

Best
EYROLLES

CLAUDE DELANNOY

Apprendre

le C++

EYROLLES

Table des matières

Avant-propos	XXIX
1 - Historique de C++	XXIX
2 - Objectif et structure de l'ouvrage	XXIX
3 - L'ouvrage, C, C++ et Java	XXX
Chapitre 1 : Présentation du langage C++	1
1 - Programmation structurée et programmation orientée objet	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
2 - C++ et la programmation structurée	5
3 - C++ et la programmation orientée objet	6
4 - C et C++	8
5 - C++ et la bibliothèque standard	8
Chapitre 2 : Généralités sur le langage C++	11
1 - Présentation par l'exemple de quelques instructions du langage C++	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
2 - Quelques règles d'écriture	18
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
3 - Création d'un programme en C++	21
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
 Chapitre 3 : Les types de base de C++	 25
1 - La notion de type	25
2 - Les types entiers	26
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
3 - Les types flottants	29
3.1 Les différents types et leur représentation en mémoire	29
3.2 Notation des constantes flottantes	30
4 - Les types caractères	31
4.1 La notion de caractère en langage C++	31
4.2 Notation des constantes caractères	31
5 - Initialisation et constantes	33
6 - Le type bool	34
 Chapitre 4 : Opérateurs et expressions	 35
1 - Originalité des notions d'opérateur et d'expression en C++	35
2 - Les opérateurs arithmétiques en C++	37
2.1 Présentation des opérateurs	37
2.2 Les priorités relatives des opérateurs	38
2.3 Comportement des opérateurs en cas d'exception	38

3 - Les conversions implicites pouvant intervenir dans un calcul d'expression	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
4 - Les opérateurs relationnels	45
5 - Les opérateurs logiques	47
5.1 Rôle	47
5.2 Court-circuit dans l'évaluation de <i>&&</i> et <i> </i>	48
6 - L'opérateur d'affectation ordinaire	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
7 - Opérateurs d'incrément et de décrémentation	50
7.1 Leur rôle	50
7.2 Leurs priorités	52
7.3 Leur intérêt	52
8 - Les opérateurs d'affectation élargie	52
9 - Les conversions forcées par une affectation	53
9.1 Cas usuels	53
9.2 Prise en compte d'un attribut de signe	54
10 - L'opérateur de cast	54
11 - L'opérateur conditionnel	55
12 - L'opérateur séquentiel	56
13 - L'opérateur <i>sizeof</i>	58
14 - Les opérateurs de manipulation de bits	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
15 - Récapitulatif des priorités de tous les opérateurs	61
Chapitre 5 : Les entrées-sorties conversationnelles de C++	63
1 - Affichage à l'écran	63
1.1 Exemple 1	64
1.2 Exemple 2	64
1.3 Les possibilités d'écriture sur <i>cout</i>	65

2 - Lecture au clavier	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 <i>Manque de synchronisme entre clavier et écran</i>	69
2.6.2 <i>Blocage de la lecture</i>	70
2.6.3 <i>Boucle infinie sur un caractère invalide</i>	70
 Chapitre 6 : Les instructions de contrôle	 73
1 - Les blocs d'instructions	74
1.1 Blocs d'instructions	74
1.2 Déclarations dans un bloc	75
2 - L'instruction if	75
2.1 Syntaxe de l'instruction if	76
2.2 Exemples	76
2.3 Imbrication des instructions if	77
3 - L'instruction switch	79
3.1 Exemples d'introduction de l'instruction switch	79
3.2 Syntaxe de l'instruction switch	82
4 - L'instruction do... while	84
4.1 Exemple d'introduction de l'instruction do... while	84
4.2 Syntaxe de l'instruction do... while	85
5 - L'instruction while	86
5.1 Exemple d'introduction de l'instruction while	86
5.2 Syntaxe de l'instruction while	87
6 - L'instruction for	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
7 - Les instructions de branchement inconditionnel : break, continue et goto	93
7.1 L'instruction break	93
7.2 L'instruction continue	94
7.3 L'instruction goto	95
 Chapitre 7 : Les fonctions	 97
1 - Exemple de définition et d'utilisation d'une fonction	98
2 - Quelques règles	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

3 - Les fonctions et leurs déclarations	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
4 - Transmission des arguments par valeur	104
5 - Transmission par référence	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
6 - Les variables globales	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
7 - Les variables locales	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
8 - Initialisation des variables	116
8.1 Les variables de classe statique	116
8.2 Les variables de classe automatique	116
9 - Les arguments par défaut	117
9.1 Exemples	117
9.2 Les propriétés des arguments par défaut	118
10 - Surdéfinition de fonctions	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
11 - Les arguments variables en nombre	124
11.1 Premier exemple	125
11.2 Second exemple	126
12 - La conséquence de la compilation séparée	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
13	- Compléments sur les références	132
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
14	- La spécification inline	136
 Chapitre 8 : Les tableaux et les pointeurs		139
1	- Les tableaux à un indice	140
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
2	- Les tableaux à plusieurs indices	143
2.1	Leur déclaration	143
2.2	Arrangement en mémoire des tableaux à plusieurs indices	143
3	- Initialisation des tableaux	144
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
4	- Notion de pointeur – Les opérateurs * et &	146
4.1	Introduction	146
4.2	Quelques exemples	147
4.3	Incrémentation de pointeurs	148
5	- Comment simuler une transmission par adresse avec un pointeur	149
6	- Un nom de tableau est un pointeur constant	151
6.1	Cas des tableaux à un indice	151
6.2	Cas des tableaux à plusieurs indices	152
7	- Les opérations réalisables sur des pointeurs	153
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
8 - La gestion dynamique : les opérateurs new et delete	157
8.1 L'opérateur new	157
8.2 L'opérateur delete	159
8.3 Exemple	159
9 - Pointeurs et surdéfinition de fonctions	161
10 - Les tableaux transmis en argument	162
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
11 - Utilisation de pointeurs sur des fonctions	166
11.1 Paramétrage d'appel de fonctions	166
11.2 Fonctions transmises en argument	167
 Chapitre 9 : Les chaînes de style C	 169
1 - Représentation des chaînes	170
1.1 La convention adoptée	170
1.2 Cas des chaînes constantes	170
2 - Lecture et écriture de chaînes de style C	172
3 - Initialisation de tableaux par des chaînes	173
3.1 Initialisation de tableaux de caractères	173
3.2 Initialisation de tableaux de pointeurs sur des chaînes	174
4 - Les arguments transmis à la fonction main	175
4.1 Comment passer des arguments à un programme	175
4.2 Comment récupérer ces arguments dans la fonction main	176
5 - Généralités sur les fonctions portant sur des chaînes de style C	177
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
6 - Les fonctions de concaténation de chaînes	178
6.1 La fonction strcat	178
6.2 La fonction strncat	179
7 - Les fonctions de comparaison de chaînes	180
8 - Les fonctions de copie de chaînes	181
9 - Les fonctions de recherche dans une chaîne	182
10 - Quelques précautions à prendre avec les chaînes de style C	182
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

Chapitre 10 : Les types structure, union et énumération	185
1 - Déclaration d'une structure	186
2 - Utilisation d'une structure	187
2.1 Utilisation des champs d'une structure	187
2.2 Utilisation globale d'une structure	188
2.3 Initialisation de structures	188
3 - Imbrication de structures	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
4 - À propos de la portée du type de structure	192
5 - Transmission d'une structure en argument d'une fonction	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
6 - Transmission d'une structure en valeur de retour d'une fonction	196
7 - Les champs de bits	197
8 - Les unions	198
9 - Les énumérations	200
9.1 Exemples introductifs	200
9.2 Propriétés du type énumération	201
Chapitre 11 : Classes et objets	203
1 - Les structures généralisées	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
2 - Notion de classe	208
3 - Affectation d'objets	212
4 - Notions de constructeur et de destructeur	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
5 - Les membres données statiques	221
5.1 Le qualificatif static pour un membre donnée	221
5.2 Initialisation des membres données statiques	222
5.3 Exemple	223

6 - Exploitation d'une classe	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
7 - Les classes en général	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
Chapitre 12 : Les propriétés des fonctions membres	231
1 - Surdéfinition des fonctions membres	231
2 - Arguments par défaut	234
3 - Les fonctions membres en ligne	235
4 - Cas des objets transmis en argument d'une fonction membre	237
5 - Mode de transmission des objets en argument	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
6 - Lorsqu'une fonction renvoie un objet	242
7 - Autoréférence : le mot clé this	243
8 - Les fonctions membres statiques	244
9 - Les fonctions membres constantes	246
9.1 Définition d'une fonction membre constante	246
9.2 Propriétés d'une fonction membre constante	247
10 - Les membres mutables	249
Chapitre 13 : Construction, destruction et initialisation des objets	251
1 - Les objets automatiques et statiques	252
1.1 Durée de vie	252
1.2 Appel des constructeurs et des destructeurs	253
1.3 Exemple	253
2 - Les objets dynamiques	255
2.1 Cas d'une classe sans constructeur	255
2.2 Cas d'une classe avec constructeur	256
2.3 Exemple	257

3 - Le constructeur de copie	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
4 - Initialisation d'un objet lors de sa déclaration	266
5 - Objets membres	268
5.1 Introduction	268
5.2 Mise en œuvre des constructeurs et des destructeurs	269
5.3 Le constructeur de copie	271
6 - Initialisation de membres dans l'en-tête d'un constructeur	272
7 - Les tableaux d'objets	273
7.1 Notations	273
7.2 Constructeurs et initialiseurs	274
7.3 Cas des tableaux dynamiques d'objets	275
8 - Les objets temporaires	276
Chapitre 14 : Les fonctions amies	279
1 - Exemple de fonction indépendante amie d'une classe	280
2 - Les différentes situations d'amitié	282
2.1 Fonction membre d'une classe, amie d'une autre classe	283
2.2 Fonction amie de plusieurs classes	284
2.3 Toutes les fonctions d'une classe amies d'une autre classe	285
3 - Exemple	285
3.1 Fonction amie indépendante	286
3.2 Fonction amie, membre d'une classe	287
4 - Exploitation de classes disposant de fonctions amies	288
Chapitre 15 : La surdéfinition d'opérateurs	291
1 - Le mécanisme de la surdéfinition d'opérateurs	292
1.1 Surdéfinition d'opérateur avec une fonction amie	293
1.2 Surdéfinition d'opérateur avec une fonction membre	294
1.3 Opérateurs et transmission par référence	296
2 - La surdéfinition d'opérateurs en général	297
2.1 Se limiter aux opérateurs existants	297
2.2 Se placer dans un contexte de classe	299
2.3 Éviter les hypothèses sur le rôle d'un opérateur	299
2.4 Cas des opérateurs ++ et --	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
3 - Surdéfinition de l'opérateur =	303
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
4 - La forme canonique d'une classe	310
4.1 Cas général	310
5 - Exemple de surdéfinition de l'opérateur []	311
6 - Surdéfinition de l'opérateur ()	314
7 - Surdéfinition des opérateurs new et delete	314
7.1 Surdéfinition de new et delete pour une classe donnée	315
7.2 Exemple	315
7.3 D'une manière générale	317
Chapitre 16 : Les conversions de type définies par l'utilisateur	319
1 - Les différentes sortes de conversions définies par l'utilisateur	320
2 - L'opérateur de cast pour la conversion type classe → type de base	322
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
3 - Le constructeur pour la conversion type de base → type classe	329
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
4 - Les conversions d'un type classe en un autre type classe	336
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
5 - Quelques conseils	341
Chapitre 17 : Les patrons de fonctions	343
1 - Exemple de création et d'utilisation d'un patron de fonctions	344
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 <i>classe</i>	347
1.4 Contraintes d'utilisation d'un patron	348
2 - Les paramètres de type d'un patron de fonctions	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
3 - Les paramètres expressions d'un patron de fonctions	353
4 - Surdéfinition de patrons	354
4.1 Exemples ne comportant que des paramètres de type	354
4.2 Exemples comportant des paramètres expressions	357
5 - Spécialisation de fonctions de patron	358
5.1 Généralités	358
5.2 Les spécialisations partielles	358
6 - Algorithme d'instanciation d'une fonction patron	359
Chapitre 18 : Les patrons de classes	363
1 - Exemple de création et d'utilisation d'un patron de classes	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
2 - Les paramètres de type d'un patron de classes	369
2.1 Les paramètres de type dans la création d'un patron de classes	369
2.2 Instanciation d'une classe patron	369
3 - Les paramètres expressions d'un patron de classes	370
3.1 Exemple	371
3.2 Les propriétés des paramètres expressions	372
4 - Spécialisation d'un patron de classes	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
5 - Paramètres par défaut	376
6 - Patrons de fonctions membres	376
7 - Identité de classes patrons	376
8 - Classes patrons et déclarations d'amitié	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
9 - Exemple de classe tableau à deux indices	379

Chapitre 19 : L'héritage simple	383
1 - La notion d'héritage	384
2 - Utilisation des membres de la classe de base dans une classe dérivée	386
3 - Redéfinition des membres d'une classe dérivée	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
4 - Appel des constructeurs et des destructeurs	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
5 - Contrôle des accès	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 <i>Rappels concernant la dérivation publique</i>	399
5.4.2 <i>Dérivation privée</i>	400
5.4.3 <i>Les possibilités de dérivation protégée</i>	401
5.5 Récapitulation	402
6 - Compatibilité entre classe de base et classe dérivée	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
7 - Le constructeur de copie et l'héritage	409
7.1 La classe dérivée ne définit pas de constructeur de copie	409
7.2 La classe dérivée définit un constructeur de copie	410
8 - L'opérateur d'affectation et l'héritage	412
8.1 La classe dérivée ne surdéfinit pas l'opérateur =	412
8.2 La classe dérivée surdéfinit l'opérateur =	412
9 - Héritage et forme canonique d'une classe	415
10 - L'héritage et ses limites	417
10.1 La situation d'héritage	417
10.1.1 <i>Le type du résultat de l'appel</i>	418
10.1.2 <i>Le type des arguments de f</i>	418
10.2 Exemples	418
10.2.1 <i>Héritage dans pointcol d'un opérateur + défini dans point</i>	419
10.2.2 <i>Héritage dans pointcol de la fonction coincide de point</i>	419

11 - Exemple de classe dérivée	420
12 - Patrons de classes et héritage	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
13 - L'héritage en pratique	426
13.1 Dérivations successives	426
13.2 Différentes utilisations de l'héritage	428
13.3 Exploitation d'une classe dérivée	428
Chapitre 20 : L'héritage multiple	431
1 - Mise en œuvre de l'héritage multiple	432
2 - Pour régler les éventuels conflits : les classes virtuelles	436
3 - Appels des constructeurs et des destructeurs : cas des classes virtuelles	437
4 - Exemple d'utilisation de l'héritage multiple et de la dérivation virtuelle	440
Chapitre 21 : Les fonctions virtuelles et le polymorphisme	443
1 - Rappel d'une situation où le typage dynamique est nécessaire	444
2 - Le mécanisme des fonctions virtuelles	444
3 - Autre situation où la ligature dynamique est indispensable	446
4 - Les propriétés des fonctions virtuelles	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 Seule une fonction membre peut être virtuelle	452
4.6.2 Un constructeur ne peut pas être virtuel	452
4.6.3 Un destructeur peut être virtuel	453
4.6.4 Cas particulier de l'opérateur d'affectation	454
5 - Les fonctions virtuelles pures pour la création de classes abstraites	455
6 - Exemple d'utilisation de fonctions virtuelles : liste hétérogène	457
7 - Le mécanisme d'identification dynamique des objets	461
8 - Identification de type à l'exécution	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
9 - Les cast dynamiques	466

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

6 - Connexion d'un flot à un fichier	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
7 - Les anciennes possibilités de formatage en mémoire	505
7.1 La classe <code>ostrstream</code>	506
7.2 La classe <code>istrstream</code>	507

Chapitre 23 : La gestion des exceptions

1 - Premier exemple d'exception	510
1.1 Comment lancer une exception : l'instruction <code>throw</code>	511
1.2 Utilisation d'un gestionnaire d'exception	511
1.3 Récapitulatif	512
2 - Second exemple	514
3 - Le mécanisme de gestion des exceptions	516
3.1 Poursuite de l'exécution du programme	516
3.2 Prise en compte des sorties de blocs	518
4 - Choix du gestionnaire	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
5 - Spécification d'interface : la fonction <code>unexpected</code>	523
6 - Les exceptions standard	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 <code>new (nothrow)</code>	527
6.4.2 Gestion des débordements de mémoire avec <code>set_new_handler</code>	528
6.5 Création d'exceptions dérivées de la classe <code>exception</code>	529
6.5.1 Exemple 1	530
6.5.2 Exemple 2	530

Chapitre 24 : Généralités sur la bibliothèque standard

1 - Notions de conteneur, d'itérateur et d'algorithme	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
2 - Les différentes sortes de conteneurs	538
2.1 Conteneurs et structures de données classiques	538
2.2 Les différentes catégories de conteneurs	539
3 - Les conteneurs dont les éléments sont des objets	539
3.1 Construction, copie et affectation	540
3.2 Autres opérations	541
4 - Efficacité des opérations sur des conteneurs	541
5 - Fonctions, prédicats et classes fonctions	542
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
6 - Conteneurs, algorithmes et relation d'ordre	545
6.1 Introduction	545
6.2 Propriétés à respecter	545
7 - Les générateurs d'opérateurs	546
Chapitre 25 : Les conteneurs séquentiels	549
1 - Fonctionnalités communes aux conteneurs vector, list et deque	550
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

2 - Le conteneur vector	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
3 - Le conteneur deque	562
3.1 Présentation générale	562
3.2 Exemple	563
4 - Le conteneur list	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
5 - Les adaptateurs de conteneur : queue, stack et priority_queue	570
5.1 L'adaptateur stack	570
5.2 L'adaptateur queue	571
5.3 L'adaptateur priority_queue	572
Chapitre 26 : Les conteneurs associatifs	575
1 - Le conteneur map	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
2 - Le conteneur multimap	586
2.1 Présentation générale	586
2.2 Exemple	587
3 - Le conteneur set	589
3.1 Présentation générale	589
3.2 Exemple	589
3.3 Le conteneur set et l'ensemble mathématique	590
4 - Le conteneur multiset	590
5 - Conteneurs associatifs et algorithmes	591
Chapitre 27 : Les algorithmes standard	593
1 - Notions générales	593
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
2 - Algorithmes d'initialisation de séquences existantes	600
2.1 Copie d'une séquence dans une autre	600
2.2 Génération de valeurs par une fonction	601
3 - Algorithmes de recherche	603
3.1 Algorithmes fondés sur une égalité ou un prédicat unaire	604
3.2 Algorithmes de recherche de maximum ou de minimum	605
4 - Algorithmes de transformation d'une séquence	606
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
5 - Algorithmes dits « de suppression »	610
6 - Algorithmes de tri	612

7 - Algorithmes de recherche et de fusion sur des séquences ordonnées	613
7.1 Algorithmes de recherche binaire	614
7.2 Algorithmes de fusion	614
8 - Algorithmes à caractère numérique	615
9 - Algorithmes à caractère ensembliste	616
10 - Algorithmes de manipulation de tas	618
Chapitre 28 : La classe string	621
1 - Généralités	622
2 - Construction	622
3 - Opérations globales	623
4 - Concaténation	624
5 - Recherche dans une chaîne	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
6 - Insertions, suppressions et remplacements	626
6.1 Insertions	626
6.2 Suppressions	627
6.3 Remplacements	628
7 - Les possibilités de formatage en mémoire	629
7.1 La classe <code>ostream</code>	629
7.2 La classe <code>istream</code>	630
7.2.1 <i>Présentation</i>	630
7.2.2 <i>Utilisation pour fiabiliser les lectures au clavier</i>	631
Chapitre 29 : Les outils numériques	635
1 - La classe <code>complex</code>	635
2 - La classe <code>valarray</code> et les classes associées	637
2.1 Constructeurs des classes <code>valarray</code>	637
2.2 L'opérateur <code>[]</code>	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
3 - La classe <code>bitset</code>	643
Chapitre 30 : Les espaces de noms	647
1 - Création d'espaces de noms	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
2 - Les instructions using	652
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
3 - Espaces de noms et recherche de fonctions	657
4 - Imbrication des espaces de noms	659
5 - Transitivité de la directive using	660
6 - Les alias	660
7 - Les espaces anonymes	661
8 - Espaces de noms et déclaration d'amitié	661
Chapitre 31 : Le préprocesseur et l'instruction typedef	663
1 - La directive #include	663
2 - La directive #define	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 typedef	671
4.1 Définition d'un synonyme de <code>int</code>	672
4.2 Définition d'un synonyme de <code>int *</code>	672
4.3 Définition d'un synonyme de <code>int[3]</code>	673
Annexes	675
Annexe A : Règles de recherche d'une fonction surdéfinie	677
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

Annexe B : Compléments sur les exceptions	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
Annexe C : Les différentes sortes de fonctions en C++	689
Annexe D : Comptage de références	691
Annexe E : Les pointeurs sur des membres	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
Annexe F : Les algorithmes standard	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
Annexe G : Les principales fonctions de la bibliothèque C standard	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