Le guide du C++ moderne

de débutant à développeur

par
Benoît Vittupier
Mehdi Benharrats
avec la contribution de Luc Hermitte
et de Zeste de Savoir

D-BOOKER
éditions

Le guide du C++ moderne • de débutant à développeur
par Benoît Vittupier, Mehdi Benharrats, avec la contribution de Luc Hermitte, et de Zeste de Savoir

ISBN (imprimé) : 978-2-8227-1176-0

Publié par les Éditions D-BookeR, 229 rue Solférino, 59000 Lille
www.d-booker.fr
contact@d-booker.fr

Les exemples (téléchargeables ou non), sauf indication contraire, sont propriété des auteurs.

Conception de la couverture : d'après une création de Marie Van Der Marlière (www.marie-graphiste.com)
Visuel de couverture : Réalisé par putracetol (#22795181 sur VectorStock.com)
Mise en page : générée sous Calenco avec des XSLT développées par la société NeoDoc (www.neodoc.biz)
Diffusion : AFPU-D (https://www.afpu-diffusion.fr)
Distribution : DILISCO (https://dilisco-diffusion.centprod.com)

Dépôt légal : Novembre 2025
Édition : 2
Version : 2.0

Table des matières

1

C’est décidé, je m’y mets !

Voilà, vous êtes décidé, vous voulez apprendre à programmer en C++. Mais quelques questions continuent à vous trotter dans la tête : Concrètement, le langage C++, c’est quoi ? Et puis pourquoi choisir le C++ ? Que vais-je savoir à l’issue de ce livre ? Y a-t-il des conditions pour pouvoir suivre ce livre ?

1.1. Le C++, qu’est-ce que c’est ?

Faisons un plongeon dans l’histoire et revenons dans les années 1970. À cette époque, Dennis Ritchie, programmeur aux laboratoires AT&T aux États-Unis, invente le langage C, conçu pour programmer le système d’exploitation UNIX. Ce langage devint très populaire à tel point qu’il est encore beaucoup utilisé aujourd’hui, notamment pour l’écriture de Linux. Un peu plus tard, au début des années 1980, Bjarne Stroustrup, lui aussi développeur aux laboratoires AT&T, décide de prendre ce langage C comme base et de lui ajouter des fonctionnalités issues d’un autre langage appelé Simula. Cette première bouture est appelée le C *with classes*. Finalement, en 1983, son créateur, estimant que le nom de son langage est trop réducteur au vu de tous les ajouts faits par rapport au C, décide de le renommer C++. Mais l’histoire ne s’arrête pas là. Au contraire, le C++ continue d’évoluer à tel point qu’on décide au début des années 1990 de *le normaliser*, c’est-à-dire d’en établir les règles officielles. Ce travail de longue haleine s’acheva en 1998 ; cette version est ainsi souvent nommée C++98. Ensuite, en 2003, des corrections ont été apportées et l’on obtint C++03.

Puis de nouveau un chantier titanesque est mis en place pour améliorer encore plus le C++, ce qui aboutit huit ans plus tard, en 2011, à la sortie de C++11, jugée par beaucoup de développeurs comme la renaissance du C++. À partir de ce moment est établi un rythme de trois ans entre chaque nouvelle mise à jour de C++, ce qui a donné C++14, C++17, C++20 et C++23. À l’heure où j’écris, C++26 est sur le point d’être finalisé.

Pour ceux qui lisent l’anglais, vous trouverez plus d’informations sur l’histoire du langage, ses choix de conception, les améliorations apportées par les différentes normes, etc. dans la publication de Bjarne Stroustrup *Thriving in a Crowded and Changing World: C++ 2006–2020* [https://d-booker.jo.my/cpp-hist].

C++ HISTORIQUE ET C++ MODERNE

Beaucoup de programmeurs utilisent le terme C++ *historique* pour désigner les normes C++98 et C++03 et le terme C++ *moderne* pour parler de C++11 et au-delà. Cette distinction n'est pas fausse, mais la différence entre le C++ historique et le C++ moderne tient moins à une question de norme qu'à des bonnes pratiques, de meilleures façons de faire, etc.

Par exemple, on peut utiliser la dernière norme mais continuer à écrire son code comme on le faisait à l'époque du commencement de C++, dans un style proche du langage C. À l'inverse, on peut être bloqué pour une raison quelconque avec un ancien compilateur et donc être forcé d'utiliser C++98 ou C++03 et pourtant écrire du code moderne, en accord avec les bonnes pratiques issues du C++ moderne.

1.2. Pourquoi apprendre le C++ ?

- Sa *popularité* : le C++ est utilisé dans de nombreux projets importants (citons LibreOffice, 7-zip ou encore KDE). Il est au programme de beaucoup de formations informatiques. Il possède une communauté très vaste, beaucoup de documentation et d'aide, surtout sur l'internet anglophone.
- Sa *rapidité* : il offre un grand contrôle sur la rapidité des programmes. C'est cette caractéristique qui fait de lui un des langages de choix pour les programmes scientifiques, par exemple. Il s'associe notamment très bien avec des langages comme Python, pour écrire des blocs de code qui doivent s'éxécuter très rapidement.
- Son *ancienneté* : c'est un langage ancien d'un point de vue informatique (30 ans, c'est énorme), ce qui donne une certaine garantie de maturité, de stabilité et de pérennité (il ne disparaîtra pas dans quelques années).
- Son *évolution* : C++11 est un véritable renouveau du C++ historique, qui le rend plus facile à utiliser et plus puissant dans les fonctionnalités qu'il offre aux développeurs. C++14, C++17 et C++20 améliorent encore la chose.
- Il est *multiparadigme* : il n'impose pas une façon unique de concevoir et découper ses programmes mais laisse le développeur libre de ses choix.

Bien entendu, tout n'est pas parfait et le C++ a aussi ses défauts.

- Son *héritage du C* : le C++ est un descendant du langage C, inventé dans les années 1970. Certains choix de conception, adaptés pour l'époque, sont plus problématiques aujourd'hui, et le C++ les traîne avec lui.

- Sa *complexité* : il ne faut pas se le cacher, atteindre une certaine maîtrise du C++ est très long et demandera des années d'expérience, notamment parce que certaines de ses fonctionnalités les plus puissantes requièrent de très bien connaître les bases.
- Sa *bibliothèque standard* : bien qu'elle permette de faire beaucoup de choses (d'ailleurs, nous ne ferons que l'aborder dans ce livre), elle n'offre pas de mécanisme natif pour manipuler des bases de données, faire des programmes en fenêtres, jouer avec le réseau, etc. Par rapport à d'autres langages comme Python, C++ peut paraître plus limité.

1.3. Développer, un métier à part entière

Eh oui ! Créer des logiciels, c'est un métier. Il y a des écoles et des études spécialisées pour ça. C'est que développer, ce n'est pas simplement écrire du code. Il y a aussi des phases de réflexion, de conception, d'écriture, de validation, de tests, de réécriture d'anciennes portions, etc. Et puis, même si ce n'est pas directement lié à la programmation, celle-ci requiert des connaissances en gestion de base de données, en usage du réseau, en management, en gestion de projet, etc. En bref, *être développeur c'est beaucoup de compétences différentes dans des domaines variés.*

La cruelle vérité qui se cache derrière tout ça, c'est que ce livre ne fera pas de vous des experts, ni des développeurs professionnels. Par contre, une fois fini, vous aurez des bases solides pour continuer votre apprentissage. La route du savoir est infinie.

1.4. Votre part du travail

Ce livre est écrit de façon à être le plus clair possible, sans vous noyer sous un flot de détails et d'explications, mais il arrivera parfois que vous ne compreniez pas un morceau de code ou une explication. C'est tout à fait normal, ça fait partie de l'apprentissage. Reprenez le passage à tête reposée, aidez-vous de schémas ou de dessins, demandez de l'aide sur les forums, et vous ne resterez jamais bloqué longtemps.

Par contre, vous devez être prêt à fournir des efforts de votre côté. Cela signifie ne pas vous ruer sur les forums à la première difficulté rencontrée, sans chercher à essayer de la résoudre par vous-même.

1.5. Les mathématiques, indispensables ?

Une des grandes appréhensions, qui revient régulièrement dans la bouche des débutants, est de savoir si les mathématiques sont un prérequis à l'apprentissage de la pro-

grammation. La réponse est non. Il est tout à fait possible d'apprendre à programmer tout en ayant un faible niveau en mathématiques. Ce livre ne requiert aucune connaissance en mathématiques, si ce n'est celle des opérations de base et, pour quelques exemples, du sinus et du cosinus.

Bien sûr, certains aspects de la programmation, comme la sécurité, la cryptographie ou les applications scientifiques demandent un bagage mathématique solide. Mais cela n'entre pas dans le cadre de ce livre.

Par contre, avoir l'esprit logique reste un plus indéniable.

1.6. L'anglais, indispensable ?

À vrai dire, pour suivre ce livre, pas besoin de savoir parler anglais. Même si nous nous reporterons régulièrement à de la documentation écrite dans la langue de Shakespeare, nous ne le ferons jamais sans l'accompagner de quelques explications et éclaircissements. Et quand bien même il resterait quelque chose sur lequel vous butiez, vous pourriez vous servir d'un traducteur automatique.

D'un point de vue plus général, si vous souhaitez continuer dans l'informatique et la programmation, il sera très difficile d'échapper à l'anglais. Beaucoup de cours, de documents, de forums sont en anglais ou en font un usage massif. L'anglais est tout simplement *la langue de l'informatique*. Sachez cependant que l'anglais informatique est simple à comprendre, car souvent écrit par des gens dont ce n'est pas la langue maternelle. Ainsi, inutile d'être bilingue, un bon dictionnaire ou un traducteur suffiront.

1.7. La documentation

En programmation, il y a un réflexe à adopter le plus rapidement possible : si on ne sait pas comment utiliser un outil, il faut aller consulter sa documentation, et ce avant de demander de l'aide sur un forum par exemple. Voici un lien vers une excellente documentation C++ : `https://en.cppreference.com/w/cpp"`. Elle est en anglais, mais pas de souci, nous sommes là avec vous. Au fur et à mesure, nous vous donnerons des liens, nous vous expliquerons comment comprendre et exploiter les informations fournies pour que, par la suite, vous puissiez le faire par vous-même.

Il y a aussi un autre outil, très utile pour rechercher une information et que vous connaissez déjà : les moteurs de recherche (Google, Bing, DuckDuckGo, Qwant, etc.). Sachez aussi que les forums sont une mine d'informations. Vous pouvez par exemple utiliser ceux de Zeste de Savoir, en n'oubliant bien évidemment pas d'utiliser le tag [c++] lors de la création d'un sujet.

Enfin, sachez qu'il existe une référence ultime appelée *la norme*, produite par un organisme de validation international appelé l'*International Organization for Standardization* ou ISO, qui explique tous les détails et les règles du C++ mais qui est un document complexe et très largement hors de portée quand on débute. La dernière version de ce document est sortie en 2023 et explique les règles de fonctionnement du C++ dans sa version de 2023. Nous le mentionnons simplement pour que vous soyez au courant de son existence et que vous ne soyez pas surpris si, lors de vos recherches sur Internet, des réponses s'y rapportent ou citent la norme.

1.8. Récapitulons !

- Le C++ a une longue histoire, mais le C++ moderne a véritablement vu le jour en 2011, avec des améliorations et corrections apportées en 2014, 2017, 2020, 2023 et bientôt 2026.
- Le C++ possède des atouts qui en font un des langages les plus utilisés sur la planète.
- En contrepartie, le C++ est un langage assez complexe qui demande de nombreuses années avant d'être maîtrisé.
- La programmation est une activité parfois complexe, mais très enrichissante et accessible.
- Ce livre se veut le plus accessible possible, mais vous devez jouer le jeu et faire des efforts de votre côté.
- Les mathématiques et l'anglais ne sont pas requis pour suivre ce livre, toutefois un minimum de logique et un niveau acceptable en anglais seront extrêmement utiles par la suite.
- La documentation nous aide à mieux comprendre le langage, son fonctionnement, etc. Elle est en anglais, mais est illustrée par des exemples.