Table des matières

I Un peu de Conception **1**

1 Génie Logiciel **3**
1.1 Le quoi et le comment 3
1.2 Cycle de vie 4
1.3 Spécifier 6
1.3.1 Approche procédurale 7
1.3.2 Des types abstraits à l'objet 8
1.4 Structurer 11
1.5 Encore le cycle de vie 16
1.6 Une autre structuration : l'héritage 18
1.7 Méthodologie objet 22

2 Programmation objet **23**
2.1 Introduction 23
2.2 Les types 25
2.2.1 Instanciation 25
2.2.2 Attributs 26
2.2.3 Privé et public 26
2.2.4 Sélection et itération 27
2.3 Généricité 28
2.3.1 Surcharge d'opérateur 29
2.3.2 Retour sur l'héritage, polymorphisme 30
2.3.3 Types paramètres 33

3 Objectif « objet » **35**

II De C à C++ **37**

4 Syntaxe et codage **41**
4.1 Commentaires 41

4.2 Déclaration des variables 42
4.2.1 Rappels sur la visibilité 42
4.2.2 Déclaration « à la volée » 43
4.2.3 Les compteurs de boucle 43
4.3 Types structurés 45
4.3.1 Déclaration des structures 45
4.3.2 Déclaration des unions 48
4.3.3 Encore les structures 50
4.3.4 Les types énumérés 52

5 Autres types, classes et références 53
5.1 Nouveau, un type booléen 53
5.2 Les classes 54
5.3 Rappels sur les pointeurs 56
5.4 Références 59
5.4.1 Utiliser les références 61
5.4.2 Référence égal *lvalue* 63

6 Opérateurs et Fonctions 65
6.1 Les opérateurs 65
6.1.1 Allocation dynamique 65
6.1.2 Conversion explicite de type 66
6.1.3 L'opérateur :: de résolution de portée 67
6.2 Les fonctions 68
6.2.1 Arguments par défaut 68
6.2.2 Fonctions ouvertes 70
6.3 Rappels sur le passage des arguments 72
6.3.1 Arguments formels, arguments réels 72
6.3.2 Les modes de passage 72
6.4 Passage par valeur en C et C++ 74
6.5 Passage par référence en C++ 75
6.6 Retourner une référence en C++ 78

7 Forme des programmes 85
7.1 La fonction `main` 85
7.2 Espaces de noms 86
7.2.1 Accès, l'opérateur `::` 89
7.2.2 Espace sans nom 92
7.2.3 Utilisation d'alias 96
7.3 Fichiers à inclure, l'espace prédéfini `std` 98
7.4 Appel de fonctions non C++ 100

8 Entrées-sorties et autres 103
8.1 Entrées-sorties en C++ . . . 103
8.2 Autres points . . . 105
8.2.1 Chaînes de caractères . . . 105
8.2.2 D'autres ressources . . . 106

9 Généricité 107
9.1 Surcharge des fonctions . . . 107
9.2 Rappels sur les opérateurs . . . 109
9.3 Surcharge des opérateurs . . . 110
9.4 Patron de programme . . . 112
9.4.1 Patron avec constante . . . 116
9.4.2 Spécialisation . . . 118
9.4.3 Précompiler les patrons ? . . . 120
9.4.4 Un exemple, les nombres complexes . . . 120

10 Où sont les objets ? 121

III Construire une classe 123

11 Une application test 125
11.1 Les récurrences linéaires . . . 125
11.2 Des algorithmes « vectoriels » . . . 126
11.2.1 Une méthode itérative . . . 127
11.2.2 Un algorithme récursif . . . 128
11.3 Cahier des charges . . . 131

12 La classe vecteur (1) 135
12.1 Déclaration, fichier `"vecteur.hpp"` . . . 136
12.1.1 Attributs : privé et public . . . 137
12.2 Implanter les données . . . 138
12.3 Les méthodes . . . 140
12.4 Accès aux données et *const-correctness* . . . 140
12.4.1 Accès en lecture . . . 140
12.4.2 Accès en lecture et en écriture . . . 142
12.5 Les fichiers de l'application . . . 145
12.5.1 Lisibilité de la partie publique . . . 145
12.5.2 Réalisation séparée `"vecteur.cpp"` . . . 146
12.5.3 Un fichier `makefile` . . . 148
12.6 Gestion des objets . . . 150

12.7 Constructeurs et destructeurs . . . 151
12.7.1 Que se passe-t-il ? un mode « trace » . . . 153
12.7.2 D'autres constructeurs . . . 156
12.7.3 Constructeur par recopie . . . 158
12.8 Appel explicite d'un constructeur . . . 162
12.9 Constructeurs et tableaux . . . 163
12.10 Nommer l'instance courante : le pointeur `this` . . . 164
12.11 Redimensionner, ajuster . . . 166
12.12 Surcharger un premier opérateur : l'affectation . . . 168
12.13 Comparer des objets . . . 171
12.14 Objets et fonction . . . 174
12.14.1 Passage par valeur . . . 174
12.14.2 Valeur de retour . . . 175
12.14.3 Des optimisations possibles . . . 177
12.15 Gestion C++ *versus* gestion C . . . 179
12.16 Tout tester . . . 181
12.17 Premières conclusions . . . 183
12.17.1 Code de la première partie . . . 185

13 La classe vecteur (2) 191
13.1 L'addition . . . 191
13.1.1 Un opérateur apparenté += . . . 192
13.1.2 Réalisation de + (1ers essais) . . . 193
13.1.3 Retourner un constructeur . . . 195
13.1.4 Réalisation de + (nouvel essai) . . . 196
13.1.5 Bilan . . . 198
13.2 Multiplier, soustraire, diviser et l'homothétie . . . 198
13.3 Translations, pair, impair et alterner . . . 201
13.4 Quelques conclusions . . . 203

14 La classe vecteur (3 et ++) 209
14.1 Fonctions amies . . . 209
14.1.1 Classes amies . . . 212
14.2 Lire et écrire des vecteurs . . . 214
14.2.1 En utilisant une fonction amie . . . 216
14.2.2 Sans fonction amie . . . 218
14.2.3 Réalisation pour vecteur . . . 220
14.3 Attributs de classe . . . 223
14.3.1 Données statiques . . . 224
14.3.2 Méthodes statiques . . . 224
14.3.3 Utilisation . . . 226

14.3.4 Compter les instances de vecteur 226
14.4 Vous avez dit `const` ? 231
14.4.1 Espionner les objets 233
14.5 Pour en finir (?) avec les vecteurs 236

15 L'application **237**
15.1 Une bibliothèque « solveur » 237
15.1.1 Classe solveur et prototypes 238
15.1.2 Codage 240
15.1.3 Quelques améliorations 245
15.2 Version finale 250

16 Derniers points **253**
16.1 Initialisation des attributs constants 253
16.2 Attributs références 254
16.3 Incrémentation, décrémentation 258
16.4 Operateurs unaires 262
16.5 Opérateur de fonction 263
16.6 Conversions de type 265
16.7 Références ou pointeurs ? 267
16.8 Pointeurs sur des attributs 267
16.9 Quels opérateurs surcharger 271
16.10 Méthode ou fonction externe ? 271

17 Programmer « objet » **273**

IV Patrons, héritage, exceptions et ++ **275**

18 Les patrons de classe **277**
18.1 Réutiliser 277
18.2 Spécialisation 285
18.3 Des bibliothèques de patrons 288

19 Héritage (1) **289**
19.1 Réutiliser encore 289
19.2 Déclaration 290
19.2.1 Polymorphisme 291
19.2.2 Héritages à gogo 292
19.2.3 Attributs privés, publics, protégés et... amitiés 294
19.2.4 Héritage protégé, privé 296

19.2.5 Constructeurs, destructeurs 298
19.2.6 Le constructeur par recopie 301
19.3 Autres méthodes 302
19.3.1 Méthodes virtuelles 302
19.3.2 Covariance, redéfinition *versus* masquage 308
19.3.3 Le destructeur virtuel 310
19.3.4 Un constructeur virtuel ? créer, cloner 313
19.3.5 L'affectation 320
19.3.6 Expressions danger ! 322
19.3.7 Fonction « externe virtuelle » 324
19.3.8 Bilan et bonnes pratiques 326
19.4 Héritage, un exemple complet 328
19.4.1 Résumé du cahier des charges 329
19.4.2 Le type point 329
19.4.3 Le types segment 330
19.4.4 Le type polygone 331
19.4.5 Une bibliothèque polygone 333
19.4.6 Des triangles et des quadrangles 335
19.4.7 Une application 338
19.5 Code complet 342
19.6 L'interface non virtuelle (`NVI`) 359
19.7 Classes abstraites 363
19.7.1 Décrire des archétypes 363
19.7.2 Méthodes virtuelles pures 364
19.7.3 Manipuler des archétypes 366
19.7.4 Dériver des types utiles 367
19.8 Créer des interfaces 376

20 Héritage (2) 377
20.1 L'héritage multiple 377
20.2 Lever les ambiguïtés 378
20.3 Le losange 386
20.3.1 Le losange, héritage « normal » 389
20.3.2 Le losange, héritage « virtuel » 393
20.3.3 Héritage virtuel et méthodes virtuelles 397
20.3.4 Conclusions sur le losange 398
20.4 Pointeurs, des soucis prévisibles 398
20.5 Héritage multiple, un exemple complet 400
20.6 Héritage multiple : oui ou non ? 408

21 `RTTI` et *cast* en tous genres **409**
21.1 Un peu de réflexivité . . . 409
21.2 Changement de type (*cast*) . . . 412
21.2.1 `const_cast` . . . 412
21.2.2 `static_cast` . . . 413
21.2.3 `reinterpret_cast` . . . 415
21.2.4 `dynamic_cast` . . . 415
21.3 `X` est-il de type `Obj` ou d'un type dérivé de `Obj` ? . . . 417

22 Exceptions, gérer les erreurs **421**
22.1 Motivations et état de l'art . . . 421
22.1.1 Cas des langages procéduraux (`Fortran`, `C`, ...) . . . 422
22.1.2 Que faire en C++ . . . 423
22.2 Exceptions en C++ . . . 425
22.2.1 Les erreurs sont des objets . . . 425
22.2.2 Utiliser les objets exception . . . 428
22.2.3 Constructeurs et destructeurs . . . 434
22.2.4 Liste d'exceptions . . . 434
22.2.5 Gérer l'arrêt du programme . . . 435
22.2.6 Utiliser les exceptions . . . 439
22.3 Construire ses propres objets exception . . . 439
22.3.1 Utilisation dans une application . . . 442
22.3.2 Un exemple de remontée d'erreur . . . 445
22.4 Erreurs et objets . . . 448
22.4.1 Organiser les classes exception . . . 453
22.5 Intégrer la gestion des exceptions . . . 456

23 Compléments++ **457**
23.1 Les objets fonction . . . 457
23.2 Itérateurs . . . 461
23.2.1 Motivations : sélection et itération . . . 461
23.2.2 Un objet itérateur . . . 462
23.2.3 Itérateur, un exemple complet . . . 462
23.2.4 Ne pas oublier les itérateurs . . . 470
23.3 Gestion mémoire et pointeurs « améliorés » . . . 470
23.3.1 Les pointeurs automatiques (`auto_ptr`) . . . 471
23.3.2 Les pointeurs « intelligents » . . . 474
23.3.3 La « `RAII` » : une technique de programmation . . . 474
23.4 Objets persistants . . . 476
23.5 Pointeurs sur les attributs (revisités) . . . 478
23.5.1 Encapsuler l'appel d'une méthode . . . 478

23.5.2 Changement d'interface 482
23.5.3 Est-ce utile ? 484
23.6 Pour aider les utilisateurs 485
23.6.1 Gérer les fichiers à inclure 485
23.6.2 Générer la documentation 487
23.7 Derniers mots sur la bibliothèque C++ et la STL 490
23.7.1 Un exemple d'utilisation de la STL 491

24 Bilan final, objets en C++ **497**

Annexes

A Précédence et associativité **503**

B Liste des mots réservés **507**

C Les fichiers à inclure **509**

D « Mini-FAQ » en 50 questions **511**

E « Quizz code » en 30 questions **525**

Table des programmes **533**

Bibliographie **545**

Mots-clefs et identificateurs **547**

Index général **549**