

Наиболее эффективное использование C++

35 новых рекомендаций по улучшению ваших программ и проектов

Скотт Мейерс

«Я искренне рекомендую книгу "Наиболее эффективное использование C++" всем, кто стремится овладеть C++ на среднем или более высоком уровне».

Эксперт C/C++ User's Journal

Автор книги «Наиболее эффективное использование C++» предлагает 35 новых способов улучшения ваших программ. Основываясь на своем многолетнем опыте, С. Мейерс объясняет, как писать наиболее эффективные программы: надежные, совместимые, переносимые и пригодные для повторного использования, то есть программы, безупречные во всех отношениях.

Настоящая книга описывает приемы, которые позволяют значительно повысить производительность программ, выбрав оптимальное соотношение затрат времени/памяти на различные операции. Здесь вы найдете примеры обработки исключений и анализ их влияния на структуру и поведение классов и функций C++, а также варианты практического применения новых возможностей языка, таких как тип `bool`, ключевые слова `mutable` и `explicit`, пространства имен, шаблоны функций-членов, стандартная библиотека шаблонов и многое другое.

Internet-магазин:
www.aliants-kniga.ru

Книга – почтой:
Россия, 123242,
Москва, а/я 20
Тел.: (495) 258-9194
books@aliants-kniga.ru

Оптовая продажа:
«Альянс-книга»
Факс: (495) 258-9195
books@aliants-kniga.ru

ISBN 5-94074-033-2



9 785940 740339



www.dmk-press.ru

Скотт Мейерс

Наиболее эффективное использование C++



Наиболее эффективное использование C++

Скотт Мейерс

35 новых рекомендаций по улучшению

ваших программ и проектов



Объектно-ориентированный язык программирования



для программистов



Серия «Для программистов»

Наиболее эффективное использование C++

35 новых рекомендаций
по улучшению ваших
программ и проектов

Скотт Мейерс



Москва

УДК 004.4
ББК 32.973.26-018.2
М97

МейерсС.
М97 Наиболее эффективное использование С++. 35 новых рекомендаций по улучшению ваших программ и проектов. – М.: ДМК Пресс. – 294 с.: ил.

ISBN 5-94074-033-2

Автор книги «Наиболее эффективное использование С++» предлагает 35 новых способов улучшения ваших программ. Основываясь на своем многолетнем опыте, С. Мейерс объясняет, как писать наиболее эффективные программы: надежные, совместимые, переносимые и пригодные для повторного использования, то есть программы, безупречные во всех отношениях.

Настоящая книга описывает приемы, которые позволяют значительно повысить производительность программ, выбрав оптимальное соотношение затрат времени/памяти на различные операции. Здесь вы найдете примеры обработки исключений и анализ их влияния на структуру и поведение классов и функций С++, а также варианты практического применения новых возможностей языка, таких как тип `bool`, ключевые слова `mutable` и `explicit`, пространства имен, шаблоны функций-членов, стандартная библиотека шаблонов и многое другое.

УДК 004.438
ББК 32.973.26-018.2

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 5-94074-033-2

© Pearson Education, Inc
© Перевод на русский язык, оформление,
издание, ДМК Пресс, 2012

Содержание

Благодарности	9
Источники идей	9
Об этой книге	11
Люди, которые мне помогли	13
Введение	14
Глава 1. Основы	23
Правило 1. Различайте указатели и ссылки	23
Правило 2. Предпочитайте приведение типов в стиле C++	25
Правило 3. Никогда не используйте полиморфизм в массивах	30
Правило 4. Избегайте неоправданного использования конструкторов по умолчанию	33
Глава 2. Операторы	38
Правило 5. Опасайтесь определяемых пользователем функций преобразования типа	38
Правило 6. Различайте префиксную и постфиксную формы операторов инкремента и декремента	45
Правило 7. Никогда не перегружайте операторы &&, и ,	48
Правило 8. Различайте значение операторов new и delete	51
Глава 3. Исключения	57
Правило 9. Чтобы избежать утечки ресурсов, используйте деструкторы	58
Правило 10. Не допускайте утечки ресурсов в конструкторах	63
Правило 11. Не распространяйте обработку исключений за пределы деструктора	71
Правило 12. Отличайте генерацию исключения от передачи параметра или вызова виртуальной функции	73
Правило 13. Перехватывайте исключения, передаваемые по ссылке	80
Правило 14. Разумно используйте спецификации исключений	84
Правило 15. Оценивайте затраты на обработку исключений	90

Глава 4. Эффективность	94
Правило 16. Не забывайте о правиле «80–20»	95
Правило 17. Используйте отложенные вычисления	97
Правило 18. Снижайте затраты на ожидаемые вычисления	106
Правило 19. Изучите причины возникновения временных объектов	110
Правило 20. Облегчайте оптимизацию возвращаемого значения	113
Правило 21. Используйте перегрузку, чтобы избежать неявного преобразования типов	116
Правило 22. По возможности применяйте оператор присваивания вместо отдельного оператора	118
Правило 23. Используйте разные библиотеки	121
Правило 24. Учитывайте затраты, связанные с виртуальными функциями, множественным наследованием, виртуальными базовыми классами и RTTI	124
Глава 5. Приемы	134
Правило 25. Делайте виртуальными конструкторы и функции, не являющиеся членами класса	134
Правило 26. Ограничивайте число объектов в классе	140
Правило 27. В зависимости от ситуации требуйте или запрещайте размещать объекты в куче	154
Правило 28. Используйте интеллектуальные указатели	167
Правило 29. Используйте подсчет ссылок	190
Правило 30. Применяйте гроху-классы	218
Правило 31. Создавайте функции, виртуальные по отношению более чем к одному объекту	231
Глава 6. Разное	254
Правило 32. Программируйте, заглядывая в будущее	254
Правило 33. Делайте нетерминальные классы абстрактными	259
Правило 34. Умейте использовать в одной программе C и C++	270
Правило 35. Ознакомьтесь со стандартом языка	276
Приложение 1. Список рекомендуемой литературы	284
Приложение 2. Реализация шаблона auto_ptr	289
Алфавитный указатель	293

Благодарности

В создании этой книги принимало участие множество людей. Одни предложили важные технические идеи, другие помогли подготовить ее к печати, а третьи просто скрашивали мою жизнь, пока я работал над ней.

Часто, когда количество людей, принимавших участие в работе над книгой, достаточно велико, появляется соблазн отказаться от перечисления участников проекта, ограничившись стандартной фразой «Список людей, работавших над книгой, слишком длинен, чтобы быть приведенным здесь». Я, однако, предпочитаю подход Джона Л. Хеннеси (John L. Hennessey) и Дэвида А. Петерсона (David A. Patterson) – см. «Компьютерные архитектуры: численный подход», изд. Морган Кауфман (Morgan Kaufman), 1-ое издание, 1990. Один из аргументов за включение полного списка благодарностей, приведенного ниже, – статистические данные для закона «80–20», на который я ссылаюсь в правиле 16.

Источники идей

За исключением прямого цитирования, весь текст этой книги принадлежит мне. Тем не менее, многие описанные в ней идеи были придуманы другими. Я всячески пытался отслеживать авторство нововведений, но мне все же пришлось включить информацию из источников, названия которых я уже не могу вспомнить, в основном это сообщения из конференций Usenet [comp.lang.c++](#) и [comp.std.c++](#).

Многие идеи в сообществе C++ зарождаются почти одновременно и совершенно независимо в головах многих людей. Ниже я указываю только, где услышал ту или иную мысль, что не всегда совпадает с тем, где она была озвучена впервые.

Брайан Керниган (Brian Kernighan) предложил использовать макроопределения для приближения к синтаксису новых операторов приведения типа, описанных в правиле 2.

Предупреждение по поводу удаления массива объектов производного класса с помощью указателя на базовый класс, изложенное в правиле 3, основано на материалах лекции Дэна Сакса (Dan Saks), прочитанной им на нескольких конференциях и торговых выставках.

Техника использования ргоху-классов из правила 5, позволяющая избежать нежелательного вызова конструкторов с одним аргументом, основана на материалах колонки Эндрю Кенига (Andrew Koenig) в журнале C++ Report за январь 1994 года.

Джеймс Канце (James Kanze) прислал сообщение в [comp.lang.c++](#) относительно реализации постфиксных декрементных и инкрементных операторов через соответствующие префиксные операторы. Этот прием рассматривается в правиле 6.

Дэвид Кок (David Cok), написав мне по одному вопросу, затронутому в «Эффективном использовании C++», привлек мое внимание к различию между `operator new` и оператором `new`, положенному в основу правила 8. Даже прочитав письмо, я не в полной мере осознал существующую разницу, но если бы не этот первый толчок, то, скорее всего, не понимал бы ее до сих пор.

Метод записи деструкторов, позволяющий избежать утечки ресурсов (см. правило 9), взят из раздела 15.3 книги Маргарет А. Эллис (Margaret A. Ellis) и Бьерна Страуструпа (Bjarne Stroustrup) *The Annotated C++ Reference Manual*. Там этот метод имеет название «Выделение ресурса – инициализация». Том Каргилл (Tom Cargill) предложил перенести акцент с выделения ресурсов на их освобождение.

Часть рассуждений в разделе, посвященном правилу 11, была навеяна содержанием главы 4 книги Taligent's Guide to Designing Programs, изд. Addison-Wesley, 1994.

Описание предварительного выделения памяти для класса `DynArray` в правиле 18 основано на статье Тома Каргилла «Динамический вектор сложнее, чем кажется», опубликованной в журнале *C++ Report* за июнь 1992 года. Информацию о более сложной архитектуре для класса динамического массива можно найти в заметке того же автора (номер *C++ Report* за январь 1994 года).

Правило 21 появилось благодаря докладу Брайана Кернигана «AWK для транслятора C++» на конференции USENIX по C++ в 1991 году. Его идея использовать перегруженные операторы (общим числом 67!) для выполнения арифметических операций с операндами разных типов хотя и не была связана с проблемой, обсуждаемой в правиле 21, но заставила меня рассмотреть множественную перегрузку операторов в качестве решения задачи по созданию временных объектов.

Мой вариант шаблона класса для подсчета объектов, рассмотренный в правиле 26, основан на сообщении Джамшида Афшара (Jamshid Afshar) в конференцию [comp.lang.c++.](#)

Идея смешанного класса, позволяющего отслеживать указатели, созданные с помощью `operator new` (см. правило 27), базируется на предложении Дона Бокса (Don Box). Стив Клемидж (Steve Clamage) придал этой идее практическое значение, объяснив, как можно использовать `dynamic_cast` для нахождения начала области памяти, занимаемой объектом.

Описание `smart`-указателей в правиле 28 основано: частично на заметке Стивена Буроффа (Steven Buroff) и Роба Мюррея (Rob Murray) *C++ Oracle* в журнале *C++ Report* за октябрь 1993 года, на классической работе Даниэла Р. Эдельсона (Daniel R. Edelson) «Интеллектуальные (`smart`) указатели: интеллектуальные, но не указатели» в материалах конференции USENIX по C++ от 1992 года, на содержимом раздела 15.9.1 книги Бьерна Страуструпа «Архитектура и развитие C++», на докладе Грегори Колвина (Gregory Colvin) «Управление памятью в C++» на учебном семинаре «Решения для C/C++ '95» и на заметке Кея Хорстманна (Kay Horstmann) в мартовском и апрельском номерах *C++ Report* за 1993 год. Но кое-что сделал и я сам.

Использованный в правиле 29 метод хранения в базовом классе счетчиков ссылок и `smart`-указателей для работы с этими счетчиками основан на идее Роба Мюррея (см. разделы 6.3.2 и 7.4.2 его книги «Стратегия и тактика в C++»). Прием, позволяющий добавлять счетчики ссылок к существующим классам, аналогичен тому, что был предложен Кеем Хорстманном в заметке, опубликованной в мартовском и апрельском номерах журнала *C++ Report* за 1993 год.

Источником для правила 30, касающегося контекстов `lvalue`, послужили комментарии к заметке Дэна Сакса в журнале *C User's Journal* (теперь *C/C++*

User's Journal) за январь 1993 года. Наблюдение, что методы классов, которые не являются методами гроху-классов, не доступны при вызове по гроху-механизму, взято из неопубликованной работы Кея Хорстманна.

Способ, как использовать динамическую информацию о типах для того, чтобы построить похожие на vtbl массивы указателей функций (в правиле 31), основан на идеях Бьерна Страуструпа, выдвинутых им в сообщениях в конференцию `comp.lang.c++.` и разделе 13.8.1 его книги «Архитектура и развитие C++».

Сведения, на базе которых появилось правило 33, частично были опубликованы в моих колонках журнала C++ Report за 1994 и 1995 года. Эти колонки, в свою очередь, включали замечания об использовании `dynamic_cast` для реализации виртуального оператора `operator=`, определяющего наличие аргументов некорректного типа, которые я получил от Клауса Крефта (Klaus Krefte).

Большая часть рассуждений в правиле 34 вызвана статьей Стива Клемиджа «Связывание C++ с другими языками» в мартовском номере журнала C++ Report за 1992 год. Мой подход к решению проблем, вызванных использованием таких функций, как `strdup`, был инициирован замечаниями читателя, не сообщившего своего имени.

Об этой книге

Просмотр черновых вариантов книги – работа неблагодарная, но жизненно необходимая. Мне повезло, что так много людей пожелали вложить в нее свое время и энергию. Хочу особенно поблагодарить: Джил Хатчитэл (Jill Huchital), Тима Джонсона (Tim Johnson), Брайана Кернигана, Ерика Наглера и Криса Ван Вых (Chris Van Wyk), потому что они прочли мою книгу (или ее значительную часть) более одного раза. Кроме этих любителей неприятной работы полностью черновики книги прочли: Катрина Эвери (Katrina Avery), Дон Бокс, Стив Буркетт (Steve Burkett), Том Каргилл, Тони Дэвис (Tony Davis), Кэролин Даби (Carolyn Duby), Брюс Экель (Bruce Eckel), Рид Флеминг (Read Fleming), Кей Хорстманн, Джеймс Канце, Расс Пейли (Russ Paily), Стив Розенталь (Steve Rosenthal), Робин Руйе (Robin Rowe), Дэн Сакс, Крис Селлз (Chris Sells), Уэбб Стейси (Webb Stacy), Дэйв Свифт (Dave Swift), Стив Виноски (Steve Vinosky) и Фред Уайлд (Fred Wild). Частично черновики прочли: Боб Бьючейн (Bob Beauchaine), Герд Хойрен (Gerd Hoeren), Джефф Джексон (Jeff Jackson) и Нэнси Л. Урбано (Nancy L. Urbano). Замечания каждого из них помогли представить материал более точно и доступно.

После выхода книги я получил исправления и предложения от множества людей. Ниже эти наблюдательные читатели перечислены в порядке получения от них сообщений: Льюис Кида (Luis Kida), Джон Поттер (John Potter), Тим Уттормарк (Tim Uttormark), Майк Фелькерсон (Mike Fulkerson), Дэн Сакс, Вольфганг Глунц (Wolfgang Glunz), Кловис Тондо (Clovis Tondo), Майкл Лофтус (Michael Loftus), Лиз Хэнкс (Liz Hanks), Вил Эверс (Wil Evers), Стефан Кухлинз (Stefan Kuhlins), Джим МакКракен (Jim McCracken), Элан Дучан (Alan Duchan), Джон Джекобсма (John Jacobsma), Рамеш Нагабушнам (Ramesh Nagabushnam), Эд Виллинк (Ed Willink), Кирк Свенсон (Kirk Swenson), Джек Ривз (Jack Reeves), Дуг

Шмидт (Doug Schmidt), Тим Бучовски (Tim Buchowski), Пол Чисхолм (Paul Chisholm), Эндрю Клейн (Andrew Klein), Эрик Наглер, Джефффри Смит (Jeffrey Smith), Сэм Бент (Sam Bent), Олег Штейнбук (Oleg Shteynbuk), Антон Доблмайер (Anton Doblmaier), Ульф Михаэлис (Ulf Michaelis), Секхар Муддана (Sekhar Muddana), Майкл Бейкер (Michael Baker), Йечил Кимчи (Yechiel Kimchi), Дэвид Папюрт (David Papurt), Йан Хэггард (Ian Haggard), Роберт Шварц (Robert Schwartz), Дэвид Хэлпин (David Halpin), Грэхам Марк (Graham Mark), Дэвид Баретт (David Barrett), Дэмьен Канарек (Damian Kanarek), Рон Коуттс (Ron Coutts), Ланс Витцель (Lance Whitesel), Йон Лачелт (Jon Lachelt), Шерил Фергюсон (Cheryl Ferguson), Мунир Махмуд (Munir Mahmood), Клаус-Георг Адами (Klaus-Georg Adams), Дэвид Гох (David Goh), Крис Морли (Chris Morley), Рейнер Баумшлагер (Rainer Baumschlager), Брайан Керниган, Чарльз Грин (Charles Green), Марк Роджерс (Mark Rodgers), Бобби Шмидт (Bobby Schmidt), Шиварамахришан Дж. (Sivaramakrishnan J.) и Эрик Андерсон (Eric Anderson). Их предложения позволили мне улучшить книгу, и я очень благодарен им за помощь.

При подготовке этой книги я сталкивался с множеством вопросов, связанных с появлением стандарта ISO/ANSI для языка C++, решить которые мне помогли Стив Клэмидж и Дэн Сакс. Они не пожалели времени, отвечая на мои беспрестанные вопросы по электронной почте.

Джон Макс Скаллер (John Max Skaller) и Стив Рамсби (Steve Rumsby) помогли мне получить текст ANSI-стандарта C++ в формате HTML до его публикации. Вивиан Ней (Vivian Neou) подсказала мне, что для просмотра HTML-документов в 16-битной системе Microsoft Windows можно использовать браузер Netscape. Я глубоко благодарен сотрудникам компании Netscape Communications за бесплатное распространение своего браузера для этой системы.

Брайан Хоббс (Bryan Hobbs) и Хачеми Зенад (Hachemi Zenad) предоставили мне предварительную версию компилятора MetaWare C++, что позволило проверить тексты программ, приведенных в этой книге, с использованием самых новых свойств языка. Кей Хорстманн помог мне с установкой и запуском компилятора в чуждых для меня мирах DOS и защищенного режима DOS. Корпорация Borland (теперь Inprise) предоставила мне последнюю бета-версию своего компилятора, а Эрик Наглер и Крис Селлз обеспечили неоценимую помощь, проверив тексты программ на недоступных для меня компиляторах.

Книга не могла бы появиться на свет без помощи сотрудников отдела корпоративной и специальной литературы издательства Addison-Wesley. Я очень обязан: Ким Доули (Kim Dawley), Лане Лэнглуа (Lana Langlois), Симоне Пэймент (Simone Payment), Марти Рабинович (Marty Rabinowitz), Прадипе Сива (Pradeepa Siva), Джону Уэйту (John Wait) и другим сотрудникам за их терпение, поддержку и помощь в подготовке этой работы.

Крис Гузиковски (Chris Guzikovsky) помогал проектировать обложку книги, а Тим Джонсон (Tim Johnson) уделил часть своего времени, обычно всецело посвященного исследованиям в области низкотемпературной физики, для критических замечаний по последним версиям этого текста.