Le C++ au pied de la lettre

Le C++ fascine par l'infini des réalisations et manières de programmer qu'il permet. Il renvoie inévitablement à des réflexions de fond sur modélisation et bonnes pratiques de programmation. Cependant, il ne porte pas de réponse en lui-même, il fournit seulement une large diversité de moyens pour élaborer, réfléchir et exprimer un programme. Il revient aux programmeurs de décider des attitudes méthodologiques afin de garantir lisibilité, efficacité et sécurité du code.

Au premier abord, le C++ déconcerte parce qu'il n'est pas un langage purement et strictement objet. Il hérite du langage C qui lui procure une base pour tout ce qui concerne les algorithmes. Variables, structures de contrôle (if, switch, boucles, fonctions), structures, tableaux et pointeurs, tout ce qui établit le langage C, sa syntaxe et l'ensemble de ses fonctionnalités, se trouvent incorporés dans le langage C++. En général, un programmeur C++ est aussi un très bon programmeur C parce qu'il est impossible de programmer en C++ et d'ignorer le C. Pour autant, ce sont deux langages distincts et souvent l'utilisation des mêmes outils de base prend des sens particuliers d'un langage à l'autre. En effet, certains outils, comme les pointeurs et les paramètres de fonctions, peuvent s'interpréter différemment dans des situations créées par l'un ou l'autre des deux langages. Ces différences d'interprétation des mêmes outils augmentent généralement la difficulté pour les apprenants lorsqu'il est question de passer d'un langage à l'autre.

Néanmoins, on trouve souvent une certaine unité des deux langages dans des programmes écrits par des programmeurs chevronnés qui sont capables d'envisager les fonctionnalités dont ils ont besoin en circulant de façon cohérente entre les deux approches possibles. L'espace très large qui ressort laisse finalement entrevoir que C et C++ ne constituent pas deux langages, mais un seul langage suffisamment souple pour autoriser des approches conceptuelles et méthodologiques différentes. Cet aspect peut aussi être déroutant pour des apprenants, et il est certain que l'aisance de programmer simultanément avec ou sans objet n'est pas aisée à transmettre. La possibilité de conjuguer les deux approches pose un vrai problème pour l'enseignement.

D'ailleurs, cette difficulté n'est pas réservée aux apprenants. La diversité conceptuelle suscitée par le C++ peut également poser des problèmes méthodologiques à des équipes de développement. La plupart des équipes auront tendance à se tailler un C++ sur mesure, par exemple en limitant certaines de ses fonctionnalités ou au contraire en en systématisant l'emploi. Ces différents contrôles et orientations de la pratique du C++ constituent pour ainsi dire des dialectes C++. Ces dialectes peuvent donner naissance à de nouveaux langages, le C# en est un exemple vraiment intéressant.

Afin de relever le défi d'un enseignement C++, ce livre se fonde sur deux idées inspirées par des structures de données, l'une partagée entre C et C++, l'autre spécifique au C++. Il s'agit de l'union et de l'héritage. Ce livre prend C et C++ comme une union, mais il considère aussi que le C++ hérite du langage C, le C constituant en quelque sorte sa classe de base. Les développeurs confirmés apprécieront. Cette double possibilité aide à relever le défi d'une approche C et C++ qui accède à l'ensemble des potentialités du langage C++.