

# Le guide du C++ moderne

---

## de débutant à développeur

par  
Benoît Vittupier  
Mehdi Benharrats  
avec la contribution de Zeste de Savoir

Cet extrait provient du livre *Le guide du C++ moderne* – de débutant à développeur  
écrit par Benoît Vittupier et Mehdi Benharrats – © 2021 Éditions D-Booker

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

ISBN (imprimé) : 978-2-8227-0881-4

Copyright © 2021 Éditions D-BookeR  
Tous droits réservés

Conformément au Code de la propriété intellectuelle, seules les copies ou reproductions strictement réservées à l'usage privé du copiste et non destinées à une utilisation collective ainsi que les analyses et les courtes citations dans un but d'exemple et d'illustration sont autorisées. Tout autre représentation ou reproduction, qu'elle soit intégrale ou partielle, requiert expressément le consentement de l'éditeur (art L 122-4, L 122-5 2 et 3a).

Publié par les Éditions D-BookeR, 229 rue Solférino, 59000 Lille  
[www.d-booker.fr](http://www.d-booker.fr)  
[contact@d-booker.fr](mailto: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](http://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](http://www.neodoc.biz))

Dépôt légal : Août 2021  
Édition : 1  
Version : 1.0

# Table des matières

---

|                                                 |           |
|-------------------------------------------------|-----------|
| <b>Bienvenue ! .....</b>                        | <b>1</b>  |
| 1. Codes sources des exemples .....             | 1         |
| 2. Remerciements .....                          | 1         |
| <b>1. C'est décidé, je m'y mets ! .....</b>     | <b>3</b>  |
| 1.1. Le C++, qu'est-ce que c'est ? .....        | 3         |
| 1.2. Pourquoi apprendre le C++ ? .....          | 4         |
| 1.3. Développer, un métier à part entière ..... | 5         |
| 1.4. Votre part du travail .....                | 5         |
| 1.5. Les mathématiques, indispensables ? .....  | 6         |
| 1.6. L'anglais, indispensable ? .....           | 6         |
| 1.7. La documentation .....                     | 6         |
| 1.8. Récapitulons ! .....                       | 7         |
| <b>Le début du voyage .....</b>                 | <b>9</b>  |
| <b>2. Le minimum pour commencer .....</b>       | <b>11</b> |
| 2.1. Des outils en ligne .....                  | 11        |
| 2.2. Des outils plus poussés .....              | 13        |
| 2.3. Un mot concernant Windows .....            | 17        |
| 2.4. Un mot concernant GNU/Linux .....          | 18        |
| Éditeur de texte .....                          | 18        |
| Compilateur .....                               | 18        |
| 2.5. Récapitulons ! .....                       | 19        |
| <b>3. Rencontre avec le C++ .....</b>           | <b>21</b> |
| 3.1. Compilons notre premier programme .....    | 21        |
| 3.2. Démystifions le code .....                 | 22        |
| Utiliser des fonctionnalités déjà codées .....  | 22        |
| Le point d'entrée .....                         | 22        |
| Voici mes instructions... .....                 | 23        |
| 3.3. Les commentaires .....                     | 24        |
| 3.4. Récapitulons ! .....                       | 25        |
| <b>4. Tout ça est bien variable .....</b>       | <b>27</b> |
| 4.1. Les littéraux .....                        | 27        |
| Les caractères .....                            | 27        |
| Les caractères spéciaux .....                   | 29        |

|                                                                    |           |
|--------------------------------------------------------------------|-----------|
| Les nombres .....                                                  | 30        |
| 4.2. Les variables .....                                           | 32        |
| Comment créer des variables en C++ ? .....                         | 33        |
| Un peu de constance, voyons ! .....                                | 37        |
| Manipulation de variables .....                                    | 38        |
| 4.3. Simple déduction ? .....                                      | 40        |
| Avec <i>const</i> .....                                            | 41        |
| Avec <i>std::string</i> .....                                      | 41        |
| Quel intérêt ? .....                                               | 42        |
| 4.4. Les entrées .....                                             | 42        |
| 4.5. Récapitulons ! .....                                          | 44        |
| <b>5. Le conditionnel conjugué en C++ .....</b>                    | <b>45</b> |
| 5.1. Les booléens .....                                            | 45        |
| 5.2. <i>if</i> – si, et seulement si... .....                      | 47        |
| 5.3. <i>else</i> – sinon... .....                                  | 49        |
| 5.4. <i>else if</i> – la combinaison des deux précédents .....     | 51        |
| 5.5. Conditions imbriquées .....                                   | 52        |
| 5.6. [Pratique] Gérer les erreurs d'entrée • Partie I .....        | 53        |
| 5.7. La logique booléenne .....                                    | 54        |
| AND – tester si deux conditions sont vraies .....                  | 54        |
| OR – tester si au moins une condition est vraie .....              | 55        |
| NOT – tester la négation .....                                     | 56        |
| 5.8. Tester plusieurs expressions .....                            | 57        |
| 5.9. Entraînez-vous ! .....                                        | 60        |
| Une pseudo-horloge .....                                           | 60        |
| Score .....                                                        | 60        |
| XOR – le OU exclusif .....                                         | 61        |
| 5.10. Récapitulons ! .....                                         | 61        |
| <b>6. Des boucles qui se répètent, répètent, répètent... .....</b> | <b>63</b> |
| 6.1. <i>while</i> – tant que... .....                              | 63        |
| 6.2. Entraînez-vous ! .....                                        | 65        |
| Une laverie .....                                                  | 65        |
| PGCD .....                                                         | 66        |
| Somme de nombres de 1 à <i>n</i> .....                             | 66        |
| 6.3. <i>do while</i> – répéter ... tant que .....                  | 66        |
| 6.4. [Pratique] Gérer les erreurs d'entrée • Partie II .....       | 67        |
| 6.5. <i>for</i> – une boucle condensée .....                       | 68        |
| 6.6. Boucles imbriquées .....                                      | 69        |

|                                                                |           |
|----------------------------------------------------------------|-----------|
| 6.7. Convention de nommage .....                               | 70        |
| 6.8. Contrôler plus finement l'exécution de la boucle .....    | 70        |
| <i>break</i> – je m'arrête là .....                            | 70        |
| <i>continue</i> – saute ton tour ! .....                       | 72        |
| 6.9. Récapitulons ! .....                                      | 72        |
| <b>7. Au tableau ! .....</b>                                   | <b>73</b> |
| 7.1. Un tableau c'est quoi ? .....                             | 73        |
| 7.2. <i>std::vector</i> – mais quel dynamisme ! .....          | 73        |
| Déclarer un <i>std::vector</i> .....                           | 74        |
| Manipuler les éléments d'un <i>std::vector</i> .....           | 75        |
| 7.3. Entraînez-vous ! .....                                    | 83        |
| Calcul de moyenne .....                                        | 83        |
| Minimum et maximum .....                                       | 83        |
| Séparer les pairs des impairs .....                            | 83        |
| Compter les occurrences d'une valeur .....                     | 83        |
| 7.4. <i>std::array</i> – le tableau statique .....             | 84        |
| Déclarer un <i>std::array</i> .....                            | 84        |
| Remplir un tableau .....                                       | 85        |
| Accéder aux éléments .....                                     | 85        |
| Connaître la taille .....                                      | 86        |
| Vérifier si le tableau est vide .....                          | 87        |
| 7.5. <i>std::string</i> – un type qui cache bien son jeu ..... | 87        |
| Connaître la taille d'une chaîne .....                         | 87        |
| Accéder à un caractère .....                                   | 88        |
| Premier et dernier caractère .....                             | 88        |
| Vérifier qu'une chaîne est vide .....                          | 89        |
| Ajouter ou supprimer un caractère à la fin .....               | 89        |
| Supprimer tous les caractères .....                            | 90        |
| Boucler sur une chaîne .....                                   | 90        |
| 7.6. Récapitulons ! .....                                      | 91        |
| <b>8. Déployons la toute puissance des conteneurs.....</b>     | <b>93</b> |
| 8.1. La pièce maîtresse : les itérateurs .....                 | 93        |
| Déclaration d'un itérateur .....                               | 93        |
| Début et fin d'un conteneur .....                              | 94        |
| Accéder à l'élément pointé .....                               | 95        |
| Se déplacer dans une collection .....                          | 95        |
| <i>const</i> et les itérateurs .....                           | 98        |
| Itérer depuis la fin .....                                     | 99        |

|                                                                              |            |
|------------------------------------------------------------------------------|------------|
| Utilisation conjointe avec les conteneurs .....                              | 100        |
| 8.2. Des algorithmes à gogo ! .....                                          | 101        |
| <i>std::count</i> – compter les occurrences d’une valeur .....               | 102        |
| <i>std::find</i> – trouver un certain élément .....                          | 103        |
| <i>std::sort</i> – trier une collection .....                                | 104        |
| <i>std::reverse</i> – inverser l’ordre des éléments d’une collection ....    | 105        |
| <i>std::remove</i> – suppression d’éléments .....                            | 106        |
| <i>std::search</i> – rechercher un sous-ensemble dans un en-<br>semble ..... | 106        |
| <i>std::equal</i> – vérifier l’égalité de deux ensembles .....               | 107        |
| Et tant d’autres .....                                                       | 108        |
| 8.3. Personnalisation à la demande .....                                     | 108        |
| Le prédicat, pivot de la personnalisation des algorithmes .....              | 108        |
| Trier une liste dans l’ordre décroissant .....                               | 109        |
| Somme et produit .....                                                       | 110        |
| Prédicats pour caractères .....                                              | 111        |
| 8.4. Entraînez-vous ! .....                                                  | 112        |
| Palindrome .....                                                             | 112        |
| <i>string_trim</i> – suppression des espaces .....                           | 112        |
| Couper une chaîne .....                                                      | 113        |
| 8.5. Récapitulons ! .....                                                    | 113        |
| <b>9. Des flux dans tous les sens .....</b>                                  | <b>115</b> |
| 9.1. Avant toute chose .....                                                 | 115        |
| Prendrez-vous une extension ? .....                                          | 115        |
| Vois sur ton chemin... .....                                                 | 116        |
| Pas d’interprétation, on est des brutes .....                                | 117        |
| 9.2. <i>std::ofstream</i> – écrire dans un fichier .....                     | 118        |
| Ouvrir le fichier .....                                                      | 118        |
| Écrire .....                                                                 | 118        |
| Ouvrir sans effacer .....                                                    | 121        |
| 9.3. <i>std::ifstream</i> – lire dans un fichier .....                       | 122        |
| Ouvrir le fichier .....                                                      | 122        |
| Lire .....                                                                   | 122        |
| Tout lire .....                                                              | 125        |
| 9.4. Entraînez-vous ! .....                                                  | 125        |
| Statistiques sur des fichiers .....                                          | 125        |
| 9.5. Encore plus de flux ! .....                                             | 126        |
| Un flux c’est quoi ? .....                                                   | 126        |
| Un <i>buffer</i> dites-vous ? .....                                          | 127        |

|                                                                |            |
|----------------------------------------------------------------|------------|
| Les modificateurs de flux .....                                | 128        |
| Même les chaînes y passent ! .....                             | 129        |
| 9.6. Récapitulons ! .....                                      | 130        |
| <b>On passe la deuxième ! .....</b>                            | <b>131</b> |
| <b>10. Découpons le code — les fonctions.....</b>              | <b>133</b> |
| 10.1. Les éléments de base d'une fonction .....                | 133        |
| Une fonction, c'est quoi ? .....                               | 133        |
| Une fonction incontournable .....                              | 135        |
| Les composants d'une fonction .....                            | 136        |
| 10.2. Entraînez-vous ! .....                                   | 139        |
| Afficher un rectangle .....                                    | 139        |
| Distributeur d'argent .....                                    | 139        |
| Parenthésage .....                                             | 140        |
| 10.3. Quelles sont vos références ? .....                      | 140        |
| La problématique .....                                         | 140        |
| Les références .....                                           | 142        |
| Paramètres de fonctions .....                                  | 143        |
| Valeur de retour .....                                         | 146        |
| Un mot sur la déduction de type .....                          | 147        |
| 10.4. Nos fonctions sont surchargées ! .....                   | 149        |
| 10.5. [Pratique] Gérer les erreurs d'entrée • Partie III ..... | 150        |
| 10.6. Dessine-moi une fonction .....                           | 151        |
| Le problème .....                                              | 151        |
| La solution .....                                              | 152        |
| Utilisons les prototypes .....                                 | 153        |
| 10.7. Récapitulons ! .....                                     | 154        |
| <b>11. Erreur, erreur, erreur... .....</b>                     | <b>155</b> |
| 11.1. L'utilisateur est un idiot .....                         | 155        |
| 11.2. À une condition... ou plusieurs .....                    | 156        |
| Contrats assurés par le compilateur .....                      | 157        |
| Vérifions nous aussi .....                                     | 157        |
| 11.3. Le développeur est un idiot .....                        | 157        |
| Préconditions et assertions .....                              | 160        |
| 11.4. Les tests unitaires à la rescousse .....                 | 161        |
| Pourquoi tester ? .....                                        | 161        |
| Un test unitaire, qu'est-ce que c'est ? .....                  | 161        |
| Écrire des tests unitaires .....                               | 162        |

|                                                                 |            |
|-----------------------------------------------------------------|------------|
| Les tests unitaires et les postconditions .....                 | 163        |
| 11.5. [Pratique] Gérer les erreurs d'entrée • Partie IV .....   | 163        |
| 11.6. L'exception à la règle .....                              | 165        |
| La problématique .....                                          | 165        |
| C'est quoi une exception ? .....                                | 166        |
| Lancement des exceptions dans 3, 2, 1... .....                  | 167        |
| Attends que je t'attrape ! .....                                | 168        |
| Attrapez-les tous ! .....                                       | 170        |
| Un type pour les gouverner tous et dans le catch les lier ..... | 171        |
| 11.7. [Pratique] Gérer les erreurs d'entrée • Partie V .....    | 173        |
| 11.8. Récapitulons ! .....                                      | 173        |
| <b>12. Des fonctions somme toute lambdas.....</b>               | <b>175</b> |
| 12.1. Une lambda, c'est quoi ? .....                            | 175        |
| 12.2. Pourquoi a-t-on besoin des lambdas ? .....                | 175        |
| 12.3. Syntaxe .....                                             | 176        |
| Quelques exemples simples .....                                 | 176        |
| 12.4. Entraînez-vous ! .....                                    | 179        |
| Vérifier si un nombre est négatif .....                         | 179        |
| Tri par valeur absolue .....                                    | 179        |
| 12.5. On va stocker ça où ? .....                               | 179        |
| 12.6. Paramètres génériques .....                               | 180        |
| 12.7. [Pratique] Gérer les erreurs d'entrée • Partie VI .....   | 182        |
| 12.8. [Pratique] Gérer les erreurs d'entrée • Partie VII .....  | 182        |
| 12.9. Capture en cours... .....                                 | 183        |
| Capture par valeur .....                                        | 183        |
| Capture par référence .....                                     | 185        |
| 12.10. Récapitulons ! .....                                     | 186        |
| <b>13. Envoyez le générique ! .....</b>                         | <b>187</b> |
| 13.1. Quel beau modèle ! .....                                  | 187        |
| La bibliothèque standard : une fourmilière de modèles .....     | 192        |
| Un point de vocabulaire .....                                   | 192        |
| 13.2. [Pratique] Gérer les erreurs d'entrée • Partie VIII ..... | 193        |
| 13.3. Instanciation explicite .....                             | 194        |
| 13.4. Ce type-là, c'est un cas ! .....                          | 198        |
| 13.5. C++20 et les lambdas templates .....                      | 202        |
| 13.6. Récapitulons ! .....                                      | 203        |
| <b>14. De nouvelles structures de données.....</b>              | <b>205</b> |
| 14.1. <i>struct</i> – un agrégat de données .....               | 205        |

|                                                                 |            |
|-----------------------------------------------------------------|------------|
| Stockage d'informations personnelles .....                      | 205        |
| Analyse de la méthode .....                                     | 206        |
| Introduction aux structures .....                               | 206        |
| Et maintenant, structurons ! .....                              | 208        |
| 14.2. <i>std::tuple</i> – une collection hétérogène .....       | 209        |
| Déclaration .....                                               | 209        |
| Accès aux éléments .....                                        | 209        |
| <i>std::tuple</i> et fonctions .....                            | 211        |
| Équations horaires .....                                        | 212        |
| 14.3. <i>std::unordered_map</i> – une table associative .....   | 214        |
| Un dictionnaire de langue Zestienne .....                       | 215        |
| Toujours plus de clés .....                                     | 217        |
| Cette clé est unique ! .....                                    | 218        |
| Cherchez un élément .....                                       | 220        |
| Un exemple concret .....                                        | 222        |
| 14.4. <i>std::unordered_set</i> – représenter un ensemble ..... | 223        |
| Inscription sur Zeste de Savoir .....                           | 223        |
| 14.5. Un peu d'ordre, voyons ! .....                            | 224        |
| 14.6. Une longue énumération .....                              | 226        |
| 14.7. Récapitulons ! .....                                      | 228        |
| <b>15. Reprendrez-vous un peu de sucre syntaxique ? .....</b>   | <b>229</b> |
| 15.1. Ce type est trop long ! .....                             | 229        |
| 15.2. Décomposons tout ça .....                                 | 230        |
| En pleine décomposition .....                                   | 230        |
| Et avant, on faisait comment ? .....                            | 232        |
| 15.3. La surcharge d'opérateurs .....                           | 234        |
| Le problème... ..                                               | 234        |
| ...et la solution .....                                         | 236        |
| Mise en pratique .....                                          | 236        |
| Et la bibliothèque standard ? .....                             | 243        |
| Entraînez-vous ! .....                                          | 244        |
| Sur le bon usage de la surcharge .....                          | 245        |
| 15.4. Récapitulons ! .....                                      | 245        |
| <b>16. [Pratique] Un gestionnaire de discographie .....</b>     | <b>247</b> |
| 16.1. Cahier des charges .....                                  | 247        |
| Ajout de morceaux .....                                         | 247        |
| Affichage de la discographie .....                              | 248        |
| Enregistrement et chargement d'une discographie .....           | 250        |

|                                                          |            |
|----------------------------------------------------------|------------|
| Fermeture du programme .....                             | 250        |
| Gestion des erreurs .....                                | 250        |
| Dernières remarques .....                                | 250        |
| 16.2. Guide de résolution .....                          | 251        |
| Analyse des besoins .....                                | 251        |
| Attaquons le code ! .....                                | 252        |
| 16.3. Code complet .....                                 | 256        |
| 16.4. Conclusion et pistes d'améliorations .....         | 266        |
| 16.5. Récapitulons ! .....                               | 267        |
| <b>17. Découpons du code — les fichiers .....</b>        | <b>269</b> |
| 17.1. Le principe .....                                  | 269        |
| Le fichier d'en-tête .....                               | 269        |
| Le fichier source .....                                  | 272        |
| Créons un lien .....                                     | 273        |
| 17.2. La sécurité, c'est important .....                 | 274        |
| 17.3. [Pratique] Découpons notre programme ! .....       | 275        |
| 17.4. Les avantages du découpage en fichiers .....       | 277        |
| 17.5. Le cas des templates .....                         | 279        |
| 17.6. Récapitulons ! .....                               | 282        |
| <b>Interlude : être développeur .....</b>                | <b>283</b> |
| <b>18. Un coup d'œil dans le rétro .....</b>             | <b>285</b> |
| 18.1. Au-delà du code .....                              | 285        |
| 18.2. Tout est une question de paradigme .....           | 286        |
| Le paradigme impératif .....                             | 286        |
| Le paradigme fonctionnel .....                           | 287        |
| Le paradigme générique .....                             | 288        |
| La programmation par contrat .....                       | 289        |
| 18.3. L'art de la conception .....                       | 289        |
| 18.4. De la persévérance avant tout .....                | 290        |
| 18.5. Récapitulons ! .....                               | 291        |
| <b>19. Mais où est la doc ? .....</b>                    | <b>293</b> |
| 19.1. Lire une page de doc .....                         | 293        |
| vector — retour sur le plus célèbre des conteneurs ..... | 294        |
| Les algorithmes .....                                    | 297        |
| Les chaînes de caractères au grand complet .....         | 298        |
| 19.2. Entraînez-vous .....                               | 299        |

|                                                        |            |
|--------------------------------------------------------|------------|
| Remplacer une chaîne de caractères par une autre ..... | 299        |
| Norme d'un vecteur .....                               | 299        |
| Nombres complexes .....                                | 300        |
| Transformations .....                                  | 300        |
| 19.3. Documenter son code avec Doxygen .....           | 301        |
| Installation d'un outil dédié .....                    | 301        |
| Écrire la documentation .....                          | 304        |
| 19.4. Quelques bonnes pratiques .....                  | 310        |
| Ce qu'il faut documenter .....                         | 310        |
| Les fichiers d'en-tête ou sources ? .....              | 311        |
| Rien ne vaut un exemple .....                          | 311        |
| Commentaire vs documentation .....                     | 312        |
| 19.5. Récapitulons ! .....                             | 314        |
| <b>20. Compilation en cours... .....</b>               | <b>315</b> |
| 20.1. Le préprocesseur .....                           | 315        |
| Inclure des fichiers .....                             | 315        |
| Conditions .....                                       | 316        |
| Debug ou release ? .....                               | 316        |
| 20.2. La compilation .....                             | 318        |
| Les templates .....                                    | 318        |
| constexpr .....                                        | 319        |
| La compilation à proprement parler .....               | 322        |
| Une étape intermédiaire cachée .....                   | 323        |
| Influer sur la compilation .....                       | 323        |
| 20.3. Le linker .....                                  | 328        |
| Une question de symboles .....                         | 329        |
| 20.4. Schéma récapitulatif .....                       | 331        |
| 20.5. Récapitulons ! .....                             | 332        |
| <b>21. Chasse aux bugs ! .....</b>                     | <b>333</b> |
| 21.1. Le principe .....                                | 333        |
| 21.2. Un code d'exemple .....                          | 334        |
| 21.3. Avec Visual Studio .....                         | 334        |
| Interface .....                                        | 335        |
| Pas-à-pas .....                                        | 337        |
| Point d'arrêt conditionnel .....                       | 338        |
| 21.4. Avec Qt Creator .....                            | 339        |
| Interface .....                                        | 339        |
| Pas-à-pas .....                                        | 341        |

|                                                      |            |
|------------------------------------------------------|------------|
| Point d'arrêt conditionnel .....                     | 342        |
| 21.5. En ligne de commande avec gdb .....            | 343        |
| Poser un point d'arrêt .....                         | 344        |
| Supprimer des points d'arrêt .....                   | 344        |
| Désactiver des points d'arrêt .....                  | 345        |
| Afficher l'état d'une variable .....                 | 345        |
| Pas-à-pas .....                                      | 345        |
| Conditions .....                                     | 346        |
| 21.6. Récapitulons ! .....                           | 347        |
| <b>22. Une foule de bibliothèques .....</b>          | <b>349</b> |
| 22.1. Quelles bibliothèques choisir ? .....          | 349        |
| 22.2. Généralités .....                              | 350        |
| Statique ou dynamique ? .....                        | 350        |
| Debug ou release ? .....                             | 350        |
| 32 ou 64 bits ? .....                                | 351        |
| 22.3. Installer Boost .....                          | 351        |
| GNU/Linux .....                                      | 352        |
| Visual Studio 2019 .....                             | 353        |
| Qt Creator et MinGW sous Windows .....               | 355        |
| 22.4. Installer SFML .....                           | 357        |
| GNU/Linux .....                                      | 358        |
| Windows .....                                        | 359        |
| 22.5. Récapitulons ! .....                           | 364        |
| <b>23. Améliorer ses projets .....</b>               | <b>365</b> |
| 23.1. git – sauvegarder et versionner son code ..... | 365        |
| Installer git .....                                  | 366        |
| Initialiser git .....                                | 366        |
| Créer un dépôt .....                                 | 366        |
| Connaître l'état de ses fichiers .....               | 367        |
| Ajouter un fichier .....                             | 368        |
| Valider les modifications .....                      | 368        |
| Voir l'historique .....                              | 369        |
| Bloquer certains fichiers .....                      | 369        |
| Aller plus loin .....                                | 371        |
| 23.2. GitLab – partager son code .....               | 371        |
| Création d'un projet .....                           | 372        |
| Pousser nos modifications .....                      | 373        |
| Aller plus loin .....                                | 374        |

|                                                                  |     |
|------------------------------------------------------------------|-----|
| 23.3. CMake – automatiser la compilation de ses programmes ..... | 374 |
| Un exemple simple .....                                          | 374 |
| Lier des bibliothèques externes .....                            | 378 |
| Définir des variables dynamiquement .....                        | 385 |
| Aller plus loin .....                                            | 385 |
| 23.4. Encore quelques outils .....                               | 386 |
| GitLab CI – de l'intégration continue .....                      | 386 |
| CppCheck – vérification du code .....                            | 386 |
| StackOverflow – la réponse à quasi toutes les questions .....    | 386 |
| 23.5. Récapitulons ! .....                                       | 387 |

## La programmation orientée objet ..... 389

### 24. Premiers pas avec la POO ..... 391

|                                                      |     |
|------------------------------------------------------|-----|
| 24.1. Le principe : des objets bien serviables ..... | 391 |
| Penser en termes de données... ..                    | 391 |
| ...ou de services ? .....                            | 392 |
| Exemple concret .....                                | 392 |
| 24.2. Un peu de vocabulaire .....                    | 393 |
| L'objet .....                                        | 393 |
| Le type .....                                        | 393 |
| L'interface d'une classe .....                       | 393 |
| 24.3. En C++, ça donne quoi ? .....                  | 394 |
| Penser services .....                                | 394 |
| Penser tests .....                                   | 394 |
| Fonctions membres .....                              | 395 |
| Fonctions libres .....                               | 396 |
| Les attributs .....                                  | 397 |
| 24.4. Désolé, cet objet n'est pas modifiable .....   | 398 |
| 24.5. On ne fait pas d'exception .....               | 401 |
| 24.6. Découpage en fichiers .....                    | 402 |
| 24.7. Entraînez-vous ! .....                         | 404 |
| Cercle .....                                         | 404 |
| Informations personnelles .....                      | 405 |
| 24.8. Récapitulons ! .....                           | 405 |

### 25. Et qui va construire tout ça ? ..... 407

|                                         |     |
|-----------------------------------------|-----|
| 25.1. Encapsulation et invariants ..... | 407 |
| Cette classe, c'est open bar .....      | 407 |
| Tout est question d'invariant .....     | 408 |

|                                                     |            |
|-----------------------------------------------------|------------|
| Encapsulons tout ça .....                           | 409        |
| Modifier sa visibilité .....                        | 409        |
| 25.2. On cherche un constructeur .....              | 411        |
| Le principe .....                                   | 411        |
| La syntaxe .....                                    | 413        |
| 25.3. Constructeur par défaut .....                 | 414        |
| 25.4. Soyons explicites .....                       | 417        |
| 25.5. En toute amitié .....                         | 421        |
| Attributs publics .....                             | 423        |
| Accesseurs .....                                    | 423        |
| La troisième, c'est la bonne .....                  | 425        |
| 25.6. Entraînez-vous ! .....                        | 426        |
| Un autre constructeur pour <i>Fraction</i> .....    | 426        |
| Une pile .....                                      | 426        |
| 25.7. Récapitulons ! .....                          | 428        |
| <b>26. Une classe de grande valeur .....</b>        | <b>429</b> |
| 26.1. Une histoire de sémantique .....              | 429        |
| 26.2. La sémantique de valeur, c'est quoi ? .....   | 431        |
| 26.3. Égalité .....                                 | 431        |
| Libre ou membre ? .....                             | 432        |
| Soyons intelligents, réutilisons .....              | 436        |
| 26.4. Le retour des opérateurs .....                | 437        |
| Les opérateurs d'affectation intégrés .....         | 437        |
| Du deux en un .....                                 | 438        |
| Les opérateurs de manipulation des flux .....       | 439        |
| Quelques bonnes pratiques .....                     | 443        |
| Les autres opérateurs .....                         | 443        |
| 26.5. Copier des objets .....                       | 444        |
| Constructeur de copie .....                         | 444        |
| Opérateur d'affectation par copie .....             | 445        |
| 26.6. Récapitulons ! .....                          | 445        |
| <b>27. [Pratique] Entrons dans la matrice .....</b> | <b>447</b> |
| 27.1. Qu'est-ce qu'une matrice ? .....              | 447        |
| 27.2. Cahier des charges .....                      | 449        |
| Les services .....                                  | 449        |
| Les détails d'implémentation .....                  | 449        |
| Les tests .....                                     | 450        |
| 27.3. Réalisation .....                             | 454        |

|                                                     |            |
|-----------------------------------------------------|------------|
| Les services .....                                  | 454        |
| Les détails d'implémentation .....                  | 456        |
| 27.4. Code complet .....                            | 461        |
| 27.5. Entraînez-vous ! .....                        | 468        |
| Entiers infinis .....                               | 468        |
| Des polynômes .....                                 | 469        |
| 27.6. Récapitulons ! .....                          | 470        |
| <b>28. Classes à sémantique d'entité .....</b>      | <b>471</b> |
| 28.1. Des objets uniques ! .....                    | 471        |
| Conséquences sur l'interface .....                  | 471        |
| Un exemple .....                                    | 472        |
| 28.2. Des sous-types... .....                       | 474        |
| Exemple .....                                       | 474        |
| Héritage et constructeurs .....                     | 475        |
| Une question de visibilité .....                    | 477        |
| 28.3. ...substituables .....                        | 479        |
| Une histoire de manchots .....                      | 481        |
| Le principe de substitution de Liskov .....         | 482        |
| Substitution dans la bibliothèque standard .....    | 483        |
| Substitution et programmation par contrat .....     | 483        |
| Héritage public et LSP .....                        | 485        |
| 28.4. Un monde virtuel .....                        | 487        |
| Un problème... .....                                | 487        |
| ...une solution .....                               | 490        |
| En interne .....                                    | 494        |
| Faisons le point .....                              | 496        |
| 28.5. Interfaces et classes abstraites .....        | 497        |
| Des fonctions virtuelles vraiment pures .....       | 497        |
| Classe abstraite et implémentation .....            | 498        |
| 28.6. La copie, une vraie source de problème .....  | 499        |
| 28.7. Récapitulons ! .....                          | 501        |
| <b>29. Ressources, indirections et handles.....</b> | <b>503</b> |
| 29.1. Retour sur la mémoire .....                   | 503        |
| Du stockage pour tous ! .....                       | 503        |
| C'est à quelle adresse ? .....                      | 506        |
| 29.2. Les indirections .....                        | 506        |
| 29.3. Les ressources et les handles .....           | 508        |
| 29.4. Restitution garantie 100% satisfaction .....  | 509        |

|                                                                              |            |
|------------------------------------------------------------------------------|------------|
| Le travail manuel ne paie pas toujours...                                    | 509        |
| Resource Acquisition Is Initialization .....                                 | 511        |
| L'exemple corrigé .....                                                      | 513        |
| 29.5. <code>std::unique_ptr</code> – pour surveiller la mémoire .....        | 515        |
| Mais quel rôle ce pointeur joue-t-il ? .....                                 | 515        |
| Un seul propriétaire...                                                      | 515        |
| ...et plusieurs observateurs .....                                           | 517        |
| Liens entre propriétaire et observateurs .....                               | 518        |
| Transfert de responsabilité .....                                            | 519        |
| Plusieurs propriétaires, plusieurs observateurs .....                        | 520        |
| 29.6. Les entités et les handles .....                                       | 521        |
| 29.7. Un problème de destruction .....                                       | 524        |
| 29.8. Les pointeurs nus, on oublie ! .....                                   | 526        |
| <code>std::function</code> – un remplaçant pour manipuler des fonctions .... | 526        |
| <code>std::optional</code> – un remplaçant pour valeur optionnelle .....     | 527        |
| 29.9. De l'importance de prendre ses responsabilités .....                   | 528        |
| 29.10. Récapitulons ! .....                                                  | 529        |
| <b>30. La sémantique de déplacement .....</b>                                | <b>531</b> |
| 30.1. T'as saisi la référence ? .....                                        | 531        |
| Le besoin...                                                                 | 531        |
| ...la cause...                                                               | 533        |
| ...et la solution .....                                                      | 534        |
| <code>std::move</code> , la mal-nommée .....                                 | 535        |
| 30.2. ♪ I like to move it, move it ♪ .....                                   | 536        |
| 30.3. En interne .....                                                       | 539        |
| 30.4. Comme les cinq doigts de la main .....                                 | 542        |
| 30.5. Récapitulons ! .....                                                   | 542        |
| <b>31. Oh, le bel héritage .....</b>                                         | <b>545</b> |
| 31.1. NVI – Un contrat est un contrat .....                                  | 545        |
| Des problèmes d'héritage...                                                  | 545        |
| Non Virtual Interface .....                                                  | 545        |
| Changements dans les contrats .....                                          | 549        |
| 31.2. <code>protected</code> – la portée intermédiaire .....                 | 550        |
| 31.3. Mettons un point final à cet héritage .....                            | 553        |
| 31.4. L'héritage multiple .....                                              | 555        |
| 31.5. L'héritage, solution miracle ? .....                                   | 558        |
| 31.6. Récapitulons ! .....                                                   | 559        |
| <b>32. Les classes templates .....</b>                                       | <b>561</b> |

|                                                          |            |
|----------------------------------------------------------|------------|
| 32.1. Les classes génériques .....                       | 561        |
| Définir une classe générique .....                       | 561        |
| Référencer une classe générique .....                    | 563        |
| Arguments types et non-types .....                       | 563        |
| Valeur par défaut .....                                  | 564        |
| 32.2. [Pratique] Un wrapper pour les pointeurs nus ..... | 565        |
| 32.3. Les traits .....                                   | 566        |
| Exemple simple .....                                     | 566        |
| Traits et programmation par contrat .....                | 567        |
| En interne .....                                         | 569        |
| 32.4. Ces modèles sont bien trop génériques... ..        | 570        |
| En voilà un bon concept ! .....                          | 571        |
| Créer ses propres concepts .....                         | 572        |
| 32.5. Un peu de politique .....                          | 575        |
| À la base, une classe très hospitalière .....            | 576        |
| Points de variation .....                                | 576        |
| Lier l'implémentation et les points de variation .....   | 578        |
| 32.6. Récapitulons ! .....                               | 582        |
| <b>33. Ça, c'est du SOLID ! .....</b>                    | <b>583</b> |
| 33.1. S – Single responsibility principle .....          | 583        |
| Le principe .....                                        | 583        |
| Un exemple problématique... ..                           | 584        |
| ...et sa solution .....                                  | 585        |
| 33.2. O – Open-closed principle .....                    | 585        |
| Le principe .....                                        | 585        |
| Un exemple problématique... ..                           | 586        |
| ...et sa solution .....                                  | 587        |
| 33.3. L – Liskov substitution principle .....            | 589        |
| Le principe .....                                        | 589        |
| Un exemple problématique... ..                           | 589        |
| ...et sa solution .....                                  | 590        |
| 33.4. I – Interface segregation principle .....          | 590        |
| Le principe .....                                        | 590        |
| Un exemple problématique... ..                           | 591        |
| ...et sa solution .....                                  | 592        |
| 33.5. D – Dependency inversion principle .....           | 594        |
| Le principe .....                                        | 594        |
| Un exemple problématique... ..                           | 595        |
| ...et sa solution .....                                  | 596        |

|                                                                      |            |
|----------------------------------------------------------------------|------------|
| 33.6. Pour quelques principes de plus .....                          | 597        |
| 33.7. Récapitulons ! .....                                           | 598        |
| <b>34. Le voyage ne fait que commencer .....</b>                     | <b>599</b> |
| 34.1. Davantage sur C++ .....                                        | 599        |
| 34.2. Davantage sur le développement .....                           | 600        |
| <b>[Pratique] Gérer les erreurs d'entrée • Solutions .....</b>       | <b>601</b> |
| <b>Solutions des exercices .....</b>                                 | <b>611</b> |
| 1. Chapitre Rencontre avec le C++ .....                              | 611        |
| 2. Chapitre Tout ça est bien variable .....                          | 611        |
| 3. Chapitre Le conditionnel conjugué en C++ .....                    | 613        |
| 4. Chapitre Des boucles qui se répètent, répètent, répètent... ..... | 616        |
| 5. Chapitre Au tableau ! .....                                       | 619        |
| 6. Chapitre Déployons la toute puissance des conteneurs.....         | 622        |
| 7. Chapitre Des flux dans tous les sens .....                        | 625        |
| 8. Chapitre Découpons le code – les fonctions.....                   | 626        |
| 9. Chapitre Erreur, erreur, erreur... .....                          | 629        |
| 10. Chapitre Des fonctions somme toute lambdas.....                  | 631        |
| 11. Chapitre De nouvelles structures de données.....                 | 632        |
| 12. Chapitre Reprendrez-vous un peu de sucre syntaxique ? .....      | 635        |
| 13. Chapitre [Pratique] Un gestionnaire de discographie .....        | 637        |
| section Représentation de la discographie .....                      | 637        |
| section Fonctions de base .....                                      | 638        |
| section Stockage dans un fichier .....                               | 642        |
| section Affichage de la discographie .....                           | 643        |
| section Système de commandes .....                                   | 646        |
| section Finalisation .....                                           | 648        |
| 14. Chapitre Découpons du code – les fichiers .....                  | 650        |
| 15. Chapitre Mais où est la doc ? .....                              | 670        |
| 16. Chapitre Premiers pas avec la POO .....                          | 673        |
| 17. Chapitre Et qui va construire tout ça ? .....                    | 675        |
| <br>Index .....                                                      | <br>677    |
| À propos des auteurs .....                                           | 687        |

# Bienvenue !

---

Bienvenue dans ce livre sur l'apprentissage de la programmation avec le langage C++. Vous débutez complètement et n'avez qu'une vague idée de ce que peut bien être la programmation ? On vous a dit d'en apprendre plus sur le C++ ? Cela tombe bien ! Notre objectif est de vous former au C++ moderne et de vous donner des bases solides sur lesquelles continuer.

Pour rendre ce voyage plus pédagogique, nous alternerons entre théorie, explications et détails techniques d'un côté et exercices, cas pratiques et questions-réponses de l'autre. L'accent sera mis sur la rigueur, la qualité et les bonnes pratiques. Le rythme sera progressif, chaque concept étant introduit en temps voulu et en lien avec ce que nous aurons déjà vu.

## URL RACCOURCIES

Dans un souci de lisibilité, et pour pouvoir les maintenir à jour, nous avons pris le parti de remplacer toutes les adresses internet par ce qu'on appelle des URL raccourcies. Une fois que vous avez accédé à la page cible, nous vous invitons à l'enregistrer avec un marque-page si vous souhaitez y revenir fréquemment. Vous disposerez alors du lien direct. Si celui-ci se périmé, n'hésitez pas à repasser par l'URL raccourcie. Si cette dernière aussi échoue, vous pouvez nous le signaler !

## 1. Codes sources des exemples

Tous les codes sources des exemples et des exercices proposés dans ce livre sont disponibles sur le dépôt GitLab `guide-du-cpp-moderne` [<http://d-booker.io.my/cpp-sces>] sous licence LGPL-3.0. Si, malgré tous nos soins, vous relevez des erreurs, pensez à nous les transmettre. Les autres lecteurs pourront ainsi en profiter.

## 2. Remerciements

Cet ouvrage n'aurait jamais vu le jour sans l'aide et le soutien de nombreuses personnes. Nous tenons en tout premier lieu à remercier notre éditrice, qui nous a donné de nombreux conseils, qui a su nous remotiver dans les moments de creux et qui a toujours été disponible pour nous aider.

Nous remercions également Luc Hermitte, pour ses nombreuses relectures, pour ses conseils, ses suggestions. En particulier, nous sommes reconnaissants de l'énorme travail

d'amélioration de la partie sur la programmation orientée objet, qui a été possible grâce à son aide et ses apports.

Il serait également injuste d'oublier Guillaume Belz, dont l'expérience dans l'enseignement du C++ et les nombreuses remarques ont façonné notre vision de la pédagogie et du C++.

Cet ouvrage doit aussi beaucoup à l'association Zeste de Savoir [<https://zestedesavoir.com>], une association à but non-lucratif centrée sur le partage gratuit, libre et de qualité du savoir. Ce projet est né sur leur site, beaucoup de chapitres y ont été relus, corrigés, améliorés grâce aux suggestions et aux efforts de différents membres, que nous ne pouvons individuellement citer tant ils sont nombreux.

À titre personnel, je (Benoît) souhaite remercier ma femme pour sa patience et sa compréhension par rapport aux longues heures que j'ai parfois passées à écrire. Ce livre est aussi possible grâce à toi !

Quant à moi (Mehdi), je souhaite remercier ma famille qui m'a toujours soutenu dans mes projets, et mes amis qui m'ont poussé à travailler ce livre quand la motivation n'était pas là. Je n'aurais pas pu y arriver sans vous !

# 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++. Ensuite, de nouvelles corrections et quelques ajustements ont été apportés pour donner C++14 et C++17. Enfin, dernièrement est sortie la norme C++20, nouvelle version majeure, qui apporte tout un lot de choses intéressantes ; et C++23 est déjà en chantier.

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* [<http://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.
- Sa *facilité d'apprentissage* : depuis sa version de 2011, le C++ est beaucoup plus facile à apprendre que par le passé. Et ça tombe bien, c'est sur cette version et les suivantes que se base ce livre.
- 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 programmation. 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++ : <http://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 2020 et explique les règles de fonctionnement du C++ dans sa version de 2020. 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 et 2020.
- 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.