Claude Delannoy

C++ pour les programmeurs C

EYROLLES

2-005-812-1

Claude Delannoy

C++ pour les programmeurs C

EYROLLES

Table des matières

Avant-propos XXIII

Chapitre 1 : Généralités concernant C++ 1

1 - La Programmation Orientée Objet 1

1.1 Problématique de la programmation 1

1.2 La programmation structurée 2

1.3 Les apports de la Programmation Orientée Objet 2

1.3.1 Objet 2

1.3.2 Encapsulation 3

1.3.3 Classe 3

1.3.4 Héritage 3

1.3.5 Polymorphisme 4

1.4 P.O.O. et langages 4

2 - C++, C ANSI et P.O.O. 5

3 - Les spécificités de C++ 5

4 - C++ et la programmation orientée objet 6

Chapitre 2 : Les incompatibilités entre C++ et C 9

1 - Les définitions de fonctions en C++ 10

2 - Les prototypes en C++ 10

3 - Arguments et valeur de retour d'une fonction 13

3.1 Points communs à C et C++ 13

3.2 Différences entre C et C++ . . . 13
3.2.1 Fonctions sans arguments . . . 14
3.2.2 Fonctions sans valeur de retour . . . 14
4 - Le qualificatif const . . . 14
4.1 Portée . . . 14
4.2 Utilisation dans une expression . . . 15
5 - Compatibilité entre le type void * et les autres pointeurs . . . 16

Chapitre 3 : Les entrées-sorties conversationnelles du C++ . . . 17

1 - Généralités . . . 17
2 - Affichage à l'écran . . . 18
2.1 Quelques exemples . . . 18
2.2 Le fichier en-tête iostream . . . 20
2.3 Les possibilités d'écriture sur cout . . . 20
3 - Lecture au clavier . . . 21
3.1 Introduction . . . 21
3.2 Les différentes possibilités de lecture sur cin . . . 22
3.3 Exemple classique d'utilisation des séparateurs . . . 23
3.4 Lecture d'une suite de caractères . . . 23
3.5 Les risques induits par la lecture au clavier . . . 24
3.5.1 Manque de synchronisme entre clavier et écran . . . 24
3.5.2 Blocage de la lecture . . . 25
3.5.3 Boucle infinie sur un caractère invalide . . . 25

Chapitre 4 : Les spécificités du C++ . . . 27

1 - Le commentaire de fin de ligne . . . 28
2 - Déclarations et initialisations . . . 28
2.1 Règles générales . . . 28
2.2 Cas des instructions structurées . . . 29
3 - La notion de référence . . . 30
3.1 Transmission des arguments en C . . . 30
3.2 Exemple de transmission d'argument par référence . . . 31
3.3 Propriétés de la transmission par référence d'un argument . . . 33
3.3.1 Induction de risques indirects . . . 33
3.3.2 Absence de conversion . . . 34
3.3.3 Cas d'un argument effectif constant . . . 34
3.3.4 Cas d'un argument muet constant . . . 35
3.4 Transmission par référence d'une valeur de retour . . . 35
3.4.1 Introduction . . . 35
3.4.2 On obtient une lvalue . . . 36
3.4.3 Conversion . . . 36
3.4.4 Valeur de retour et constance . . . 37

3.5 La référence d'une manière générale ... 37
3.5.1 La notion de référence est plus générale que celle d'argument ... 38
3.5.2 Initialisation de référence ... 38
4 - Les arguments par défaut ... 39
4.1 Exemples ... 39
4.2 Les propriétés des arguments par défaut ... 41
5 - Surdéfinition de fonctions ... 42
5.1 Mise en œuvre de la surdéfinition de fonctions ... 42
5.2 Exemples de choix d'une fonction surdéfinie ... 43
5.3 Règles de recherche d'une fonction surdéfinie ... 46
5.3.1 Cas des fonctions à un argument ... 46
5.3.2 Cas des fonctions à plusieurs arguments ... 47
5.3.3 Le mécanisme de la surdéfinition de fonctions ... 47
6 - Les opérateurs new et delete ... 49
6.1 Exemples d'utilisation de new ... 49
6.2 Syntaxe et rôle de new ... 50
6.3 Exemples d'utilisation de l'opérateur delete ... 50
6.4 Syntaxe et rôle de l'opérateur delete ... 51
6.5 L'opérateur new (nothrow) ... 51
6.6 Gestion des débordements de mémoire avec set_new_handler ... 52
7 - La spécification inline ... 53
7.1 Rappels concernant les macros et les fonctions ... 53
7.2 Utilisation de fonctions en ligne ... 54
8 - Les espaces de noms ... 56
9 - Le type bool ... 57
10 - Les nouveaux opérateurs de cast ... 57

Chapitre 5 : Classes et objets ... 61

1 - Les structures en C++ ... 62
1.1 Rappel : les structures en C ... 62
1.2 Déclaration d'une structure comportant des fonctions membres ... 63
1.3 Définition des fonctions membres ... 64
1.4 Utilisation d'une structure comportant des fonctions membres ... 65
1.5 Exemple récapitulatif ... 66
2 - Notion de classe ... 67
3 - Affectation d'objets ... 70
4 - Notions de constructeur et de destructeur ... 72
4.1 Introduction ... 72
4.2 Exemple de classe comportant un constructeur ... 73
4.3 Construction et destruction des objets ... 75
4.4 Rôles du constructeur et du destructeur ... 76
4.5 Quelques règles ... 79

Table des matières

Avant-propos XXIII

Chapitre 1 : Généralités concernant C++ 1

1 - La Programmation Orientée Objet 1

1.1 Problématique de la programmation. 1

1.2 La programmation structurée 2

1.3 Les apports de la Programmation Orientée Objet 2

1.3.1 Objet 2

1.3.2 Encapsulation 3

1.3.3 Classe 3

1.3.4 Héritage 3

1.3.5 Polymorphisme 4

1.4 P.O.O. et langages 4

2 - C++, C ANSI et P.O.O. 5

3 - Les spécificités de C++ 5

4 - C++ et la programmation orientée objet 6

Chapitre 2 : Les incompatibilités entre C++ et C 9

1 - Les définitions de fonctions en C++ 10

2 - Les prototypes en C++ 10

3 - Arguments et valeur de retour d'une fonction 13

3.1 Points communs à C et C++ 13

3.2 Différences entre C et C++ . . . 13
3.2.1 Fonctions sans arguments . . . 14
3.2.2 Fonctions sans valeur de retour . . . 14
4 - Le qualificatif const . . . 14
4.1 Portée . . . 14
4.2 Utilisation dans une expression . . . 15
5 - Compatibilité entre le type void * et les autres pointeurs . . . 16

Chapitre 3 : Les entrées-sorties conversationnelles du C++ . . . 17

1 - Généralités . . . 17
2 - Affichage à l'écran . . . 18
2.1 Quelques exemples . . . 18
2.2 Le fichier en-tête iostream . . . 20
2.3 Les possibilités d'écriture sur cout . . . 20
3 - Lecture au clavier . . . 21
3.1 Introduction . . . 21
3.2 Les différentes possibilités de lecture sur cin . . . 22
3.3 Exemple classique d'utilisation des séparateurs . . . 23
3.4 Lecture d'une suite de caractères . . . 23
3.5 Les risques induits par la lecture au clavier . . . 24
3.5.1 Manque de synchronisme entre clavier et écran . . . 24
3.5.2 Blocage de la lecture . . . 25
3.5.3 Boucle infinie sur un caractère invalide . . . 25

Chapitre 4 : Les spécificités du C++ . . . 27

1 - Le commentaire de fin de ligne . . . 28
2 - Déclarations et initialisations . . . 28
2.1 Règles générales . . . 28
2.2 Cas des instructions structurées . . . 29
3 - La notion de référence . . . 30
3.1 Transmission des arguments en C . . . 30
3.2 Exemple de transmission d'argument par référence . . . 31
3.3 Propriétés de la transmission par référence d'un argument . . . 33
3.3.1 Induction de risques indirects . . . 33
3.3.2 Absence de conversion . . . 34
3.3.3 Cas d'un argument effectif constant . . . 34
3.3.4 Cas d'un argument muet constant . . . 35
3.4 Transmission par référence d'une valeur de retour . . . 35
3.4.1 Introduction . . . 35
3.4.2 On obtient une lvalue . . . 36
3.4.3 Conversion . . . 36
3.4.4 Valeur de retour et constance . . . 37

3.5 La référence d'une manière générale ... 37
3.5.1 La notion de référence est plus générale que celle d'argument ... 38
3.5.2 Initialisation de référence ... 38
4 - Les arguments par défaut ... 39
4.1 Exemples ... 39
4.2 Les propriétés des arguments par défaut ... 41
5 - Surdéfinition de fonctions ... 42
5.1 Mise en œuvre de la surdéfinition de fonctions ... 42
5.2 Exemples de choix d'une fonction surdéfinie ... 43
5.3 Règles de recherche d'une fonction surdéfinie ... 46
5.3.1 Cas des fonctions à un argument ... 46
5.3.2 Cas des fonctions à plusieurs arguments ... 47
5.3.3 Le mécanisme de la surdéfinition de fonctions ... 47
6 - Les opérateurs new et delete ... 49
6.1 Exemples d'utilisation de new ... 49
6.2 Syntaxe et rôle de new ... 50
6.3 Exemples d'utilisation de l'opérateur delete ... 50
6.4 Syntaxe et rôle de l'opérateur delete ... 51
6.5 L'opérateur new (nothrow) ... 51
6.6 Gestion des débordements de mémoire avec set_new_handler ... 52
7 - La spécification inline ... 53
7.1 Rappels concernant les macros et les fonctions ... 53
7.2 Utilisation de fonctions en ligne ... 54
8 - Les espaces de noms ... 56
9 - Le type bool ... 57
10 - Les nouveaux opérateurs de cast ... 57

Chapitre 5 : Classes et objets ... 61

1 - Les structures en C++ ... 62
1.1 Rappel : les structures en C ... 62
1.2 Déclaration d'une structure comportant des fonctions membres ... 63
1.3 Définition des fonctions membres ... 64
1.4 Utilisation d'une structure comportant des fonctions membres ... 65
1.5 Exemple récapitulatif ... 66
2 - Notion de classe ... 67
3 - Affectation d'objets ... 70
4 - Notions de constructeur et de destructeur ... 72
4.1 Introduction ... 72
4.2 Exemple de classe comportant un constructeur ... 73
4.3 Construction et destruction des objets ... 75
4.4 Rôles du constructeur et du destructeur ... 76
4.5 Quelques règles ... 79

5 - Les membres données statiques 80
5.1 Le qualificatif static pour un membre donnée 80
5.2 Initialisation des membres données statiques 81
5.3 Exemple 82
6 - Exploitation d'une classe 83
6.1 La classe comme composant logiciel 83
6.2 Protection contre les inclusions multiples 85
6.3 Cas des membres données statiques 86
6.4 En cas de modification d'une classe 86
6.4.1 La déclaration des membres publics n'a pas changé 86
6.4.2 La déclaration des membres publics a changé 87
7 - Les classes en général 87
7.1 Les autres sortes de classes en C++ 87
7.2 Ce qu'on peut trouver dans la déclaration d'une classe 87
7.3 Déclaration d'une classe 88

Chapitre 6 : Les propriétés des fonctions membres 91
1 - Surdéfinition des fonctions membres 91
1.1 Exemple 92
1.2 Incidence du statut public ou privé d'une fonction membre 93
2 - Arguments par défaut 94
3 - Les fonctions membres en ligne 95
4 - Cas des objets transmis en argument d'une fonction membre 97
5 - Mode de transmission des objets en argument 99
5.1 Transmission de l'adresse d'un objet 99
5.2 Transmission par référence 101
5.3 Les problèmes posés par la transmission par valeur 102
6 - Lorsqu'une fonction renvoie un objet 102
7 - Autoréférence : le mot clé this 103
8 - Les fonctions membres statiques 104
9 - Les fonctions membres constantes 106
9.1 Rappels sur l'utilisation de const en C 106
9.2 Définition d'une fonction membre constante 107
9.3 Propriétés d'une fonction membre constante 107
10 - Les membres mutables 109

Chapitre 7 : Construction, destruction et initialisation des objets 111
1 - Les objets automatiques et statiques 112
1.1 Durée de vie 112
1.2 Appel des constructeurs et des destructeurs 113
1.3 Exemple 113

2 - Les objets dynamiques ... 115
2.1 Les structures dynamiques ... 115
2.2 Les objets dynamiques ... 116
2.2.1 Points communs avec les structures dynamiques ... 116
2.2.2 Les nouvelles possibilités des opérateurs new et delete ... 117
2.2.3 Exemple ... 117
3 - Le constructeur de recopie ... 118
3.1 Présentation ... 118
3.1.1 Il n'existe pas de constructeur approprié ... 119
3.1.2 Il existe un constructeur approprié ... 119
3.1.3 Lorsqu'on souhaite interdire la contruction par recopie ... 120
3.2 Exemple 1 : objet transmis par valeur ... 121
3.2.1 Emploi du constructeur de recopie par défaut ... 121
3.2.2 Définition d'un constructeur de recopie ... 123
3.3 Exemple 2 : objet en valeur de retour d'une fonction ... 125
4 - Initialisation d'un objet lors de sa déclaration ... 127
5 - Objets membres ... 129
5.1 Introduction ... 129
5.2 Mise en œuvre des constructeurs et des destructeurs ... 129
5.3 Le constructeur de recopie ... 132
6 - Initialisation de membres dans l'en-tête d'un constructeur ... 132
7 - Les tableaux d'objets ... 133
7.1 Notations ... 133
7.2 Constructeurs et initialiseurs ... 134
7.3 Cas des tableaux dynamiques d'objets ... 135
8 - Les objets temporaires ... 136

Chapitre 8 : Les fonctions amies ... 141

1 - Exemple de fonction indépendante amie d'une classe ... 142
2 - Les différentes situations d'amitié ... 144
2.1 Fonction membre d'une classe, amie d'une autre classe ... 145
2.2 Fonction amie de plusieurs classes ... 146
2.3 Toutes les fonctions d'une classe amies d'une autre classe ... 147
3 - Exemple ... 147
3.1 Fonction amie indépendante ... 148
3.2 Fonction amie, membre d'une classe ... 149
4 - Exploitation de classes disposant de fonctions amies ... 150

Chapitre 9 : La surdéfinition d'opérateurs ... 151

1 - Le mécanisme de la surdéfinition d'opérateurs ... 152
1.1 Surdéfinition d'opérateur avec une fonction amie ... 153
1.2 Surdéfinition d'opérateur avec une fonction membre ... 154
1.3 Opérateurs et transmission par référence ... 156

2 - La surdéfinition d'opérateurs en général . . . 157
2.1 Se limiter aux opérateurs existants . . . 157
2.2 Se placer dans un contexte de classe . . . 158
2.3 Eviter les hypothèses sur le rôle d'un opérateur . . . 159
2.4 Cas des opérateurs ++ et -- . . . 160
2.5 Les opérateurs = et & ont une signification prédéfinie . . . 161
2.6 Les conversions . . . 162
2.7 Choix entre fonction membre et fonction amie . . . 163
3 - Exemple de surdéfinition de l'opérateur = . . . 163
3.1 Rappels concernant le constructeur par recopie . . . 163
3.2 Cas de l'affectation . . . 164
3.3 Algorithme proposé . . . 165
3.4 Valeur de retour . . . 167
3.5 En définitive . . . 167
3.6 Exemple de programme complet . . . 167
3.7 Lorsqu'on souhaite interdire l'affectation . . . 169
4 - La forme canonique d'une classe . . . 170
5 - Exemple de surdéfinition de l'opérateur [ ] . . . 171
6 - Surdéfinition de l'opérateur () . . . 173
7 - Surdéfinition des opérateurs new et delete . . . 174
7.1 Surdéfinition de new et delete pour une classe donnée . . . 175
7.2 Exemple . . . 175
7.3 D'une manière générale . . . 177

Chapitre 10 : Les conversions de type définies par l'utilisateur . . . 181
1 - Les différentes sortes de conversions définies par l'utilisateur . . . 182
2 - L'opérateur de cast pour la conversion type classe –> type de base . . . 184
2.1 Définition de l'opérateur de cast . . . 184
2.2 Exemple d'utilisation . . . 184
2.3 Appel implicite de l'opérateur de cast lors d'un appel de fonction . . . 185
2.4 Appel implicite de l'opérateur de cast dans l'évaluation d'une expression . . . 187
2.5 Conversions en chaîne . . . 188
2.6 En cas d'ambiguïté . . . 190
3 - Le constructeur pour la conversion type de base -> type classe . . . 191
3.1 Exemple . . . 191
3.2 Le constructeur dans une chaîne de conversions . . . 193
3.3 Choix entre constructeur ou opérateur d'affectation . . . 193
3.4 Emploi d'un constructeur pour élargir la signification d'un opérateur . . . 195
3.5 Interdire les conversions implicites par le constructeur : le rôle d'explicit . . . 197
4 - Les conversions d'un type classe en un autre type classe . . . 198
4.1 Exemple simple d'opérateur de cast . . . 198
4.2 Exemple de conversion par un constructeur . . . 199
4.3 Pour donner une signification à un opérateur défini dans une autre classe . . . 200
5 - Quelques conseils . . . 203

Chapitre 11 : Les patrons de fonctions ... 205

1 - Exemple de création et d'utilisation d'un patron de fonctions ... 206
1.1 Création d'un patron de fonctions ... 206
1.2 Premières utilisations du patron de fonctions ... 207
1.3 Autres utilisations du patron de fonctions ... 208
*1.3.1 Application au type char ** ... 208
1.3.2 Application à un type classe ... 209
1.4 Contraintes d'utilisation d'un patron ... 210

2 - Les paramètres de type d'un patron de fonctions ... 211
2.1 Utilisation des paramètres de type dans la définition d'un patron ... 211
2.2 Identification des paramètres de type d'une fonction patron ... 212
2.3 Nouvelle syntaxe d'initialisation des variables des types standard ... 213
2.4 Limitations des patrons de fonctions ... 214

3 - Les paramètres expressions d'un patron de fonctions ... 215

4 - Surdéfinition de patrons ... 216
4.1 Exemples ne comportant que des paramètres de type ... 216
4.2 Exemples comportant des paramètres expressions ... 219

5 - Spécialisation de fonctions de patron ... 220
5.1 Généralités ... 220
5.2 Les spécialisations partielles ... 221

6 - Algorithme d'instanciation d'une fonction patron ... 222

Chapitre 12 : Les patrons de classes ... 225

1 - Exemple de création et d'utilisation d'un patron de classes ... 226
1.1 Création d'un patron de classes ... 226
1.2 Utilisation d'un patron de classes ... 228
1.3 Contraintes d'utilisation d'un patron de classes ... 228
1.4 Exemple récapitulatif ... 229

2 - Les paramètres de type d'un patron de classes ... 231
2.1 Les paramètres de type dans la création d'un patron de classes ... 231
2.2 Instanciation d'une classe patron ... 231

3 - Les paramètres expressions d'un patron de classes ... 233
3.1 Exemple ... 233
3.2 Les propriétés des paramètres expressions ... 235

4 - Spécialisation d'un patron de classes ... 235
4.1 Exemple de spécialisation d'une fonction membre ... 236
4.2 Les différentes possibilités de spécialisation ... 237
4.2.1 On peut spécialiser une fonction membre pour tous les paramètres ... 237
4.2.2 On peut spécialiser une fonction membre ou une classe ... 237
4.2.3 On peut prévoir des spécialisations partielles de patrons de classes ... 237

5 - Paramètres par défaut ... 238

6 - Patrons de fonctions membres ... 238

7 - Identité de classes patrons 239

8 - Classes patrons et déclarations d'amitié 239

8.1 Déclaration de classes ou fonctions "ordinaires" amies 239

8.2 Déclaration d'instances particulières de classes patrons ou de fonctions patrons 240

8.3 Déclaration d'un autre patron de fonctions ou de classes 241

9 - Exemple de classe tableau à deux indices 241

Chapitre 13 : L'héritage simple 245

1 - La notion d'héritage 246

2 - Utilisation des membres de la classe de base dans une classe dérivée 248

3 - Redéfinition des membres d'une classe dérivée 250

3.1 Redéfinition des fonctions membres d'une classe dérivée 250

3.2 Redéfinition des membres données d'une classe dérivée 252

3.3 Redéfinition et surdéfinition 252

4 - Appel des constructeurs et des destructeurs 254

4.1 Rappels 254

4.2 La hiérarchisation des appels 255

4.3 Transmission d'informations entre constructeurs 255

4.4 Exemple 256

4.5 Compléments 258

5 - Contrôle des accès 258

5.1 Les membres protégés 259

5.2 Exemple 259

5.3 Intérêt du statut protégé 260

5.4 Dérivation publique et dérivation privée 261

5.4.1 Rappels concernant la dérivation publique 261

5.4.2 Dérivation privée 262

5.5 Les possibilités de dérivation protégée 263

5.6 Récapitulation 264

6 - Compatibilité entre classe de base et classe dérivée 264

6.1 Conversion d'un type dérivé en un type de base 265

6.2 Conversion de pointeurs 266

6.3 Limitations liées au typage statique des objets 266

6.4 Les risques de violation des protections de la classe de base 269

7 - Le constructeur de recopie et l'héritage 270

7.1 La classe dérivée ne définit pas de constructeur de recopie 271

7.2 La classe dérivée définit un constructeur de recopie 271

8 - L'opérateur d'affectation et l'héritage 273

8.1 La classe dérivée ne surdéfinit pas l'opérateur = ... 273

8.2 La classe dérivée surdéfinit l'opérateur = ... 274

9 - Héritage et forme canonique d'une classe 277

10 - L'héritage et ses limites ... 278
10.1 La situation d'héritage ... 278
10.1.1 Le type du résultat de l'appel ... 279
10.1.2 Le type des arguments de f ... 279
10.2 Exemples ... 280
10.2.1 Héritage dans pointcol d'un opérateur + défini dans point ... 280
10.2.2 Héritage dans pointcol de la fonction coincide de point ... 280
11 - Exemple de classe dérivée ... 281
12 - Patrons de classes et héritage ... 284
12.1 Classe "ordinaire" dérivant d'une classe patron ... 285
12.2 Dérivation de patrons avec les mêmes paramètres ... 286
12.3 Dérivation de patrons avec introduction d'un nouveau paramètre ... 286
13 - L'héritage en pratique ... 287
13.1 Dérivations successives ... 287
13.2 Différentes utilisations de l'héritage ... 289
13.3 Exploitation d'une classe dérivée ... 289

Chapitre 14 : L'héritage multiple ... 291

1 - Mise en œuvre de l'héritage multiple ... 292
2 - Pour régler les éventuels conflits : les classes virtuelles ... 296
3 - Appels des constructeurs et des destructeurs : cas des classes virtuelles ... 298
4 - Exemple d'utilisation de l'héritage multiple et de la dérivation virtuelle ... 300

Chapitre 15 : Les fonctions virtuelles et le polymorphisme ... 305

1 - Rappel d'une situation où le typage dynamique est nécessaire ... 306
2 - Le mécanisme des fonctions virtuelles ... 306
3 - Autre situation où la ligature dynamique est indispensable ... 308
4 - Les propriétés des fonctions virtuelles ... 311
4.1 Leurs limitations sont celles de l'héritage ... 311
4.2 La redéfinition d'une fonction virtuelle n'est pas obligatoire ... 312
4.3 Fonctions virtuelles et surdéfinition ... 313
4.4 Le type de retour d'une fonction virtuelle redéfinie ... 313
4.5 On peut déclarer une fonction virtuelle dans n'importe quelle classe ... 314
4.6 Quelques restrictions et conseils ... 314
4.6.1 Seule une fonction membre peut être virtuelle ... 314
4.6.2 Un constructeur ne peut pas être virtuel ... 315
4.6.3 Un destructeur peut être virtuel ... 315
4.6.4 Cas particulier de l'opérateur d'affectation ... 316
5 - Les fonctions virtuelles pures pour la création de classes abstraites ... 317
6 - Exemple d'utilisation de fonctions virtuelles : liste hétérogène ... 319
7 - Le mécanisme d'identification dynamique des objets ... 324

8 - Identification de type à l'exécution 326
8.1 Utilisation du champ name de type_info 326
8.2 Utilisation des opérateurs de comparaison de type_info 328
8.3 Exemple avec des références 328
9 - Les cast dynamiques 329

Chapitre 16 : Les flots 333

1 - Présentation générale de la classe ostream 335
1.1 L'opérateur << 335
1.2 Les flots prédéfinis 336
1.3 La fonction put 337
1.4 La fonction write 337
1.4.1 Cas des caractères 337
1.4.2 Autres cas 337
1.5 Quelques possibilités de formatage avec << 338
1.5.1 Action sur la base de numération 338
1.5.2 Action sur le gabarit de l'information écrite 339
1.5.3 Action sur la précision de l'information écrite 341
1.5.4 Choix entre notation flottante ou exponentielle 341
2 - Présentation générale de la classe istream 342
2.1 L'opérateur >> 343
2.1.1 Cas des caractères 343
2.1.2 Cas des chaînes de caractères 344
2.1.3 Les types acceptés par >> 344
2.2 La fonction get 345
2.3 Les fonctions getline et gcount 345
2.4 La fonction read 347
2.4.1 Cas des caractères 347
2.4.2 Autres cas 347
2.5 Quelques autres fonctions 347
3 - Statut d'erreur d'un flot 348
3.1 Les bits d'erreur 348
3.2 Actions concernant les bits d'erreur 348
3.2.1 Accès aux bits d'erreur 349
3.2.2 Modification du statut d'erreur 349
3.3 Surdéfinition des opérateurs () et ! 349
3.4 Exemples 350
4 - Surdéfinition de << et >> pour les types définis par l'utilisateur 352
4.1 Méthode 352
4.2 Exemple 353
5 - Gestion du formatage 355
5.1 Le statut de formatage d'un flot 356
5.2 Description du mot d'état du statut de formatage 357

5.3 Action sur le statut de formatage ... 358
5.3.1 Les manipulateurs non paramétriques ... 358
5.3.2 Les manipulateurs paramétriques ... 359
5.3.3 Les fonctions membres ... 360
6 - Connexion d'un flot à un fichier ... 362
6.1 Connexion d'un flot de sortie à un fichier ... 362
6.2 Connexion d'un flot d'entrée à un fichier ... 364
6.3 Les possibilités d'accès direct ... 365
6.4 Les différents modes d'ouverture d'un fichier ... 367
7 - Les anciennes possibilités de formatage en mémoire ... 368
7.1 La classe ostrstream ... 369
7.2 La classe istrstream ... 370

Chapitre 17 : La gestion des exceptions ... 373

1 - Premier exemple d'exception ... 374
1.1 Comment lancer une exception : l'instruction throw ... 374
1.2 Utilisation d'un gestionnaire d'exception ... 375
1.3 Récapitulatif ... 376
2 - Second exemple ... 378
3 - Le mécanisme de gestion des exceptions ... 380
3.1 Poursuite de l'exécution du programme ... 380
3.2 Prise en compte des sorties de blocs ... 382
4 - Choix du gestionnaire ... 382
4.1 Le gestionnaire reçoit toujours une copie ... 383
4.2 Règles de choix d'un gestionnaire d'exception ... 383
4.3 Le cheminement des exceptions ... 384
4.4 Redéclenchement d'une exception ... 386
5 - Spécification d'interface : la fonction unexpected ... 387
6 - Les exceptions standard ... 390
6.1 Généralités ... 390
6.2 Les exceptions déclenchées par la bibliothèque standard ... 390
6.3 Les exceptions utilisables dans un programme ... 391
6.4 Création d'exceptions dérivées de la classe exception ... 391
6.4.1 Exemple 1 ... 392
6.4.2 Exemple 2 ... 393

Chapitre 18 : Généralités sur la bibliothèque standard ... 395

1 - Notions de conteneur, d'itérateur et d'algorithme ... 395
1.1 Notion de conteneur ... 396
1.2 Notion d'itérateur ... 396
1.3 Parcours d'un conteneur avec un itérateur ... 397
1.3.1 Parcours direct ... 397
1.3.2 Parcours inverse ... 398

1.4 Intervalle d'itérateur . . . 398
1.5 Notion d'algorithme . . . 399
1.6 Itérateurs et pointeurs . . . 400
2 - Les différentes sortes de conteneurs . . . 400
2.1 Conteneurs et structures de données classiques . . . 400
2.2 Les différentes catégories de conteneurs . . . 401
3 - Les conteneurs dont les éléments sont des objets . . . 401
3.1 Construction, copie et affectation . . . 402
3.2 Autres opérations . . . 403
4 - Efficacité des opérations sur des conteneurs . . . 403
5 - Fonctions, prédicats et classes fonctions . . . 404
5.1 Fonction unaire . . . 404
5.2 Prédicats . . . 405
5.3 Classes et objets fonctions . . . 405
5.3.1 Utilisation d'objet fonction comme fonction de rappel . . . 405
5.3.2 Classes fonction prédéfinies . . . 406
6 - Conteneurs, algorithmes et relation d'ordre . . . 407
6.1 Introduction . . . 407
6.2 Propriétés à respecter . . . 408
7 - Les générateurs d'opérateurs . . . 409

Chapitre 19 : Les conteneurs séquentiels . . . 411

1 - Fonctionnalités communes aux conteneurs vector, list et deque . . . 412
1.1 Construction . . . 412
1.1.1 Construction d'un conteneur vide . . . 412
1.1.2 Construction avec un nombre donné d'éléments . . . 412
1.1.3 Construction avec un nombre donné d'éléments initialisés à une valeur . . . 412
1.1.4 Construction à partir d'une séquence . . . 413
1.1.5 Construction à partir d'un autre conteneur de même type . . . 413
1.2 Modifications globales . . . 413
1.2.1 Opérateur d'affectation . . . 414
1.2.2 La fonction membre assign . . . 414
1.2.3 La fonction clear . . . 415
1.2.4 La fonction swap . . . 415
1.3 Comparaison de conteneurs . . . 415
1.3.1 L'opérateur == . . . 415
1.3.2 L'opérateur < . . . 416
1.3.3 Exemples . . . 416
1.4 Insertion ou suppression d'éléments . . . 416
1.4.1 Insertion . . . 417
1.4.2 Suppression . . . 417
1.4.3 Cas des insertions/suppressions en fin : pop_back et push_back . . . 418

2 - Le conteneur vector ... 418
2.1 Accès aux éléments existants ... 419
2.1.1 Accès par itérateur ... 419
2.1.2 Accès par indice ... 419
2.1.3 Cas de l'accès au dernier élément ... 420
2.2 Insertions et suppressions ... 420
2.3 Gestion de l'emplacement mémoire ... 420
2.3.1 Introduction ... 420
2.3.2 Invalidation d'itérateurs ou de références ... 421
2.3.3 Outils de gestion de l'emplacement mémoire d'un vecteur ... 421
2.4 Exemple ... 422
2.5 Cas particulier des vecteurs de booléens ... 423
3 - Le conteneur deque ... 424
3.1 Présentation générale ... 424
3.2 Exemple ... 425
4 - Le conteneur list ... 426
4.1 Accès aux éléments existants ... 426
4.2 Insertions et suppressions ... 427
4.2.1 Suppression des éléments de valeur donnée ... 427
4.2.2 Suppression des éléments répondant à une condition ... 427
4.3 Opérations globales ... 428
4.3.1 Tri d'une liste ... 428
4.3.2 Suppression des éléments en double ... 428
4.3.3 Fusion de deux listes ... 429
4.3.4 Transfert d'une partie de liste dans une autre ... 430
4.4 Gestion de l'emplacement mémoire ... 430
4.5 Exemple ... 431
5 - Les adaptateurs de conteneur : queue, stack et priority_queue ... 432
5.1 L'adaptateur stack ... 432
5.2 L'adaptateur queue ... 433
5.3 L'adaptateur priority_queue ... 434

Chapitre 20 : Les conteneurs associatifs ... 437

1 - Le conteneur map ... 438
1.1 Exemple introductif ... 438
1.2 Le patron de classes pair ... 440
1.3 Construction d'un conteneur de type map ... 440
1.3.1 Constructions utilisant la relation d'ordre par défaut ... 441
1.3.2 Choix de l'ordre intrinsèque du conteneur ... 441
1.3.3 Pour connaître la relation d'ordre utilisée par un conteneur ... 442
1.3.4 Conséquences du choix de l'ordre d'un conteneur ... 443
1.4 Accès aux éléments ... 443
1.4.1 Accès par l'opérateur [ ] ... 443
1.4.2 Accès par itérateur ... 443
1.4.3 Recherche par la fonction membre find ... 444

1.5 Insertions et suppressions ... 444
1.5.1 Insertions ... 445
1.5.2 Suppressions ... 446
1.6 Gestion mémoire ... 446
1.7 Autres possibilités ... 447
1.8 Exemple ... 447
2 - Le conteneur multimap ... 448
2.1 Présentation générale ... 448
2.2 Exemple ... 449
3 - Le conteneur set ... 451
3.1 Présentation générale ... 451
3.2 Exemple ... 451
3.3 Le conteneur set et l'ensemble mathématique ... 452
4 - Le conteneur multiset ... 452
5 - Conteneurs associatifs et algorithmes ... 453

Chapitre 21 : Les algorithmes standard ... 455

1 - Notions générales ... 455
1.1 Algorithmes et itérateurs ... 455
1.2 Les catégories d'itérateurs ... 456
1.2.1 Itérateur en entrée ... 456
1.2.2 Itérateur en sortie ... 456
1.2.3 Hiérarchie des catégories d'itérateurs ... 457
1.3 Algorithmes et séquences ... 457
1.4 Itérateur d'insertion ... 458
1.5 Itérateur de flot ... 460
1.5.1 Itérateur de flot de sortie ... 460
1.5.2 Itérateur de flot d'entrée ... 461
2 - Algorithmes d'initialisation de séquences existantes ... 462
2.1 Copie d'une séquence dans une autre ... 462
2.2 Génération de valeurs par une fonction ... 463
3 - Algorithmes de recherche ... 466
3.1 Algorithmes fondés sur une égalité ou un prédicat unaire ... 466
3.2 Algorithmes de recherche de maximum ou de minimum ... 467
4 - Algorithmes de transformation d'une séquence ... 468
4.1 Remplacement de valeurs ... 469
4.2 Permutations de valeurs ... 469
4.2.1 Rotation ... 469
4.2.2 Génération de permutations ... 470
4.2.3 Permutations aléatoires ... 471
4.3 Partitions ... 472
5 - Algorithmes dits "de suppression" ... 472
6 - Algorithmes de tris ... 474

7 - Algorithmes de recherche et de fusion sur des séquences ordonnées ... 476
7.1 Algorithmes de recherche binaire ... 476
7.2 Algorithmes de fusion ... 476
8 - Algorithmes à caractère numérique ... 478
9 - Algorithmes à caractère ensembliste ... 479
10 - Algorithmes de manipulation de tas ... 480

Chapitre 22 : La classe string ... 485

1 - Généralités ... 486
2 - Construction ... 486
3 - Opérations globales ... 487
4 - Concaténation ... 488
5 - Recherche dans une chaîne ... 488
5.1 Recherche d'une chaîne ou d'un caractère ... 489
5.2 Recherche d'un caractère présent ou absent d'une suite ... 489
6 - Insertions, suppressions et remplacements ... 490
6.1 Insertions ... 490
6.2 Suppressions ... 491
6.3 Remplacements ... 492
7 - Les possibilités de formatage en mémoire ... 493
7.1 La classe ostringstream ... 494
7.2 La classe istringstream ... 495
7.2.1 Présentation ... 495
7.2.2 Utilisation pour fiabiliser les lectures au clavier ... 495

Chapitre 23 : Les outils numériques ... 499

1 - La classe complex ... 499
2 - La classe valarray et les classes associées ... 501
2.1 Constructeurs des classes valarray ... 501
2.2 L'opérateur [] ... 502
2.3 Affectation et changement de taille ... 502
2.4 Calcul vectoriel ... 502
2.5 Sélection de valeurs par masque ... 504
2.6 Sections de vecteurs ... 505
2.7 Vecteurs d'indices ... 506
3 - La classe bitset ... 507

Chapitre 24 : Les espaces de noms ... 511

1 - Création d'espaces de noms ... 511
1.1 Exemple de création d'un nouvel espace de noms ... 512
1.2 Exemple avec deux espaces de noms ... 513

11 - Initialisation de tableaux de caractères 538
12 - Les noms de fonctions 538

Annexe C : Compléments sur les exceptions 539
1 - Les problèmes posés par les objets automatiques 539
2 - La technique de gestion de ressources par initialisation 540
3 - Le concept de pointeur intelligent : la classe auto_ptr 542

Annexe D : Les différentes sortes de fonctions en C++ 545

Annexe E : Comptage de références 547

Annexe F : Les pointeurs sur des membres 551
1 - Rappels sur les pointeurs sur des fonctions en C 551
2 - Les pointeurs sur des fonctions membres 552
3 - Les pointeurs sur des membres données 553
4 - L'héritage et les pointeurs sur des membres 554

Annexe G : Les algorithmes standard 557
1 - Algorithmes d'initialisation de séquences existantes 558
2 - Algorithmes de recherche 559
3 - Algorithmes de transformation d'une séquence 561
4 - Algorithmes de suppression 564
5 - Algorithmes de tri 566
6 - Algorithmes de recherche et de fusions sur des séquences ordonnées 568
7 - Algorithmes à caractère numérique 570
8 - Algorithmes à caractère ensembliste 571
9 - Algorithmes de manipulation de tas 574
10 - Algorithmes divers 575

Correction des exercices 577

Index 593

1.3 Espace de noms et fichier en-tête 514
1.4 Instructions figurant dans un espace de noms 514
1.5 Création incrémentale d'espaces de noms 515
2 - Les instructions using 516
2.1 La déclaration using pour les symboles 516
2.1.1 Présentation générale 516
2.1.2 Masquage et ambiguïtés 518
2.2 La directive using pour les espaces de noms 519
3 - Espaces de noms et surdéfinition 521
4 - Imbrication des espaces de noms 523
5 - Transitivité de la directive using 523
6 - Les alias 524
7 - Les espaces anonymes 524
8 - Espaces de noms et déclaration d'amitié 525

Annexes 527

Annexe A : Mise en correspondance d'arguments 529
1 - Détermination des fonctions candidates 529
2 - Algorithme de recherche d'une fonction à un seul argument 530
2.1 Recherche d'une correspondance exacte 530
2.2 Promotions numériques 531
2.3 Conversions standard 531
2.4 Conversions définies par l'utilisateur 532
2.5 Fonctions à arguments variables 532
2.6 Exception : cas des champs de bits 532
3 - Fonctions à plusieurs arguments 533
4 - Fonctions membres 533

Annexe B : Utilisation de code écrit en C 535
1 - Prototypes 535
2 - Fonctions sans arguments 535
3 - Fonctions sans valeur de retour 535
4 - Le qualificatif const 536
5 - Les pointeurs de type void * 536
6 - Mots clés 536
7 - Les constantes de type caractère 537
8 - Les définitions multiples 537
9 - L'instruction goto 538
10 - Les énumérations 538