Стефан К. Дьюхэрст

Скользкие места C++

Как избежать проблем при проектировании и компиляции ваших программ

C++ Gotchas

Avoiding Common Problems in Coding and Design

Stephen C. Dewhurst

Addison-Wesley

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Скользкие места C++

Как избежать проблем при проектировании и компиляции ваших программ

Стефан К. Дьюхэрст

ДМК Пресс
издательство
Москва, 2006

УДК 004.4
ББК 32.973.26-018.2
Д92

Стефан К. Дьюхэрст
Д92 Скользкие места C++. Как избежать проблем при проектировании и компиляции ваших программ. – М.: ДМК Пресс, 2006. – 264 с.: ил.

ISBN 5-94074-083-9

Вы держите в руках руководство по тому, как не допускать и исправлять 99% типичных, разрушительных и просто любопытных ошибок при проектировании и реализации программ на языке C++. Эту книгу можно рассматривать также, как взгляд посвященного на нетривиальные особенности и приемы программирования на C++.

Обсуждаются как наиболее распространенные «ляпы», имеющиеся почти в любой программе на C++, так и сложные ошибки в использовании синтаксиса, препроцессора, преобразований типов, инициализации, управления памятью и ресурсами, полиморфизма, а также при проектировании классов и иерархий. Все ошибки и их последствия обсуждаются в контексте. Подробно описываются способы разрешения указанных проблем.

Автор знакомит читателей с идиомами и паттернами проектирования, с помощью которых можно решать типовые задачи. Читатель также узнает много нового о плохо понимаемых возможностях C++, которые применяются в продвинутых программах и проектах. На сайте http://www.semantics.org можно найти полный код примеров из книги.

В книге рассказывается, как миновать наиболее серьезные опасности, подстерегающие программиста на C++. Программисты найдут в ней практические рекомендации, которые позволят им стать настоящими экспертами.

Издание предназначено для всех программистов, желающих научиться писать правильные и корректно работающие программы на языке C++.

УДК 004.4
ББК 32.973.26-018.2

Original English language edition published by Pearson Education, Inc. Copyright © 2003 by Syngress Publishing, Inc. All rights reserved.

Все права защищены. Любая часть этой книги не может быть воспроизведена в какой бы то ни было форме и какими бы то ни было средствами без письменного разрешения владельцев авторских прав.

Материал, изложенный в данной книге, многократно проверен. Но поскольку вероятность технических ошибок все равно существует, издательство не может гарантировать абсолютную точность и правильность приводимых сведений. В связи с этим издательство не несет ответственности за возможные ошибки, связанные с использованием книги.

ISBN 0-321-12518-5 (англ.)
ISBN 5-94074-083-9

Copyright © 2003 by Pearson Education, Inc.
© Перевод на русский язык, оформление, издание Издательство ДМК-пресс, 2006

Содержание

Предисловие 9
Благодарности 13
Глава 1. Основы
Совет 1. Избыточное комментирование 15
Совет 2. Магические числа 17
Совет 3. Глобальные переменные 19
Совет 4. Отличайте перегрузку от инициализации аргументов по умолчанию 21
Совет 5. О неправильной интерпретации ссылок 22
Совет 6. О неправильной интерпретации const 25
Совет 7. Не забывайте о тонкостях базового языка 26
Совет 8. Отличайте доступность от видимости 29
Совет 9. О неграмотности 33
Лексика 33
Нулевые указатели 34
Акронимы 35
Совет 10. Не игнорируйте идиомы 35
Совет 11. Не мудрствуйте лукаво 38
Совет 12. Не ведите себя как дети 40
Глава 2. Синтаксис
Совет 13. Не путайте массивы с инициализаторами 42
Совет 14. Неопределенный порядок вычислений 43
Порядок вычисления аргументов функции 43
Порядок вычисления подвыражений 44
Порядок вычисления размещающего new 45
Операторы, которые фиксируют порядок вычислений 46
Некорректная перегрузка операторов 47
Совет 15. Помните о предшествовании 47
Приоритеты и ассоциативность 47
Проблемы, связанные с приоритетом операторов 48
Проблемы, связанные с ассоциативностью 49
Совет 16. Подводные камни в предложении for 50
Совет 17. Принцип «максимального куска» 53

Совет 18. О порядке следования спецификаторов в объявлениях 54
Совет 19. Функция или объект? 55
Совет 20. Перестановка квалификаторов типа 55
Совет 21. Автоинициализация 56
Совет 22. Статические и внешние типы 58
Совет 23. Аномалия при поиске операторной функции 58
Совет 24. Тонкости оператора -> 60

Глава 3. Препроцессор

Совет 25. Определение литералов с помощью #define 62
Совет 26. Определение псевдофункций с помощью #define 64
Совет 27. Не увлекайтесь использованием директивы #if 66
Использование директивы #if для отладки 66
Использование #if для переносимости 68
А как насчет классов? 69
Практика – критерий истины 70
Совет 28. Побочные эффекты в утверждениях 70

Глава 4. Преобразования

Совет 29. Преобразование посредством void * 73
Совет 30. Срезка 76
Совет 31. Преобразование в указатель на константу 78
Совет 32. Преобразование в указатель на указатель на константу 79
Совет 33. Преобразование указателя на указатель на базовый класс 82
Совет 34. Проблемы с указателем на многомерный массив 82
Совет 35. Бесконтрольное понижающее приведение 84
Совет 36. Неправильное использование операторов преобразования .. 84
Совет 37. Непреднамеренное преобразование с помощью конструктора 88
Совет 38. Приведение типов в случае множественного наследования ... 91
Совет 39. Приведение неполных типов 92
Совет 40. Приведения в старом стиле 93
Совет 41. Статические приведения 94
Совет 42. Инициализация формальных аргументов временными объектами 97
Совет 43. Время жизни временных объектов 100
Совет 44. Ссылки и временные объекты 101
Совет 45. Неоднозначность при использовании dynamic_cast 104
Совет 46. Контравариантность 108

Глава 5. Инициализация

Совет 47. Не путайте инициализацию и присваивание 111

Совет 48. Правильно выбирайте область видимости переменной 114
Совет 49. Внимательно относитесь к операциям копирования 116
Совет 50. Побитовое копирование объектов классов 119
Совет 51. Не путайте инициализацию и присваивание в конструкторах 121
Совет 52. Несогласованный порядок членов в списке инициализации ... 123
Совет 53. Инициализация виртуальных базовых классов 124
Совет 54. Инициализация базового класса в конструкторе копирования 128
Совет 55. Порядок инициализации статических данных во время выполнения 131
Совет 56. Прямая инициализация и инициализация копированием ... 133
Совет 57. Прямая инициализация аргументов 136
Совет 58. Что такое оптимизация возвращаемого значения? 137
Совет 59. Инициализация статических членов в конструкторе 141

Глава 6. Управление памятью и ресурсами

Совет 60. Различайте выделение и освобождение памяти для скаляров и для массивов 143
Совет 61. Контроль ошибок при выделении памяти 146
Совет 62. Подмена глобальных new и delete 148
Совет 63. Об области видимости и активации функций-членов new и delete 150
Совет 64. Строковые литералы в выражении throw 151
Совет 65. Обрабатывайте исключения правильно 154
Совет 66. Внимательно относитесь к адресам локальных объектов ... 157
Исчезающие фреймы стека 157
Затирание статических переменных 158
Идиоматические трудности 159
Проблемы локальной области видимости 159
Исправление ошибки путем добавления static 160
Совет 67. Помните, что захват ресурса есть инициализация 161
Совет 68. Правильно используйте auto_ptr 164

Глава 7. Полиморфизм

Совет 69. Кодирование типов 168
Совет 70. Невиртуальный деструктор базового класса 172
Неопределенное поведение 172
Виртуальные статические функции-члены 173
Всех обманем 174
Исключения из правил 175
Совет 71. Сокрытие невиртуальных функций 176
Совет 72. Не делайте шаблонные методы слишком гибкими 179
Совет 73. Перегрузка виртуальных функций 180

Совет 74. Виртуальные функции с аргументами по умолчанию 181
Совет 75. Вызовы виртуальных функций из конструкторов и деструкторов 183
Совет 76. Виртуальное присваивание 185
Совет 77. Различайте перегрузку, переопределение и сокрытие 187
Совет 78. О реализации виртуальных функций и механизма переопределения 192
Совет 79. Вопросы доминирования 197

Глава 8. Проектирование классов

Совет 80. Интерфейсы get/set 201
Совет 81. Константные и ссылочные данные-члены 204
Совет 82. В чем смысл константных функций-членов? 206
Синтаксис 206
Простая семантика и механизм работы 207
Семантика константной функции-члена 208
Совет 83. Различайте агрегирование и использование 210
Совет 84. Не злоупотребляйте перегрузкой операторов 214
Совет 85. Приоритеты и перегрузка 216
Совет 86. Операторы, являющиеся членами и друзьями класса 217
Совет 87. Проблемы инкремента и декремента 218
Совет 88. Неправильная интерпретация шаблонных операций копирования 221

Глава 9. Проектирование иерархий

Совет 89. Массивы объектов класса 224
Совет 90. Не всегда один контейнер можно подставить вместо другого 226
Совет 91. Что такое защищенный доступ? 229
Совет 92. Применение открытого наследования для повторного использования кода 232
Совет 93. Конкретные открытые базовые классы 235
Совет 94. Не пренебрегайте вырожденными иерархиями 236
Совет 95. Не злоупотребляйте наследованием 237
Совет 96. Управление на основе типов 240
Совет 97. Космические иерархии 242
Совет 98. Задание «интимных» вопросов объекту 244
Совет 99. Опрос возможностей 248

Список литературы 252

Предметный указатель 253



Предисловие

Эта книга – результат почти двадцатилетней работы, полной мелких разочарований, серьезных ошибок, бессонных ночей и выходных, добровольно проведенных за клавиатурой компьютера. Я включил в нее 99 глав, в которых описываются «скользкие места» (gotcha) в языке C++, которые иногда являются источниками распространенных ошибок и путаницы, а иногда просто вызывают интерес. С большинством из них я сталкивался лично (как это ни печально).

У слова «gotcha» довольно туманная история и множество определений. В этой книге мы будем понимать под ним типичную проблему, возникающую при проектировании и программировании на языке C++, которую можно предотвратить. В книге описаны самые разные проблемы такого рода: мелкие синтаксические тонкости, серьезные огрехи при проектировании и поведение, которое противно всем «нормам общежития».

Почти десять лет, как я начал включать замечания об отдельных скользких местах в материалы курса по C++, который я читаю. Мне казалось, что, обращая внимание студентов на типичные ошибки и просчеты, и одновременно показывая, как следует решать задачу правильно, я буду способствовать тому, что новые поколения программистов на C++ не станут повторять грехов своих предшественников. В общем и целом, эта идея оказалась удачной, и меня попросили подготовить собрание взаимосвязанных скользких мест для презентации на конференциях. Презентации завоевали популярность (не я один такой?), в результате чего я получил предложение написать книгу на эту тему.

Когда заходит речь о том, как не поскользнуться при работе с C++ или исправить последствия ошибки, нельзя не затронуть такие смежные вопросы, как наиболее распространенные паттерны проектирования, идиомы и технические детали языка.

Эта книга не о паттернах проектирования, но мы часто будем ссылаться на них как на средство обойти скользкое место. Названия паттернов по традиции принято писать с большой буквы, например: Template Method (Шаблонный Метод) или Bridge (Мост). При упоминании паттерна мы вкратце опишем его суть, если это не слишком сложно, но за подробным обсуждением отсылаем к работам, специально посвященным паттернам. Более полное описание конкретных паттернов, равно как и глубокое обсуждение этой темы в общем, можно найти в книге Эриха Гаммы и др. «Design patterns»*. Описания паттернов Acyclic Visitor (Ациклический ациклический Посетительпосетитель), Monostate (Моносостояниемоносостояние) и Null Object (Пустой пустой Объектобъект) можно найти в книге

---

* Имеется русский перевод: Гамма и др. «Паттерны проектирования». (*Прим. перев.*)

Robert Martin «Agile Software Development» («Разработка программ с удовольствием»).

С точки зрения скользких мест, у паттернов проектирования есть два важных свойства. Во-первых, каждый паттерн – это описание апробированной, неизменно приносящей успех техники проектирования, которую можно адаптировать под конкретные условия, возникающие при решении новых задач. Во-вторых, и это даже более важно, само упоминание о том, что в приложении используется тот или иной паттерн, документирует не только примененную технику, но также причины и результаты ее применения.

Например, если мы видим, что при проектировании программы был использован паттерн Bridge, то сразу понимаем, что реализация абстрактного типа данных разбита на интерфейсный класс и класс реализации. Кроме того, мы знаем, что сделано это было для того, чтобы разорвать связь между интерфейсом и реализацией, в результате чего изменение реализации никак не затронет пользователей интерфейса. Знаем мы и то, какие накладные расходы во время выполнения влечет за собой такое разделение, как организован исходный код, реализующий абстрактный тип данных, и целый ряд других деталей. Название паттерна – это недвусмысленная ссылка на кладезь информации и опыта, стоящий за соответствующей техникой. Обдуманное и правильное применение паттернов и связанной с ними терминологии при проектировании и документировании программ помогает понять код и не споткнуться на скользком месте.

C++ — это сложный язык программирования, а чем сложнее язык, тем важнее употребление идиом. В контексте языка программирования под идиомой понимается широко распространенная и имеющая всем понятный смысл комбинация низкоуровневых языковых средств, приводящая к появлению высокоуровневой конструкции. Такова же роль паттернов в проектировании программ. Поэтому мы и можем говорить об операциях копирования, функциональных объектах, интеллектуальных указателях и возбуждении исключений в C++, не опускаясь до уровня деталей реализации.

Важно подчеркнуть, что идиома — это не просто известная комбинация языковых средств, но и определенные ожидания относительно того, как эта комбинация будет себя вести. Каков смысл операции копирования? Чего ожидать в случае возбуждения исключения? Многие советы в этой книге касаются распознавания идиом и их использования в проектировании и кодировании. Можно сказать и так: многие из описанных скользких мест — не что иное, как игнорирование какой-то идиомы C++, а для решения проблемы часто всего-то и нужно, что следовать подходящей идиоме (см. «Совет 10»).

Немало глав в этой книге посвящено описанию некоторых нюансов языка, которые часто понимают неправильно, что и приводит к проблемам. Хотя некоторые примеры могут показаться надуманными, но незнание соответствующего материала не позволит вам стать настоящим экспертом по C++. Эти «закоулки» сами по себе могли бы стать предметом весьма любопытных и полезных исследований. Существуют они в C++ не без причины, а опытные программисты нередко прибегают к ним при разработке нетривиальных приложений.