Avant-propos

1 Historique de C++

La programmation orientée objet (en abrégé P.O.O.) est dorénavant universellement reconnue pour les avantages qu'elle procure. Notamment, elle améliore largement la productivité des développeurs, la robustesse, la portabilité et l'extensibilité de leurs programmes. Enfin, et surtout, elle permet de développer des composants logiciels entièrement réutilisables.

Un certain nombre de langages dits "langages orientés objet" (L.O.O.) ont été définis de toutes pièces pour appliquer les concepts de P.O.O. C'est ainsi que sont apparus dans un premier temps des langages comme Smalltalk, Simula ou Eiffel puis, plus récemment, Java. Le langage C++, quant à lui, a été conçu suivant une démarche quelque peu différente par B. Stroustrup (AT&T) ; son objectif a été, en effet, d'adjoindre au langage C un certain nombre de spécificités lui permettant d'appliquer les concepts de P.O.O. Ainsi, C++ présente-t-il sur un vrai L.O.O. l'originalité d'être fondé sur un langage répandu. Ceci laisse au programmeur toute liberté d'adopter un style plus ou moins orienté objet, en se situant entre les deux extrêmes que constituent la poursuite d'une programmation classique d'une part, une pure P.O.O. d'autre part. Si une telle liberté présente le risque de céder, dans un premier temps, à la facilité en mélangeant les genres (la P.O.O. ne renie pas la programmation classique - elle l'enrichit), elle permet également une transition en douceur vers la P.O.O pure, avec tout le bénéfice qu'on peut en escompter à terme.

De sa conception jusqu'à sa normalisation, le langage C++ a quelque peu évolué. Initialement, un certain nombre de publications de AT&T ont servi de référence du langage. Les dernières en date sont : la version 2.0 en 1989, les versions 2.1 et 3 en 1991. C'est cette dernière qui a servi de base au travail du comité ANSI lequel, sans la remettre en cause, l'a enrichie de

quelques extensions et surtout de composants standard originaux se présentant sous forme de fonctions et de classes génériques qu'on désigne souvent par le sigle S.T.L[1]. La norme définitive de C++ a été publiée par l'ANSI et par l'ISO en 1998, et a fait l'objet d'une révision publiée en 2003 sous la référence ISO 14882:2003.

2 Objectifs et structure de l'ouvrage

Cet ouvrage a été spécifiquement conçu pour tous ceux qui, possédant déjà une pratique du langage C[2], souhaitent maîtriser la programmation orientée objet en C++. Il s'adresse à la fois aux étudiants, aux développeurs et aux enseignants en informatique.

Conçu sous forme d'un cours complet, il expose progressivement à la fois :

- les différentes notions fondamentales de la P.O.O. et la façon dont elles s'expriment en C++ (classes et objets, méthodes, constructeur, destructeur, héritage, polymorphisme),
- les spécificités, non orientées objet, du langage C++, c'est-à-dire celles qui permettent à C++ d'être un C amélioré (référence, argument par défaut, surdéfinition de fonctions, fonctions en ligne, espaces de noms...),
- les spécificités orientées objet du C++ : fonctions amies, surdéfinition d'opérateurs, patrons de classes et de fonctions, héritage multiple, flots, bibliothèque standard.

Chacune de ces notions est illustrée systématiquement par un programme complet, assorti d'un exemple d'exécution montrant comment la mettre en œuvre dans un contexte réel. Celui-ci peut également servir à une prise de connaissance intuitive ou à une révision rapide de la notion en question, à une expérimentation directe dans votre propre environnement de travail ou encore de point de départ à une expérimentation personnelle.

Les chapitres les plus importants ont été dotés d'exercices[3] comportant :

- des suggestions de manipulations destinées à mieux vous familiariser avec votre environnement ; par effet d'entraînement, elles vous feront probablement imaginer d'autres expérimentations de votre cru ;
- des programmes à rédiger ; dans ce cas, un exemple de correction est fourni en fin de volume.

L'aspect didactique a été privilégié, sans pour autant nuire à l'exhaustivité de l'ouvrage. Nous couvrons l'ensemble de la programmation en C++, des notions fondamentales de la P.O.O. jusqu'aux aspects très spécifiques au langage (mais néanmoins fondamentaux), afin

1. *Standard Template Library*.

2. Le cas échéant, on pourra trouver un cours complet de langage C dans *Programmer en langage C*, ou une référence exhaustive de sa norme dans *Langage C*, du même auteur, chez le même éditeur.

3. De nombreux autres exercices peuvent être trouvés dans *Exercices en langage C++* du même auteur, chez le même éditeur.

de rendre le lecteur parfaitement opérationnel dans la conception, le développement et la mise au point de ses propres classes. C'est ainsi que nous avons soigneusement étudié les conséquences de la liberté qu'offre C++ de choisir le mode de gestion de la mémoire allouée aux objets (automatique ou dynamique)[1]. De même, nous avons largement insisté sur le rôle du constructeur de recopie, ainsi que sur la redéfinition de l'opérateur d'affectation, éléments qui conduisent à la notion de "classe canonique". Toujours dans le même esprit, nous avons pris soin de bien développer les notions indispensables que sont la ligature dynamique et les classes abstraites, lesquelles débouchent sur la notion la plus puissante du langage qu'est le polymorphisme. De même, la S.T.L. a été étudiée en détail, après avoir pris soin d'exposer préalablement d'une part les notions de classes et de fonctions génériques, d'autre part celles de conteneur, d'itérateur et d'algorithmes qui conditionnent la bonne utilisation de la plupart de ses composants.

3 L'ouvrage, la norme de C++, C et Java

Cet ouvrage est entièrement fondé sur la norme ANSI/ISO du langage C++. Dès le début, le lecteur est sensibilisé aux quelques incompatibilités existant entre C++ et C, de sorte qu'il pourra réutiliser convenablement en C++ du code écrit en C. D'autre part, compte tenu de la popularité du langage Java, nous avons introduit de nombreuses remarques titrées *En Java*. Elles mettent l'accent sur les différences majeures existant entre Java et C++. Elles seront utiles au lecteur qui, après la maîtrise du C++, souhaitera aborder l'étude de Java.

Cet ouvrage correspond en fait à une refonte des éditions précédentes de *Programmer en C++*. Nous continuons d'y mentionner les apports de la norme par rapport à la version 3 du langage, publiée en 1991, ainsi que les quelques différences avec les versions antérieures. Ces remarques, initialement prévues pour faciliter l'utilisation d'anciens environnements de programmation, deviennent de moins en moins pertinentes ; mais, dans la mesure où elles ne pertubent pas l'apprentissage du langage, nous avons préféré les conserver pour leur caractère historique ; en particulier, elles mettent en avant les points délicats du langage pour lesquels la genèse a été quelque peu difficile.

1. Certains langages objet, dont Java, gèrent tous les objets de manière dynamique.