](#_Toc126660847)

[Historique 8](#_Toc126660848)

[Performances 8](#_Toc126660849)

[Pour la consommation d'énergie 9](#_Toc126660850)

[Pour le temps d'exécution 9](#_Toc126660851)

[Pour la consommation maximale de mémoire 9](#_Toc126660852)

Table des figures

[Figure 1 Exemple de code source dans le langage JavaScript 4](#_Toc126660153)

[Figure 2 Photographie de Ken Thompson et Dennis Ritchie 7](#_Toc126660154)

[Figure 3 Exemple de code source en Simula 7](#_Toc126660155)

Table des codes

[Code 1 « Hello Word » en C 7](#_Toc126660280)

[Code 2 « Hello World » en C++ 8](#_Toc126660281)

[Code 3 « Hello World » en Go 8](#_Toc126660282)

# Contextualisation

Ce document présente un exemple de mise en forme pour le projet Word du cours Outils de gestion et de soutien du Cégep du Vieux Montréal. Pour plus d'information sur le travail demandé, se référer à l’énoncé du projet.

Le contenu de ce document est une adaptation libre de la page Wikipédia sur les langages de programmation (accessible via [ce lien](https://fr.wikipedia.org/wiki/Langage_de_programmation)).

# Principe

Figure Exemple de code source dans le langage JavaScript

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, de significations, mais aussi d'un environnement de traduction censé rendre sa syntaxe compréhensible par la machine.

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.

# 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.

## 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. 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 telles 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 programme2. 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).

# 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. 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 permettent d'adopter plusieurs paradigmes de programmation à condition que ceux-ci soient compatibles.

## Impératif/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 est un exemple de langage procédural. Ce dernier a été inventé au cours de l'année 1972 dans les Laboratoires Bell. Il était développé en même temps qu'Unix par Dennis Ritchie et Ken Thompson. Son créateur déclara que :

Pour les infrastructures technologiques, le langage C sera difficile à remplacer.

- Dennis Ritchie

#include <iostream>

int main() {

std::cout << "Hello World!";

return 0;}

Code 1 « Hello Word » en C

Figure 2 Photographie de Ken Thompson et Dennis Ritchie

## Orienté objet

Figure 3 Exemple de code source en Simula

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.

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 de grands programmes. Le langage C++ est un autre exemple de langage exploitant l'orienté objet. Son créateur, Bjarne Stroustrup, introduira, grâce au paradigme objet, la bibliothèque standard.

La bibliothèque standard évite aux programmeurs d'avoir à réinventer la roue.

- Bjarne Stroustrup

#include <iostream>

int main() {

std::cout << "Hello World!";

return 0;

}

Code 2 « Hello World » en C++

## 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 multicœurs 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.

package main

import "fmt"

func main() {

fmt.Println("hello world")

}

Code 3 « Hello World » en Go

# 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.

# Performances

Six chercheurs de trois universités portugaises ont mené une étude comparative de 27 langages de programmation, intitulée « Energy Efficiency Across Programming Languages ». Ils ont étudié la consommation d'énergie, le temps d'exécution et l'utilisation de la mémoire. Pour obtenir un ensemble de programmes comparables, les chercheurs ont exploré le Computer Language Benchmarks Game (CLBG).

Le tableau obtenu présente les résultats globaux (en moyenne) pour la consommation d'énergie (Energy), le temps d'exécution (Time) et la consommation maximale de la mémoire (Mb) normalisés par rapport au langage le plus efficace pour le critère mesuré.

Les paragraphes suivants présentent les cinq meilleurs langages pour trois catégories. Les résultats sont présentés dans un format normalisé.

## Pour la consommation d'énergie

1. C : 1,00
2. Rust : 1
3. ,03
4. C++ : 1,34
5. Ada : 1,70
6. Java : 1,98

## Pour le temps d'exécution

1. C : 1,00
2. Rust : 1,04
3. C++ : 1,56
4. Ada : 1,85
5. Java : 1,89

## Pour la consommation maximale de mémoire

1. Pascal : 1,00
2. Go : 1,05
3. C : 1,17
4. Fortran : 1,24
5. C++ : 1,34