— *It looked like a good thing to do.*

Dennis Ritchie, à propos de l'invention du langage C

— *The more general aim was to design a language in which I could write programs that were both efficient and elegant. Many languages force you to choose between those two alternatives.*

Bjarne Stroustrup, à propos de l'invention du langage C++

1

Notes sur le C++

Le langage C est apparu au cours de l'année 1972 dans les Laboratoires Bell. Il fut développé en même temps qu'UNIX par Dennis Ritchie et Ken Thompson. L'objectif était d'écrire un langage de programmation permettant le développement rapide de systèmes d'exploitation (OS). Dennis Ritchie fit évoluer un langage existant –le langage B– dans une nouvelle version suffisamment différente pour qu'elle soit appelée C. Le succès du C fut rapide, en raison de sa portabilité [1] et des performances du code écrit.

Dix ans plus tard, Bjarne Stroustrup, intéressé par la programmation orientée objet qui commençait à se répendre chez les universitaires, développa le langage C++, successeur du C, alors qu'il travaillait dans le laboratoire de recherche Bell d'AT&T. Il s'agissait en l'occurrence d'améliorer le langage C en lui ajoutant une composante objet (le C++ s'appelait alors C with classes). La proximité du C++ avec le C, couplée aux facilités qu'il offrait firent qu'il fut rapidement adopté.

Le C++ a été largement adopté par la communauté des développeurs et peut être considéré comme le langage de référence des années 80 et 90. Dans les années 2000, des langages plus faciles [2] mais aux performances comparables (Java et C# par exemple) sont préférés pour le développement de nouveaux projets. Cependant, le C++ reste le langage de référence : de très nombreux projets débutés dans les années 80/90 ont été codés en C++ et n'ont pas été portés vers des langages plus récents. En finance par exemple, il reste le langage le plus utilisé aujourd'hui (2012) car de très nombreuses bibliothèques toujours en production ont été développées en C++, même si les nouveaux projets ont tendance à regarder du côté de langages plus récents, objets ou fonctionnels.

1. C'est-à-dire la possibilité d'utiliser un programme tapé sur un ordinateur donné et un système d'exploitation donné sur un autre ordinateur et un autre système d'exploitation

2. tout est question de point de vue évidemment

Plus proche de la machine que les langages plus récents, le C++ permet au prix d'une complexité plus forte un meilleur contrôle sur la machine et donc sur les performances.

Le C++ est un langage multi-paradigme, pouvant à la fois être procédural[3] et orienté objet, permettant de la programmation générique très poussée (jusqu'au template méta-programming en fait), mais aussi un début de programmation fonctionelle. Le C++ est un langage particulièrement difficile, suffisamment proche du langage machine pour que nous ayons besoin parfois de connaissances hardwares pour optimiser notre code, mais aussi suffisamment abstrait pour exprimer une réelle complexité dans les concepts manipulés. C'est un langage plutôt lourd, plus conçu pour développer des OS ou des gros projets que des petits scripts pour administrer une machine.

Enfin, comme nous le verrons dans ce polycopié, le C++ est un langage bâtard. Pour qu'un langage soit cohérent, il est nécessaire que le principal de ses concepts clefs soit défini dès sa création, qu'une vision à long terme du langage soit explicitée dès le départ. Le C++ est né à une période où de nombreux concepts émergeaient. Construit "par dessus" le C, il reprend l'intégralité de la spec du C (pour convaincre les développeurs C de passer sous C++) et a ajouté au fur et à mesure de l'avancement des concepts, des couches supplémentaires au langage. L'enrichissement de ces concepts a mené à une sorte d'hydre, un langage puissant mais peu structuré qui s'est développé dans de nombreuses voies différentes. C'est donc à la fois un langage de référence et d'expérimentation.

A qui s'adresse le C++ ? Pourquoi enseigner et utiliser le C++ aujourd'hui ?

Le C++ est un langage très performant en vitesse d'execution (surtout lorsqu'il n'abuse pas des concepts introduits entre le C et le C++), particulièrement adapté à la création de bibliothèques scientifiques ou d'applications temps réels comme dans le cadre de radars, de contrôleur de tableaux de bord dans les avions, de trading très haute fréquence, etc.

Cette haute performance a un prix : le C++ est un langage complexe, laborieux, peu adapté pour de nombreuses choses naturelles dans des langages plus récents comme le développement web par exemple.Sauf contraintes énormes sur la performance, très peu de vrais projets informatiques ont donc de raisons valables de se lancer aujourd'hui dans ce langage.

L'intérêt pédagogique du C++ est soumis à débat : d'un côté, il est suffisamment difficile pour forcer l'étudiant à assimiler de nombreux concepts qu'il serait possible de contourner dans d'autres langages. De l'autre, il est actuellement souvent un mauvais choix de technologie pour monter un projet logiciel. En raison des contraintes industrielles actuelles, les jeunes diplômés de l'ENSAE se trouvent cependant amenés à travailler sur du code en C++, ou sur des langages pour lesquels la transition est assez aisée (notamment le Java et le C#). C'est pour cette raison que l'enseignement à l'ENSAE reste pour quelques

3. Plus par souci de compatibilité avec le C qu'autre chose d'ailleurs, je vous conseille vivement de ne pas coder en procédural...

années encore en C++. J'invite cependant les étudiants désireux de monter un ambitieux projet logiciel à préférer un des langages plus récents précédemment cités.

Ce document est la version écrite du cours d'introduction au C++ dispensé à l'ENSAE. Dans le cas de l'informatique, il est difficile de proposer le même contenu en cours magistral et dans un polycopié. La syntaxe du C++ ou la compilation peuvent être rapidement expliquées à l'oral par l'exemple, alors que la même explication dans ce document prend nécessairement des volumes plus considérables. Ceci a pour conséquence que les détails et les contraintes techniques ont mécaniquement dans ce document une place relative plus importante qu'en cours. C'est malheureusement au prix d'une place relative plus faible pour les concepts de programmation objet qui me semblent bien plus fondamentaux que la syntaxe du langage par exemple.

J'ai choisi d'écrire ce document en dépit de l'existence de centaines de documents similaires, disponibles en livre imprimé ou sur le web. A mes yeux, trop de ces documents n'étaient pas adaptés à l'enseignement du C++ pour des ingénieurs. En effet, il existe effectivement des "bibles", comme les ouvrages de Stourstrup [?] ou [?], très complètes et techniques mais qui s'adressent à des lecteurs déja familiers avec la programmation, ou alors des livres de "vulgarisation" trop peu ambitieux pour des ingénieurs. Avec ce document, j'espère fournir un cours qui soit adapté à la fois aux attentes des étudiants, mais aussi à leur vitesse de compréhension.

Dans la mesure du possible, j'ai consenti à de multiples ellipses dans les premiers chapitres, prenant le risque d'être imprécis, afin d'aller au plus vite au coeur du langage. Ces ellipses volontaires sont aussi un moyen de dépoussiérer l'apprentissage du C++ en choisissant de ne pas expliquer les concepts du C présents dans le C++ mais qui n'ont plus beaucoup de sens aujourd'hui (je pense par exemple aux structures, dont je fais le moins mention possible).

J'ai pris le parti d'organiser ce cours en deux grandes parties qui filent la métaphore des langues vivantes : je présente dans une première grande partie la grammaire et l'orthographe du C++. La deuxième partie, de loin la plus importante et sur laquelle vous serez jugés, sera celle du style. Il s'agit dans cette deuxième partie de comprendre comment architecturer votre code, de comprendre pourquoi certains designs sont bien supérieurs à d'autres.

Enfin, un langage de programmation ne s'apprend que d'une seule manière : par la *pratique*. Répétons-le : sans écrire de programme, il n'est pas possible de maîtriser un langage. Pour ce faire, des Travaux Dirigés, avec un corrigé détaillé vous sont fournis sur le wiki de l'école (Pamplemousse).