

**Claude Delannoy**

**Apprendre**

**le C++**

**EYROLLES**

# Table des matières

<b>Avant-propos</b> .....	XXIX
<b>1 - Historique de C++</b> .....	XXIX
<b>2 - Objectif et structure de l'ouvrage</b> .....	XXIX
<b>3 - L'ouvrage, C, C++ et Java</b> .....	XXX
<b>Chapitre 1 : Présentation du langage C++</b> .....	1
<b>1 - Programmation structurée et programmation orientée objet</b> .....	2
1.1 Problématique de la programmation .....	2
1.2 La programmation structurée .....	2
1.3 Les apports de la programmation orientée objet .....	3
1.3.1 <i>Objet</i> .....	3
1.3.2 <i>Encapsulation</i> .....	3
1.3.3 <i>Classe</i> .....	4
1.3.4 <i>Héritage</i> .....	4
1.3.5 <i>Polymorphisme</i> .....	4
1.4 P.O.O., langages de programmation et C++ .....	4
<b>2 - C++ et la programmation structurée</b> .....	5
<b>3 - C++ et la programmation orientée objet</b> .....	6
<b>4 - C et C++</b> .....	8
<b>5 - C++ et la bibliothèque standard</b> .....	8
<b>Chapitre 2 : Généralités sur le langage C++</b> .....	11
<b>1 - Présentation par l'exemple de quelques instructions du langage C++</b> .....	12
1.1 Un exemple de programme en langage C++ .....	12
1.2 Structure d'un programme en langage C++ .....	13
1.3 Déclarations .....	13
1.4 Pour écrire des informations : utiliser le flot cout .....	14
1.5 Pour faire une répétition : l'instruction for .....	14

1.6 Pour lire des informations : utiliser le flot cin . . . . .	15
1.7 Pour faire des choix : l'instruction if . . . . .	15
1.8 Les directives à destination du préprocesseur . . . . .	16
1.9 L'instruction using . . . . .	17
1.10 Exemple de programme utilisant le type caractère . . . . .	17
<b>2 - Quelques règles d'écriture . . . . .</b>	<b>18</b>
2.1 Les identificateurs . . . . .	18
2.2 Les mots-clés . . . . .	19
2.3 Les séparateurs . . . . .	19
2.4 Le format libre . . . . .	19
2.5 Les commentaires . . . . .	20
2.5.1 Les commentaires libres . . . . .	20
2.5.2 Les commentaires de fin de ligne . . . . .	21
<b>3 - Création d'un programme en C++ . . . . .</b>	<b>21</b>
3.1 L'édition du programme . . . . .	22
3.2 La compilation . . . . .	22
3.3 L'édition de liens . . . . .	22
3.4 Les fichiers en-tête . . . . .	23
 <b>Chapitre 3 : Les types de base de C++ . . . . .</b>	 <b>25</b>
<b>1 - La notion de type . . . . .</b>	<b>25</b>
<b>2 - Les types entiers . . . . .</b>	<b>26</b>
2.1 Les différents types usuels d'entiers prévus par C++ . . . . .	26
2.2 Leur représentation en mémoire . . . . .	27
2.3 Les types entiers non signés . . . . .	28
2.4 Notation des constantes entières . . . . .	28
<b>3 - Les types flottants . . . . .</b>	<b>29</b>
3.1 Les différents types et leur représentation en mémoire . . . . .	29
3.2 Notation des constantes flottantes . . . . .	30
<b>4 - Les types caractères . . . . .</b>	<b>31</b>
4.1 La notion de caractère en langage C++ . . . . .	31
4.2 Notation des constantes caractères . . . . .	31
<b>5 - Initialisation et constantes . . . . .</b>	<b>33</b>
<b>6 - Le type bool . . . . .</b>	<b>34</b>
 <b>Chapitre 4 : Opérateurs et expressions . . . . .</b>	 <b>35</b>
<b>1 - Originalité des notions d'opérateur et d'expression en C++ . . . . .</b>	<b>35</b>
<b>2 - Les opérateurs arithmétiques en C++ . . . . .</b>	<b>37</b>
2.1 Présentation des opérateurs . . . . .	37
2.2 Les priorités relatives des opérateurs . . . . .	38
2.3 Comportement des opérateurs en cas d'exception . . . . .	38

<b>3 - Les conversions implicites pouvant intervenir dans un calcul d'expression</b> .....	40
3.1 Notion d'expression mixte .....	40
3.2 Les conversions usuelles d'ajustement de type .....	40
3.3 Les promotions numériques usuelles .....	41
3.3.1 Généralités .....	41
3.3.2 Cas du type <i>char</i> .....	42
3.3.3 Cas du type <i>bool</i> .....	43
3.4 Les conversions en présence de types non signés .....	43
3.4.1 Cas des entiers .....	43
3.4.2 Cas des caractères .....	44
<b>4 - Les opérateurs relationnels</b> .....	45
<b>5 - Les opérateurs logiques</b> .....	47
5.1 Rôle .....	47
5.2 Court-circuit dans l'évaluation de <i>&amp;&amp;</i> et <i>  </i> .....	48
<b>6 - L'opérateur d'affectation ordinaire</b> .....	49
6.1 Notion de lvalue .....	49
6.2 L'opérateur d'affectation possède une associativité de droite à gauche .....	50
6.3 L'affectation peut entraîner une conversion .....	50
<b>7 - Opérateurs d'incrément et de décrémentation</b> .....	50
7.1 Leur rôle .....	50
7.2 Leurs priorités .....	52
7.3 Leur intérêt .....	52
<b>8 - Les opérateurs d'affectation élargie</b> .....	52
<b>9 - Les conversions forcées par une affectation</b> .....	53
9.1 Cas usuels .....	53
9.2 Prise en compte d'un attribut de signe .....	54
<b>10 - L'opérateur de cast</b> .....	54
<b>11 - L'opérateur conditionnel</b> .....	55
<b>12 - L'opérateur séquentiel</b> .....	56
<b>13 - L'opérateur sizeof</b> .....	58
<b>14 - Les opérateurs de manipulation de bits</b> .....	58
14.1 Présentation des opérateurs de manipulation de bits .....	58
14.2 Les opérateurs bit à bit .....	59
14.3 Les opérateurs de décalage .....	60
14.4 Exemples d'utilisation des opérateurs de bits .....	60
<b>15 - Récapitulatif des priorités de tous les opérateurs</b> .....	61
 <b>Chapitre 5 : Les entrées-sorties conversationnelles de C++</b> .....	 63
<b>1 - Affichage à l'écran</b> .....	63
1.1 Exemple 1 .....	64
1.2 Exemple 2 .....	64
1.3 Les possibilités d'écriture sur <i>cout</i> .....	65

<b>2 - Lecture au clavier</b> .....	66
2.1 Introduction .....	66
2.2 Les différentes possibilités de lecture sur cin .....	67
2.3 Notions de tampon et de caractères séparateurs .....	67
2.4 Premières règles utilisées par >> .....	67
2.5 Présence d'un caractère invalide dans une donnée .....	68
2.6 Les risques induits par la lecture au clavier .....	69
2.6.1 Manque de synchronisme entre clavier et écran .....	69
2.6.2 Blocage de la lecture .....	70
2.6.3 Boucle infinie sur un caractère invalide .....	70
 <b>Chapitre 6 : Les instructions de contrôle</b> .....	 73
<b>1 - Les blocs d'instructions</b> .....	74
1.1 Blocs d'instructions .....	74
1.2 Déclarations dans un bloc .....	75
<b>2 - L'instruction if</b> .....	75
2.1 Syntaxe de l'instruction if .....	76
2.2 Exemples .....	76
2.3 Imbrication des instructions if .....	77
<b>3 - L'instruction switch</b> .....	79
3.1 Exemples d'introduction de l'instruction switch .....	79
3.2 Syntaxe de l'instruction switch .....	82
<b>4 - L'instruction do... while</b> .....	84
4.1 Exemple d'introduction de l'instruction do... while .....	84
4.2 Syntaxe de l'instruction do... while .....	85
<b>5 - L'instruction while</b> .....	86
5.1 Exemple d'introduction de l'instruction while .....	86
5.2 Syntaxe de l'instruction while .....	87
<b>6 - L'instruction for</b> .....	88
6.1 Exemple d'introduction de l'instruction for .....	88
6.2 L'instruction for en général .....	89
6.3 Syntaxe de l'instruction for .....	90
<b>7 - Les instructions de branchement inconditionnel : break, continue et goto</b> .....	93
7.1 L'instruction break .....	93
7.2 L'instruction continue .....	94
7.3 L'instruction goto .....	95
 <b>Chapitre 7 : Les fonctions</b> .....	 97
<b>1 - Exemple de définition et d'utilisation d'une fonction</b> .....	98
<b>2 - Quelques règles</b> .....	100
2.1 Arguments muets et arguments effectifs .....	100
2.2 L'instruction return .....	100
2.3 Cas des fonctions sans valeur de retour ou sans argument .....	101

<b>3 - Les fonctions et leurs déclarations</b> .....	103
3.1 Les différentes façons de déclarer une fonction .....	103
3.2 Où placer la déclaration d'une fonction .....	103
3.3 Contrôles et conversions induites par le prototype .....	104
<b>4 - Transmission des arguments par valeur</b> .....	104
<b>5 - Transmission par référence</b> .....	106
5.1 Exemple de transmission d'argument par référence .....	106
5.2 Propriétés de la transmission par référence d'un argument .....	107
5.2.1 <i>Induction de risques indirects</i> .....	107
5.2.2 <i>Absence de conversion</i> .....	108
5.2.3 <i>Cas d'un argument effectif constant</i> .....	108
5.2.4 <i>Cas d'un argument muet constant</i> .....	108
<b>6 - Les variables globales</b> .....	109
6.1 Exemple d'utilisation de variables globales .....	109
6.2 La portée des variables globales .....	110
6.3 La classe d'allocation des variables globales .....	111
<b>7 - Les variables locales</b> .....	111
7.1 La portée des variables locales .....	111
7.2 Les variables locales automatiques .....	112
7.3 Les variables locales statiques .....	113
7.4 Variables locales à un bloc .....	114
7.5 Le cas des fonctions récursives .....	115
<b>8 - Initialisation des variables</b> .....	116
8.1 Les variables de classe statique .....	116
8.2 Les variables de classe automatique .....	116
<b>9 - Les arguments par défaut</b> .....	117
9.1 Exemples .....	117
9.2 Les propriétés des arguments par défaut .....	118
<b>10 - Surdéfinition de fonctions</b> .....	119
10.1 Mise en œuvre de la surdéfinition de fonctions .....	120
10.2 Exemples de choix d'une fonction surdéfinie .....	121
10.3 Règles de recherche d'une fonction surdéfinie .....	123
10.3.1 <i>Cas des fonctions à un argument</i> .....	123
10.3.2 <i>Cas des fonctions à plusieurs arguments</i> .....	124
<b>11 - Les arguments variables en nombre</b> .....	124
11.1 Premier exemple .....	125
11.2 Second exemple .....	126
<b>12 - La conséquence de la compilation séparée</b> .....	127
12.1 Compilation séparée et prototypes .....	127
12.2 Fonction manquante lors de l'édition de liens .....	128
12.3 Le mécanisme de la surdéfinition de fonctions .....	129

12.4	Compilation séparée et variables globales. . . . .	130
12.4.1	<i>La portée d'une variable globale – la déclaration extern</i> . . . . .	130
12.4.2	<i>Les variables globales et l'édition de liens</i> . . . . .	131
12.4.3	<i>Les variables globales cachées – la déclaration static</i> . . . . .	132
<b>13</b>	<b>- Compléments sur les références</b> . . . . .	<b>132</b>
13.1	Transmission par référence d'une valeur de retour . . . . .	132
13.1.1	<i>Introduction</i> . . . . .	133
13.1.2	<i>On obtient une lvalue</i> . . . . .	133
13.1.3	<i>Conversion</i> . . . . .	134
13.1.4	<i>Valeur de retour et constance</i> . . . . .	134
13.2	La référence d'une manière générale. . . . .	135
13.2.1	<i>La notion de référence est plus générale que celle d'argument.</i> . . . . .	135
13.2.2	<i>Initialisation de référence</i> . . . . .	135
<b>14</b>	<b>- La spécification inline</b> . . . . .	<b>136</b>
 <b>Chapitre 8 : Les tableaux et les pointeurs</b> . . . . .		<b>139</b>
<b>1</b>	<b>- Les tableaux à un indice</b> . . . . .	<b>140</b>
1.1	Exemple d'utilisation d'un tableau en C++ . . . . .	140
1.2	Quelques règles . . . . .	141
1.2.1	<i>Les éléments de tableau</i> . . . . .	141
1.2.2	<i>Les indices</i> . . . . .	141
1.2.3	<i>La dimension d'un tableau</i> . . . . .	142
1.2.4	<i>Débordement d'indice</i> . . . . .	142
<b>2</b>	<b>- Les tableaux à plusieurs indices</b> . . . . .	<b>143</b>
2.1	Leur déclaration. . . . .	143
2.2	Arrangement en mémoire des tableaux à plusieurs indices. . . . .	143
<b>3</b>	<b>- Initialisation des tableaux</b> . . . . .	<b>144</b>
3.1	Initialisation de tableaux à un indice . . . . .	144
3.2	Initialisation de tableaux à plusieurs indices . . . . .	144
3.3	Initialiseurs et classe d'allocation . . . . .	145
<b>4</b>	<b>- Notion de pointeur – Les opérateurs * et &amp;</b> . . . . .	<b>146</b>
4.1	Introduction . . . . .	146
4.2	Quelques exemples . . . . .	147
4.3	Incrémentement de pointeurs . . . . .	148
<b>5</b>	<b>- Comment simuler une transmission par adresse avec un pointeur</b> . . . . .	<b>149</b>
<b>6</b>	<b>- Un nom de tableau est un pointeur constant</b> . . . . .	<b>151</b>
6.1	Cas des tableaux à un indice . . . . .	151
6.2	Cas des tableaux à plusieurs indices . . . . .	152
<b>7</b>	<b>- Les opérations réalisables sur des pointeurs</b> . . . . .	<b>153</b>
7.1	La comparaison de pointeurs . . . . .	153
7.2	La soustraction de pointeurs . . . . .	154
7.3	Les affectations de pointeurs et le pointeur nul . . . . .	154

7.4 Les conversions de pointeurs .....	154
7.5 Les pointeurs génériques .....	155
<b>8 - La gestion dynamique : les opérateurs new et delete .....</b>	<b>157</b>
8.1 L'opérateur new .....	157
8.2 L'opérateur delete .....	159
8.3 Exemple .....	159
<b>9 - Pointeurs et surdéfinition de fonctions .....</b>	<b>161</b>
<b>10 - Les tableaux transmis en argument .....</b>	<b>162</b>
10.1 Cas des tableaux à un indice .....	162
10.1.1 Premier exemple : tableau de taille fixe .....	162
10.1.2 Second exemple : tableau de taille variable .....	164
10.2 Cas des tableaux à plusieurs indices .....	164
10.2.1 Premier exemple : tableau de taille fixe .....	164
10.2.2 Second exemple : tableau de dimensions variables .....	165
<b>11 - Utilisation de pointeurs sur des fonctions .....</b>	<b>166</b>
11.1 Paramétrage d'appel de fonctions .....	166
11.2 Fonctions transmises en argument .....	167
<b>Chapitre 9 : Les chaînes de style C .....</b>	<b>169</b>
<b>1 - Représentation des chaînes .....</b>	<b>170</b>
1.1 La convention adoptée .....	170
1.2 Cas des chaînes constantes .....	170
<b>2 - Lecture et écriture de chaînes de style C .....</b>	<b>172</b>
<b>3 - Initialisation de tableaux par des chaînes .....</b>	<b>173</b>
3.1 Initialisation de tableaux de caractères .....	173
3.2 Initialisation de tableaux de pointeurs sur des chaînes .....	174
<b>4 - Les arguments transmis à la fonction main .....</b>	<b>175</b>
4.1 Comment passer des arguments à un programme .....	175
4.2 Comment récupérer ces arguments dans la fonction main .....	176
<b>5 - Généralités sur les fonctions portant sur des chaînes de style C .....</b>	<b>177</b>
5.1 Ces fonctions travaillent toujours sur des adresses .....	177
5.2 La fonction strlen .....	178
5.3 Le cas des fonctions de concaténation .....	178
<b>6 - Les fonctions de concaténation de chaînes .....</b>	<b>178</b>
6.1 La fonction strcat .....	178
6.2 La fonction strncat .....	179
<b>7 - Les fonctions de comparaison de chaînes .....</b>	<b>180</b>
<b>8 - Les fonctions de copie de chaînes .....</b>	<b>181</b>
<b>9 - Les fonctions de recherche dans une chaîne .....</b>	<b>182</b>
<b>10 - Quelques précautions à prendre avec les chaînes de style C .....</b>	<b>182</b>
10.1 Une chaîne de style C possède une vraie fin, mais pas de vrai début .....	183
10.2 Les risques de modification des chaînes constantes .....	183

<b>Chapitre 10 : Les types structure, union et énumération</b> .....	185
<b>1 - Déclaration d'une structure</b> .....	186
<b>2 - Utilisation d'une structure</b> .....	187
2.1 Utilisation des champs d'une structure .....	187
2.2 Utilisation globale d'une structure .....	188
2.3 Initialisation de structures .....	188
<b>3 - Imbrication de structures</b> .....	189
3.1 Structure comportant des tableaux .....	190
3.2 Tableaux de structures .....	191
3.3 Structures comportant d'autres structures .....	191
3.4 Cas particulier de structure renfermant un pointeur .....	192
<b>4 - À propos de la portée du type de structure</b> .....	192
<b>5 - Transmission d'une structure en argument d'une fonction</b> .....	193
5.1 Transmission d'une structure par valeur .....	194
5.2 Transmission d'une structure par référence .....	194
5.3 Transmission de l'adresse d'une structure : l'opérateur -> .....	195
<b>6 - Transmission d'une structure en valeur de retour d'une fonction</b> .....	196
<b>7 - Les champs de bits</b> .....	197
<b>8 - Les unions</b> .....	198
<b>9 - Les énumérations</b> .....	200
9.1 Exemples introductifs .....	200
9.2 Propriétés du type énumération .....	201
 <b>Chapitre 11 : Classes et objets</b> .....	 203
<b>1 - Les structures généralisées</b> .....	204
1.1 Déclaration des fonctions membres d'une structure .....	204
1.2 Définition des fonctions membres d'une structure .....	205
1.3 Utilisation d'une structure généralisée .....	206
1.4 Exemple récapitulatif .....	207
<b>2 - Notion de classe</b> .....	208
<b>3 - Affectation d'objets</b> .....	212
<b>4 - Notions de constructeur et de destructeur</b> .....	213
4.1 Introduction .....	213
4.2 Exemple de classe comportant un constructeur .....	214
4.3 Construction et destruction des objets .....	216
4.4 Rôles du constructeur et du destructeur .....	217
4.5 Quelques règles .....	220
<b>5 - Les membres données statiques</b> .....	221
5.1 Le qualificatif static pour un membre donnée .....	221
5.2 Initialisation des membres données statiques .....	222
5.3 Exemple .....	223

<b>6 - Exploitation d'une classe</b> .....	225
6.1 La classe comme composant logiciel .....	225
6.2 Protection contre les inclusions multiples .....	227
6.3 Cas des membres données statiques .....	227
6.4 En cas de modification d'une classe .....	227
6.4.1 <i>La déclaration des membres publics n'a pas changé</i> .....	228
6.4.2 <i>La déclaration des membres publics a changé</i> .....	228
<b>7 - Les classes en général</b> .....	228
7.1 Les autres sortes de classes en C++ .....	228
7.2 Ce qu'on peut trouver dans la déclaration d'une classe .....	229
7.3 Déclaration d'une classe .....	230
<b>Chapitre 12 : Les propriétés des fonctions membres</b> .....	231
<b>1 - Surdéfinition des fonctions membres</b> .....	231
<b>2 - Arguments par défaut</b> .....	234
<b>3 - Les fonctions membres en ligne</b> .....	235
<b>4 - Cas des objets transmis en argument d'une fonction membre</b> .....	237
<b>5 - Mode de transmission des objets en argument</b> .....	239
5.1 Transmission de l'adresse d'un objet .....	239
5.2 Transmission par référence .....	241
5.3 Les problèmes posés par la transmission par valeur .....	242
<b>6 - Lorsqu'une fonction renvoie un objet</b> .....	242
<b>7 - Autoréférence : le mot clé this</b> .....	243
<b>8 - Les fonctions membres statiques</b> .....	244
<b>9 - Les fonctions membres constantes</b> .....	246
9.1 Définition d'une fonction membre constante .....	246
9.2 Propriétés d'une fonction membre constante .....	247
<b>10 - Les membres mutables</b> .....	249
<b>Chapitre 13 : Construction, destruction et initialisation des objets</b> .....	251
<b>1 - Les objets automatiques et statiques</b> .....	252
1.1 Durée de vie .....	252
1.2 Appel des constructeurs et des destructeurs .....	253
1.3 Exemple .....	253
<b>2 - Les objets dynamiques</b> .....	255
2.1 Cas d'une classe sans constructeur .....	255
2.2 Cas d'une classe avec constructeur .....	256
2.3 Exemple .....	257

<b>3 - Le constructeur de copie</b> .....	258
3.1 Présentation .....	258
3.1.1 <i>Il n'existe pas de constructeur approprié</i> .....	258
3.1.2 <i>Il existe un constructeur approprié</i> .....	259
3.1.3 <i>Lorsqu'on souhaite interdire la construction par copie</i> .....	259
3.2 Exemple 1 : objet transmis par valeur .....	260
3.2.1 <i>Emploi du constructeur de copie par défaut</i> .....	261
3.2.2 <i>Définition d'un constructeur de copie</i> .....	262
3.3 Exemple 2 : objet en valeur de retour d'une fonction .....	265
<b>4 - Initialisation d'un objet lors de sa déclaration</b> .....	266
<b>5 - Objets membres</b> .....	268
5.1 Introduction .....	268
5.2 Mise en œuvre des constructeurs et des destructeurs .....	269
5.3 Le constructeur de copie .....	271
<b>6 - Initialisation de membres dans l'en-tête d'un constructeur</b> .....	272
<b>7 - Les tableaux d'objets</b> .....	273
7.1 Notations .....	273
7.2 Constructeurs et initialiseurs .....	274
7.3 Cas des tableaux dynamiques d'objets .....	275
<b>8 - Les objets temporaires</b> .....	276
 <b>Chapitre 14 : Les fonctions amies</b> .....	279
<b>1 - Exemple de fonction indépendante amie d'une classe</b> .....	280
<b>2 - Les différentes situations d'amitié</b> .....	282
2.1 <i>Fonction membre d'une classe, amie d'une autre classe</i> .....	283
2.2 <i>Fonction amie de plusieurs classes</i> .....	284
2.3 <i>Toutes les fonctions d'une classe amies d'une autre classe</i> .....	285
<b>3 - Exemple</b> .....	285
3.1 <i>Fonction amie indépendante</i> .....	286
3.2 <i>Fonction amie, membre d'une classe</i> .....	287
<b>4 - Exploitation de classes disposant de fonctions amies</b> .....	288
 <b>Chapitre 15 : La surdéfinition d'opérateurs</b> .....	291
<b>1 - Le mécanisme de la surdéfinition d'opérateurs</b> .....	292
1.1 <i>Surdéfinition d'opérateur avec une fonction amie</i> .....	293
1.2 <i>Surdéfinition d'opérateur avec une fonction membre</i> .....	294
1.3 <i>Opérateurs et transmission par référence</i> .....	296
<b>2 - La surdéfinition d'opérateurs en général</b> .....	297
2.1 <i>Se limiter aux opérateurs existants</i> .....	297
2.2 <i>Se placer dans un contexte de classe</i> .....	299
2.3 <i>Éviter les hypothèses sur le rôle d'un opérateur</i> .....	299
2.4 <i>Cas des opérateurs ++ et --</i> .....	300

2.5 L'opérateur = possède une signification prédéfinie .....	301
2.6 Les conversions .....	302
2.7 Choix entre fonction membre et fonction amie .....	303
<b>3 - Surdéfinition de l'opérateur =</b> .....	<b>303</b>
3.1 Rappels concernant le constructeur par recopie .....	303
3.2 Cas de l'affectation .....	304
3.3 Algorithme proposé .....	305
3.4 Valeur de retour .....	307
3.5 En définitive .....	307
3.6 Exemple de programme complet .....	307
3.7 Lorsqu'on souhaite interdire l'affectation .....	309
<b>4 - La forme canonique d'une classe</b> .....	<b>310</b>
4.1 Cas général .....	310
<b>5 - Exemple de surdéfinition de l'opérateur []</b> .....	<b>311</b>
<b>6 - Surdéfinition de l'opérateur ()</b> .....	<b>314</b>
<b>7 - Surdéfinition des opérateurs new et delete</b> .....	<b>314</b>
7.1 Surdéfinition de new et delete pour une classe donnée .....	315
7.2 Exemple .....	315
7.3 D'une manière générale .....	317
<b>Chapitre 16 : Les conversions de type définies par l'utilisateur</b> .....	<b>319</b>
<b>1 - Les différentes sortes de conversions définies par l'utilisateur</b> .....	<b>320</b>
<b>2 - L'opérateur de cast pour la conversion type classe → type de base</b> .....	<b>322</b>
2.1 Définition de l'opérateur de cast .....	322
2.2 Exemple d'utilisation .....	322
2.3 Appel implicite de l'opérateur de cast lors d'un appel de fonction .....	324
2.4 Appel implicite de l'opérateur de cast dans l'évaluation d'une expression .....	325
2.5 Conversions en chaîne .....	327
2.6 En cas d'ambiguïté .....	329
<b>3 - Le constructeur pour la conversion type de base → type classe</b> .....	<b>329</b>
3.1 Exemple .....	329
3.2 Le constructeur dans une chaîne de conversions .....	331
3.3 Choix entre constructeur ou opérateur d'affectation .....	332
3.4 Emploi d'un constructeur pour élargir la signification d'un opérateur .....	333
3.5 Interdire les conversions implicites par le constructeur : le rôle d'explicit .....	336
<b>4 - Les conversions d'un type classe en un autre type classe</b> .....	<b>336</b>
4.1 Exemple simple d'opérateur de cast .....	336
4.2 Exemple de conversion par un constructeur .....	337
4.3 Pour donner une signification à un opérateur défini dans une autre classe .....	339
<b>5 - Quelques conseils</b> .....	<b>341</b>
<b>Chapitre 17 : Les patrons de fonctions</b> .....	<b>343</b>
<b>1 - Exemple de création et d'utilisation d'un patron de fonctions</b> .....	<b>344</b>
1.1 Création d'un patron de fonctions .....	344
1.2 Premières utilisations du patron de fonctions .....	345

1.3 Autres utilisations du patron de fonctions . . . . .	346
1.3.1 Application au type <i>char *</i> . . . . .	346
1.3.2 Application à un type classe . . . . .	347
1.4 Contraintes d'utilisation d'un patron . . . . .	348
<b>2 - Les paramètres de type d'un patron de fonctions</b> . . . . .	349
2.1 Utilisation des paramètres de type dans la définition d'un patron . . . . .	349
2.2 Identification des paramètres de type d'une fonction patron . . . . .	350
2.3 Nouvelle syntaxe d'initialisation des variables des types standard . . . . .	351
2.4 Limitations des patrons de fonctions . . . . .	352
<b>3 - Les paramètres expressions d'un patron de fonctions</b> . . . . .	353
<b>4 - Surdéfinition de patrons</b> . . . . .	354
4.1 Exemples ne comportant que des paramètres de type . . . . .	354
4.2 Exemples comportant des paramètres expressions . . . . .	357
<b>5 - Spécialisation de fonctions de patron</b> . . . . .	358
5.1 Généralités . . . . .	358
5.2 Les spécialisations partielles . . . . .	358
<b>6 - Algorithme d'instanciation d'une fonction patron</b> . . . . .	359
<b>Chapitre 18 : Les patrons de classes</b> . . . . .	363
<b>1 - Exemple de création et d'utilisation d'un patron de classes</b> . . . . .	364
1.1 Création d'un patron de classes . . . . .	364
1.2 Utilisation d'un patron de classes . . . . .	366
1.3 Contraintes d'utilisation d'un patron de classes . . . . .	366
1.4 Exemple récapitulatif . . . . .	367
<b>2 - Les paramètres de type d'un patron de classes</b> . . . . .	369
2.1 Les paramètres de type dans la création d'un patron de classes . . . . .	369
2.2 Instanciation d'une classe patron . . . . .	369
<b>3 - Les paramètres expressions d'un patron de classes</b> . . . . .	370
3.1 Exemple . . . . .	371
3.2 Les propriétés des paramètres expressions . . . . .	372
<b>4 - Spécialisation d'un patron de classes</b> . . . . .	373
4.1 Exemple de spécialisation d'une fonction membre . . . . .	373
4.2 Les différentes possibilités de spécialisation . . . . .	374
4.2.1 On peut spécialiser une fonction membre pour tous les paramètres . . . . .	374
4.2.2 On peut spécialiser une fonction membre ou une classe . . . . .	375
4.2.3 On peut prévoir des spécialisations partielles de patrons de classes . . . . .	375
<b>5 - Paramètres par défaut</b> . . . . .	376
<b>6 - Patrons de fonctions membres</b> . . . . .	376
<b>7 - Identité de classes patrons</b> . . . . .	376
<b>8 - Classes patrons et déclarations d'amitié</b> . . . . .	377
8.1 Déclaration de classes ou fonctions « ordinaires » amies . . . . .	377
8.2 Déclaration d'instances particulières de classes patrons ou de fonctions patrons . . . . .	378
8.3 Déclaration d'un autre patron de fonctions ou de classes . . . . .	378
<b>9 - Exemple de classe tableau à deux indices</b> . . . . .	379

<b>Chapitre 19 : L'héritage simple</b> .....	383
<b>1 - La notion d'héritage</b> .....	384
<b>2 - Utilisation des membres de la classe de base dans une classe dérivée</b> .....	386
<b>3 - Redéfinition des membres d'une classe dérivée</b> .....	388
3.1 Redéfinition des fonctions membres d'une classe dérivée .....	388
3.2 Redéfinition des membres données d'une classe dérivée .....	390
3.3 Redéfinition et surdéfinition .....	390
<b>4 - Appel des constructeurs et des destructeurs</b> .....	392
4.1 Rappels .....	392
4.2 La hiérarchisation des appels .....	393
4.3 Transmission d'informations entre constructeurs .....	393
4.4 Exemple .....	395
4.5 Compléments .....	396
<b>5 - Contrôle des accès</b> .....	397
5.1 Les membres protégés .....	397
5.2 Exemple .....	398
5.3 Intérêt du statut protégé .....	398
5.4 Dérivation publique et dérivation privée .....	399
5.4.1 Rappels concernant la dérivation publique .....	399
5.4.2 Dérivation privée .....	400
5.4.3 Les possibilités de dérivation protégée .....	401
5.5 Récapitulation .....	402
<b>6 - Compatibilité entre classe de base et classe dérivée</b> .....	403
6.1 Conversion d'un type dérivé en un type de base .....	404
6.2 Conversion de pointeurs .....	404
6.3 Limitations liées au typage statique des objets .....	405
6.4 Les risques de violation des protections de la classe de base .....	408
<b>7 - Le constructeur de recopie et l'héritage</b> .....	409
7.1 La classe dérivée ne définit pas de constructeur de recopie .....	409
7.2 La classe dérivée définit un constructeur de recopie .....	410
<b>8 - L'opérateur d'affectation et l'héritage</b> .....	412
8.1 La classe dérivée ne surdéfinit pas l'opérateur = .....	412
8.2 La classe dérivée surdéfinit l'opérateur = .....	412
<b>9 - Héritage et forme canonique d'une classe</b> .....	415
<b>10 - L'héritage et ses limites</b> .....	417
10.1 La situation d'héritage .....	417
10.1.1 Le type du résultat de l'appel .....	418
10.1.2 Le type des arguments de <i>f</i> .....	418
10.2 Exemples .....	418
10.2.1 Héritage dans pointcol d'un opérateur + défini dans point .....	419
10.2.2 Héritage dans pointcol de la fonction coincide de point .....	419

<b>11 - Exemple de classe dérivée</b> .....	420
<b>12 - Patrons de classes et héritage</b> .....	423
12.1 Classe « ordinaire » dérivant d'une classe patron .....	424
12.2 Dérivation de patrons avec les mêmes paramètres .....	425
12.3 Dérivation de patrons avec introduction d'un nouveau paramètre .....	425
<b>13 - L'héritage en pratique</b> .....	426
13.1 Dérivations successives .....	426
13.2 Différentes utilisations de l'héritage .....	428
13.3 Exploitation d'une classe dérivée .....	428
<b>Chapitre 20 : L'héritage multiple</b> .....	431
<b>1 - Mise en œuvre de l'héritage multiple</b> .....	432
<b>2 - Pour régler les éventuels conflits : les classes virtuelles</b> .....	436
<b>3 - Appels des constructeurs et des destructeurs : cas des classes virtuelles</b> .....	437
<b>4 - Exemple d'utilisation de l'héritage multiple et de la dérivation virtuelle</b> .....	440
<b>Chapitre 21 : Les fonctions virtuelles et le polymorphisme</b> .....	443
<b>1 - Rappel d'une situation où le typage dynamique est nécessaire</b> .....	444
<b>2 - Le mécanisme des fonctions virtuelles</b> .....	444
<b>3 - Autre situation où la ligature dynamique est indispensable</b> .....	446
<b>4 - Les propriétés des fonctions virtuelles</b> .....	449
4.1 Leurs limitations sont celles de l'héritage .....	449
4.2 La redéfinition d'une fonction virtuelle n'est pas obligatoire .....	450
4.3 Fonctions virtuelles et surdéfinition .....	451
4.4 Le type de retour d'une fonction virtuelle redéfinie .....	451
4.5 On peut déclarer une fonction virtuelle dans n'importe quelle classe .....	452
4.6 Quelques restrictions et conseils .....	452
4.6.1 <i>Seule une fonction membre peut être virtuelle</i> .....	452
4.6.2 <i>Un constructeur ne peut pas être virtuel</i> .....	452
4.6.3 <i>Un destructeur peut être virtuel</i> .....	453
4.6.4 <i>Cas particulier de l'opérateur d'affectation</i> .....	454
<b>5 - Les fonctions virtuelles pures pour la création de classes abstraites</b> .....	455
<b>6 - Exemple d'utilisation de fonctions virtuelles : liste hétérogène</b> .....	457
<b>7 - Le mécanisme d'identification dynamique des objets</b> .....	461
<b>8 - Identification de type à l'exécution</b> .....	463
8.1 Utilisation du champ name de type_info .....	464
8.2 Utilisation des opérateurs de comparaison de type_info .....	465
8.3 Exemple avec des références .....	466
<b>9 - Les cast dynamiques</b> .....	466

<b>Chapitre 22 : Les flots</b> .....	469
<b>1 - Présentation générale de la classe ostream</b> .....	471
1.1 L'opérateur << .....	471
1.2 Les flots prédéfinis .....	472
1.3 La fonction put .....	472
1.4 La fonction write .....	473
1.4.1 <i>Cas des caractères</i> .....	473
1.4.2 <i>Autres cas</i> .....	473
1.5 Quelques possibilités de formatage avec << .....	473
1.5.1 <i>Action sur la base de numération</i> .....	474
1.5.2 <i>Action sur le gabarit de l'information écrite</i> .....	475
1.5.3 <i>Action sur la précision de l'information écrite</i> .....	476
1.5.4 <i>Choix entre notation flottante ou exponentielle</i> .....	477
1.5.5 <i>Un programme de facturation amélioré</i> .....	478
<b>2 - Présentation générale de la classe istream</b> .....	479
2.1 L'opérateur >> .....	479
2.1.1 <i>Cas des caractères</i> .....	480
2.1.2 <i>Cas des chaînes de style C</i> .....	480
2.1.3 <i>Les types acceptés par &gt;&gt;</i> .....	481
2.2 La fonction get .....	481
2.3 Les fonctions getline et gcount .....	482
2.4 La fonction read .....	484
2.4.1 <i>Cas des caractères</i> .....	484
2.4.2 <i>Autres cas</i> .....	484
2.5 Quelques autres fonctions .....	484
<b>3 - Statut d'erreur d'un flot</b> .....	484
3.1 Les bits d'erreur .....	485
3.2 Actions concernant les bits d'erreur .....	485
3.2.1 <i>Accès aux bits d'erreur</i> .....	485
3.2.2 <i>Modification du statut d'erreur</i> .....	486
3.3 Surdéfinition des opérateurs () et ! .....	486
3.4 Exemples .....	487
<b>4 - Surdéfinition de &lt;&lt; et &gt;&gt; pour les types définis par l'utilisateur</b> .....	489
4.1 Méthode .....	489
4.2 Exemple .....	490
<b>5 - Gestion du formatage</b> .....	492
5.1 Le statut de formatage d'un flot .....	493
5.2 Description du mot d'état du statut de formatage .....	494
5.3 Action sur le statut de formatage .....	495
5.3.1 <i>Les manipulateurs non paramétriques</i> .....	495
5.3.2 <i>Les manipulateurs paramétriques</i> .....	496
5.3.3 <i>Les fonctions membres</i> .....	497
5.3.4 <i>Exemple</i> .....	499

<b>6 - Connexion d'un flot à un fichier</b> .....	499
6.1 Connexion d'un flot de sortie à un fichier .....	499
6.2 Connexion d'un flot d'entrée à un fichier .....	501
6.3 Les possibilités d'accès direct .....	502
6.4 Les différents modes d'ouverture d'un fichier .....	504
<b>7 - Les anciennes possibilités de formatage en mémoire</b> .....	505
7.1 La classe ostrstream .....	506
7.2 La classe istrstream .....	507
<b>Chapitre 23 : La gestion des exceptions</b> .....	509
<b>1 - Premier exemple d'exception</b> .....	510
1.1 Comment lancer une exception : l'instruction throw .....	511
1.2 Utilisation d'un gestionnaire d'exception .....	511
1.3 Récapitulatif .....	512
<b>2 - Second exemple</b> .....	514
<b>3 - Le mécanisme de gestion des exceptions</b> .....	516
3.1 Poursuite de l'exécution du programme .....	516
3.2 Prise en compte des sorties de blocs .....	518
<b>4 - Choix du gestionnaire</b> .....	518
4.1 Le gestionnaire reçoit toujours une copie .....	519
4.2 Règles de choix d'un gestionnaire d'exception .....	519
4.3 Le cheminement des exceptions .....	520
4.4 Redéclenchement d'une exception .....	522
<b>5 - Spécification d'interface : la fonction unexpected</b> .....	523
<b>6 - Les exceptions standard</b> .....	526
6.1 Généralités .....	526
6.2 Les exceptions déclenchées par la bibliothèque standard .....	526
6.3 Les exceptions utilisables dans un programme .....	527
6.4 Cas particulier de la gestion dynamique de mémoire .....	527
6.4.1 L'opérateur new (nothrow) .....	527
6.4.2 Gestion des débordements de mémoire avec set_new_handler .....	528
6.5 Création d'exceptions dérivées de la classe exception .....	529
6.5.1 Exemple 1 .....	530
6.5.2 Exemple 2 .....	530
<b>Chapitre 24 : Généralités sur la bibliothèque standard</b> .....	533
<b>1 - Notions de conteneur, d'itérateur et d'algorithme</b> .....	533
1.1 Notion de conteneur .....	534
1.2 Notion d'itérateur .....	534
1.3 Parcours d'un conteneur avec un itérateur .....	535
1.3.1 Parcours direct .....	535
1.3.2 Parcours inverse .....	536

1.4 Intervalle d'itérateur . . . . .	536
1.5 Notion d'algorithme . . . . .	537
1.6 Itérateurs et pointeurs . . . . .	538
<b>2 - Les différentes sortes de conteneurs . . . . .</b>	<b>538</b>
2.1 Conteneurs et structures de données classiques . . . . .	538
2.2 Les différentes catégories de conteneurs . . . . .	539
<b>3 - Les conteneurs dont les éléments sont des objets . . . . .</b>	<b>539</b>
3.1 Construction, copie et affectation . . . . .	540
3.2 Autres opérations . . . . .	541
<b>4 - Efficacité des opérations sur des conteneurs . . . . .</b>	<b>541</b>
<b>5 - Fonctions, prédicats et classes fonctions . . . . .</b>	<b>542</b>
5.1 Fonction unaire . . . . .	542
5.2 Prédicats . . . . .	543
5.3 Classes et objets fonctions . . . . .	543
5.3.1 Utilisation d'objet fonction comme fonction de rappel . . . . .	543
5.3.2 Classes fonctions prédéfinies . . . . .	544
<b>6 - Conteneurs, algorithmes et relation d'ordre . . . . .</b>	<b>545</b>
6.1 Introduction . . . . .	545
6.2 Propriétés à respecter . . . . .	545
<b>7 - Les générateurs d'opérateurs . . . . .</b>	<b>546</b>
<b>Chapitre 25 : Les conteneurs séquentiels . . . . .</b>	<b>549</b>
<b>1 - Fonctionnalités communes aux conteneurs vector, list et deque . . . . .</b>	<b>550</b>
1.1 Construction . . . . .	550
1.1.1 Construction d'un conteneur vide . . . . .	550
1.1.2 Construction avec un nombre donné d'éléments . . . . .	550
1.1.3 Construction avec un nombre donné d'éléments initialisés à une valeur . . . . .	550
1.1.4 Construction à partir d'une séquence . . . . .	551
1.1.5 Construction à partir d'un autre conteneur de même type . . . . .	551
1.2 Modifications globales . . . . .	551
1.2.1 Opérateur d'affectation . . . . .	552
1.2.2 La fonction membre assign . . . . .	552
1.2.3 La fonction clear . . . . .	553
1.2.4 La fonction swap . . . . .	553
1.3 Comparaison de conteneurs . . . . .	553
1.3.1 L'opérateur == . . . . .	553
1.3.2 L'opérateur < . . . . .	554
1.3.3 Exemples . . . . .	554
1.4 Insertion ou suppression d'éléments . . . . .	554
1.4.1 Insertion . . . . .	555
1.4.2 Suppression . . . . .	555
1.4.3 Cas des insertions/suppressions en fin : pop_back et push_back . . . . .	556

<b>2 - Le conteneur vector</b> .....	556
2.1 Accès aux éléments existants .....	557
2.1.1 Accès par itérateur .....	557
2.1.2 Accès par indice .....	557
2.1.3 Cas de l'accès au dernier élément .....	558
2.2 Insertions et suppressions .....	558
2.3 Gestion de l'emplacement mémoire .....	558
2.3.1 Introduction .....	558
2.3.2 Invalidation d'itérateurs ou de références .....	559
2.3.3 Outils de gestion de l'emplacement mémoire d'un vecteur .....	559
2.4 Exemple .....	560
2.5 Cas particulier des vecteurs de booléens .....	561
<b>3 - Le conteneur deque</b> .....	562
3.1 Présentation générale .....	562
3.2 Exemple .....	563
<b>4 - Le conteneur list</b> .....	564
4.1 Accès aux éléments existants .....	564
4.2 Insertions et suppressions .....	564
4.2.1 Suppression des éléments de valeur donnée .....	565
4.2.2 Suppression des éléments répondant à une condition .....	565
4.3 Opérations globales .....	565
4.3.1 Tri d'une liste .....	566
4.3.2 Suppression des éléments en double .....	566
4.3.3 Fusion de deux listes .....	567
4.3.4 Transfert d'une partie de liste dans une autre .....	568
4.4 Gestion de l'emplacement mémoire .....	568
4.5 Exemple .....	569
<b>5 - Les adaptateurs de conteneur : queue, stack et priority_queue</b> .....	570
5.1 L'adaptateur stack .....	570
5.2 L'adaptateur queue .....	571
5.3 L'adaptateur priority_queue .....	572
<b>Chapitre 26 : Les conteneurs associatifs</b> .....	575
<b>1 - Le conteneur map</b> .....	576
1.1 Exemple introductif .....	576
1.2 Le patron de classes pair .....	578
1.3 Construction d'un conteneur de type map .....	578
1.3.1 Constructions utilisant la relation d'ordre par défaut .....	579
1.3.2 Choix de l'ordre intrinsèque du conteneur .....	579
1.3.3 Pour connaître la relation d'ordre utilisée par un conteneur .....	580
1.3.4 Conséquences du choix de l'ordre d'un conteneur .....	581
1.4 Accès aux éléments .....	581
1.4.1 Accès par l'opérateur [ ] .....	581
1.4.2 Accès par itérateur .....	581
1.4.3 Recherche par la fonction membre find .....	582

1.5 Insertions et suppressions . . . . .	582
1.5.1 Insertions . . . . .	583
1.5.2 Suppressions . . . . .	584
1.6 Gestion mémoire . . . . .	584
1.7 Autres possibilités . . . . .	585
1.8 Exemple . . . . .	585
<b>2 - Le conteneur multimap . . . . .</b>	<b>586</b>
2.1 Présentation générale . . . . .	586
2.2 Exemple . . . . .	587
<b>3 - Le conteneur set . . . . .</b>	<b>589</b>
3.1 Présentation générale . . . . .	589
3.2 Exemple . . . . .	589
3.3 Le conteneur set et l'ensemble mathématique . . . . .	590
<b>4 - Le conteneur multiset . . . . .</b>	<b>590</b>
<b>5 - Conteneurs associatifs et algorithmes . . . . .</b>	<b>591</b>
 <b>Chapitre 27 : Les algorithmes standard . . . . .</b>	 <b>593</b>
<b>1 - Notions générales . . . . .</b>	<b>593</b>
1.1 Algorithmes et itérateurs . . . . .	593
1.2 Les catégories d'itérateurs . . . . .	594
1.2.1 Itérateur en entrée . . . . .	594
1.2.2 Itérateur en sortie . . . . .	594
1.2.3 Hiérarchie des catégories d'itérateurs . . . . .	595
1.3 Algorithmes et séquences . . . . .	595
1.4 Itérateur d'insertion . . . . .	596
1.5 Itérateur de flot . . . . .	598
1.5.1 Itérateur de flot de sortie . . . . .	598
1.5.2 Itérateur de flot d'entrée . . . . .	599
<b>2 - Algorithmes d'initialisation de séquences existantes . . . . .</b>	<b>600</b>
2.1 Copie d'une séquence dans une autre . . . . .	600
2.2 Génération de valeurs par une fonction . . . . .	601
<b>3 - Algorithmes de recherche . . . . .</b>	<b>603</b>
3.1 Algorithmes fondés sur une égalité ou un prédicat unaire . . . . .	604
3.2 Algorithmes de recherche de maximum ou de minimum . . . . .	605
<b>4 - Algorithmes de transformation d'une séquence . . . . .</b>	<b>606</b>
4.1 Remplacement de valeurs . . . . .	606
4.2 Permutations de valeurs . . . . .	607
4.2.1 Rotation . . . . .	607
4.2.2 Génération de permutations . . . . .	607
4.2.3 Permutations aléatoires . . . . .	609
4.3 Partitions . . . . .	610
<b>5 - Algorithmes dits « de suppression » . . . . .</b>	<b>610</b>
<b>6 - Algorithmes de tri . . . . .</b>	<b>612</b>

<b>7 - Algorithmes de recherche et de fusion sur des séquences ordonnées</b> .....	613
7.1 Algorithmes de recherche binaire .....	614
7.2 Algorithmes de fusion .....	614
<b>8 - Algorithmes à caractère numérique</b> .....	615
<b>9 - Algorithmes à caractère ensembliste</b> .....	616
<b>10 - Algorithmes de manipulation de tas</b> .....	618
<b>Chapitre 28 : La classe string</b> .....	621
<b>1 - Généralités</b> .....	622
<b>2 - Construction</b> .....	622
<b>3 - Opérations globales</b> .....	623
<b>4 - Concaténation</b> .....	624
<b>5 - Recherche dans une chaîne</b> .....	624
5.1 Recherche d'une chaîne ou d'un caractère .....	625
5.2 Recherche d'un caractère présent ou absent d'une suite .....	625
<b>6 - Insertions, suppressions et remplacements</b> .....	626
6.1 Insertions .....	626
6.2 Suppressions .....	627
6.3 Remplacements .....	628
<b>7 - Les possibilités de formatage en mémoire</b> .....	629
7.1 La classe ostream .....	629
7.2 La classe istream .....	630
7.2.1 Présentation .....	630
7.2.2 Utilisation pour fiabiliser les lectures au clavier .....	631
<b>Chapitre 29 : Les outils numériques</b> .....	635
<b>1 - La classe complex</b> .....	635
<b>2 - La classe valarray et les classes associées</b> .....	637
2.1 Constructeurs des classes valarray .....	637
2.2 L'opérateur []. .....	638
2.3 Affectation et changement de taille .....	638
2.4 Calcul vectoriel .....	638
2.5 Sélection de valeurs par masque .....	640
2.6 Sections de vecteurs .....	641
2.7 Vecteurs d'indices .....	642
<b>3 - La classe bitset</b> .....	643
<b>Chapitre 30 : Les espaces de noms</b> .....	647
<b>1 - Création d'espaces de noms</b> .....	647
1.1 Exemple de création d'un nouvel espace de noms .....	648
1.2 Exemple avec deux espaces de noms .....	649

1.3 Espace de noms et fichier en-tête .....	650
1.4 Instructions figurant dans un espace de noms .....	650
1.5 Création incrémentale d'espaces de noms .....	651
<b>2 - Les instructions using .....</b>	<b>652</b>
2.1 La déclaration using pour les symboles .....	652
2.1.1 <i>Présentation générale</i> .....	652
2.1.2 <i>Masquage et ambiguïtés</i> .....	654
2.2 La directive using pour les espaces de noms .....	655
<b>3 - Espaces de noms et recherche de fonctions .....</b>	<b>657</b>
<b>4 - Imbrication des espaces de noms .....</b>	<b>659</b>
<b>5 - Transitivité de la directive using .....</b>	<b>660</b>
<b>6 - Les alias .....</b>	<b>660</b>
<b>7 - Les espaces anonymes .....</b>	<b>661</b>
<b>8 - Espaces de noms et déclaration d'amitié .....</b>	<b>661</b>
<b>Chapitre 31 : Le préprocesseur et l'instruction typedef .....</b>	<b>663</b>
1 - La directive <b>#include</b> .....	663
2 - La directive <b>#define</b> .....	664
2.1 Définition de symboles .....	664
2.2 Définition de macros .....	666
3 - La compilation conditionnelle .....	668
3.1 Incorporation liée à l'existence de symboles .....	669
3.2 Incorporation liée à la valeur d'une expression .....	670
4 - La définition de synonymes avec <b>typedef</b> .....	671
4.1 Définition d'un synonyme de int .....	672
4.2 Définition d'un synonyme de int * .....	672
4.3 Définition d'un synonyme de int[3] .....	673
<b>Annexes .....</b>	<b>675</b>
<b>Annexe A : Règles de recherche d'une fonction surdéfinie .....</b>	<b>677</b>
1 - Détermination des fonctions candidates .....	677
2 - Algorithme de recherche d'une fonction à un seul argument .....	678
2.1 Recherche d'une correspondance exacte .....	678
2.2 Promotions numériques .....	679
2.3 Conversions standards .....	679
2.4 Conversions définies par l'utilisateur .....	680
2.5 Fonctions à arguments variables .....	680
2.6 Exception : cas des champs de bits .....	680
3 - Fonctions à plusieurs arguments .....	681
4 - Fonctions membres .....	681

<b>Annexe B : Compléments sur les exceptions</b> .....	683
1 - Les problèmes posés par les objets automatiques .....	683
2 - La technique de gestion de ressources par initialisation .....	684
3 - Le concept de pointeur intelligent : la classe <code>auto_ptr</code> .....	686
<b>Annexe C : Les différentes sortes de fonctions en C++</b> .....	689
<b>Annexe D : Comptage de références</b> .....	691
<b>Annexe E : Les pointeurs sur des membres</b> .....	695
1 - Les pointeurs sur des fonctions membres .....	695
2 - Les pointeurs sur des membres données .....	696
3 - L'héritage et les pointeurs sur des membres .....	697
<b>Annexe F : Les algorithmes standard</b> .....	699
1 - Algorithmes d'initialisation de séquences existantes .....	700
2 - Algorithmes de recherche .....	701
3 - Algorithmes de transformation d'une séquence .....	703
4 - Algorithmes de suppression .....	706
5 - Algorithmes de tri .....	708
6 - Algorithmes de recherche et de fusion sur des séquences ordonnées .....	710
7 - Algorithmes à caractère numérique .....	712
8 - Algorithmes à caractère ensembliste .....	713
9 - Algorithmes de manipulation de tas .....	716
10 - Algorithmes divers .....	717
<b>Annexe G : Les principales fonctions de la bibliothèque C standard</b> ..	719
1 - Entrées-sorties ( <code>cstdio</code> ) .....	720
1.1 Gestion des fichiers .....	720
1.2 Écriture formatée .....	721
1.3 Les codes de format utilisables avec ces trois fonctions .....	722
1.4 Lecture formatée .....	724
1.5 Règles communes à ces fonctions .....	725
1.6 Les codes de format utilisés par ces fonctions .....	725
1.7 Entrées-sorties de caractères .....	727
1.8 Entrées-sorties sans formatage .....	728
1.9 Action sur le pointeur de fichier .....	729
1.10 Gestion des erreurs .....	729

2 - Tests de caractères et conversions majuscules-minuscules (ctype) . . . . .	729
3 - Manipulation de chaînes (cstring) . . . . .	730
4 - Fonctions mathématiques (cmath) . . . . .	732
5 - Utilitaires (cstdlib) . . . . .	733
6 - Macro de mise au point (cassert) . . . . .	734
7 - Gestion des erreurs (cerrno) . . . . .	735
8 - Branchements non locaux (setjmp) . . . . .	735
<b>Annexe H : Les incompatibilités entre C et C++ . . . . .</b>	<b>737</b>
1 - Prototypes . . . . .	737
2 - Fonctions sans arguments . . . . .	737
3 - Fonctions sans valeur de retour . . . . .	738
4 - Le qualificatif const . . . . .	738
5 - Les pointeurs de type void * . . . . .	738
6 - Mots-clés . . . . .	738
7 - Les constantes de type caractère . . . . .	739
8 - Les définitions multiples . . . . .	739
9 - L'instruction goto . . . . .	740
10 - Les énumérations . . . . .	740
11 - Initialisation de tableaux de caractères . . . . .	740
12 - Les noms de fonctions . . . . .	741
<b>Index . . . . .</b>	<b>743</b>