Новые сложные задачи на C++

40 новых головоломных задач с решениями

Герб Саттер

Издательский дом “Вильямс”
Москва • Санкт-Петербург • Киев
2008

Exceptional C++ Style

40 New Engineering Puzzles, Programming Problems, and Solutions

Herb Sutter

ADDISON–WESLEY

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

Новые сложные задачи на C++

ББК 32.973.26-018.2.75
С21
УДК 681.3.07

Издательский дом "Вильямс"

Зав. редакцией *С.Н. Тригуб*

Перевод с английского и редакция канд. техн. наук *И.В. Красикова*

Научный консультант канд. техн. наук *А.Н. Кротов*

По общим вопросам обращайтесь в Издательский дом "Вильямс" по адресу:
info@williamspublishing.com, http://www.williamspublishing.com

Саттер, Герб.

С21 Новые сложные задачи на C++. : Пер. с англ. — М. : ООО "И. Д. Вильямс", 2008. — 272 с. : ил. — Парал. тит. англ.

ISBN 978-5-8459-0823-0 (рус.)

Данная книга представляет собой продолжение вышедшей ранее книги *Решение сложных задач на C++*. В форме задач и их решений рассматриваются современные методы проектирования и программирования на C++. В книге сконцентрирован богатый многолетний опыт программирования на C++ не только самого автора, но и всего сообщества программистов на C++, так что некоторые рекомендации автора могут показаться неожиданными даже опытным программистам-профессионалам. Автор рассматривает и конкретные методики, приемы и идиомы программирования, однако основная тема книги — это стиль программирования, причем в самом широком понимании этого слова. Особое внимание во всех задачах книги уделено вопросу проектирования, которое должно обеспечить максимальную надежность, безопасность, производительность и сопровождаемость создаваемого программного обеспечения.

Книга рассчитана в первую очередь на профессиональных программистов с глубокими знаниями языка, однако она будет полезна любому, кто захочет углубить свои знания в данной области.

ББК 32.973.26-018.2.75

ISBN 978-5-8459-0823-0 (рус.)
ISBN 0-201-76042-8 (англ.)

Оглавление

Содержание

Тине, моей жене и лучшему другу

Предисловие

Место действия: Будапешт. Жаркий летний вечер. Мы смотрим через Дунай, на восточный берег реки.

На обложке книги вы видите фото, на котором изображена эта пастельная европейская картина. Что первое бросается вам в глаза? Почти наверняка — здание парламента в левой части фотографии. Массивное неоготическое здание приковывает взгляд своим изящным куполом, массой вычурных шпилей, десятками статуй и прочими украшениями, контрастируя с простыми строгими линиями зданий на набережной Дуная.

Откуда же такое отличие? Строительство здания парламента было завершено в 1902 году, в то время как остальные здания на набережной были построены в разрушенном Будапеште после второй мировой войны.

"Ну и что же, — скажете вы, — какое отношение это имеет к книге?"

Стиль — это всегда нечто большее, чем просто внешний вид, и за ним скрывается целая философия и мировоззрение — будь то в архитектуре строительства или в архитектуре программного обеспечения. Я думаю, что вам попадались программы, напоминающие своей "пышностью" и размерами здание парламента, равно как уверен, что вам доводилось видеть и программы, напоминающие блочно-панельное строительство.

Стиль или суть?

Что же важнее? Чему лучше и правильнее отдать предпочтение? Вы уверены, что знаете точный ответ на этот вопрос? Так, понятие "лучше" лишено смысла, пока не определена мера, которой следует мерить. Лучше для чего? Лучше в какой ситуации? Скорее всего, ответ на этот вопрос представляет собой определенный компромисс и начинается со слов "Это зависит от..."

Это книга о поиске баланса между многими мелкими аспектами дизайна и реализации программ на C++. Глубокое знание ваших инструментов и исходных материалов весьма способствует пониманию того, когда их стоит использовать.

Так *лучше* ли здание парламента и его стиль, чем у зданий, находящихся рядом с ним? Очень легко, не думая, ответить "да". Но ответ должен основываться не только на эмоциях, но и на логике. Представьте, насколько не просто построить такое здание и поддерживать его в должном состоянии.

- *Строительство.* В 1902 году, когда закончилось его строительство, это здание было самым большим в мире зданием парламента. Эта грандиозность, конечно же, сказалась на его стоимости, продолжительности строительства и количестве затраченных усилий. Так был создан "белый слон", т.е. нечто интересное само по себе, но со стоимостью, которую не оправдывает никакой интерес. Как вы думаете, сколько обычного жилья, которое пусть и не потрясает воображение,

но дает кров над головой, можно было бы построить при тех же капиталовложениях? Наверное, ответ на этот вопрос мог бы впечатлить многих. Позвольте напомнить вам, что все мы работаем в той отрасли промышленности, где давление сроков разработки ощущается особо сильно — и в принципе несравнимо с таковым во времена постройки рассматриваемого здания.

- *Поддержка*. Присмотритесь к фотографии, и вы увидите, что часть здания покрыта лесами. Реставрационные работы идут здесь годами, и на это затрачиваются такие суммы, что, пожалуй, было бы проще снести это здание и построить что-то новое. На фотографии не видно (да и не может быть видно) кое-что еще. Например, скульптуры, украшающие здание, были сделаны из плохо подобранного материала, который слишком легко разрушается, так что их реставрация и замена были начаты едва ли не сразу же после завершения строительства, и все эти украшения, "рюшечки и финтифлюшечки" — предмет постоянной заботы реставраторов уже *более века*.

Так и в программировании — очень важно найти золотую середину между стоимостью и функциональностью, между элегантностью и сопровождаемостью, между возможностями развития и украшательством.

С подобными проблемами и поиском компромиссов мы вынуждены сталкиваться ежедневно при разработке программного обеспечения на C++. Среди вопросов, которые рассматриваются в данной книге, есть и такие: делает ли безопасность кода по отношению к исключениям лучше сам код? Если да — то что именно означает "делает его лучше", и не может ли возникнуть ситуация, когда это не так уж и хорошо? А как насчет инкапсуляции? Делает ли она программу лучше? Почему? При каких условиях это не так? Если вас заинтересовали эти вопросы — книга перед вами, прочтите ее. Кстати, встраиваемые функции — это хорошая оптимизация? Следует ли к ней прибегать? (Будьте очень-очень осторожны при ответе на этот вопрос.) Что общего между возможностью экспорта в C++ и зданием парламента? А между `std::string` и монолитной архитектурой зданий на набережной Дуная?

После рассмотрения множества различных технологий и возможностей C++, в конце книги целый раздел отведен для анализа реальных примеров опубликованных исходных текстов. Мы выясним, что авторам этих фрагментов удалось, что не совсем, и как исправить баланс между затрачиваемыми усилиями и хорошим стилем.

Я надеюсь, что эта книга, а также предыдущие книги по данной теме[1] помогут вам шире взглянуть на C++, прибавят вам знаний о деталях и тонкостях языка, расскажут о его внутренних взаимосвязях и помогут вам в поиске золотой середины при разработке собственных программ.

Взгляните еще раз на фотографию на обложке книги, в правый верхний угол. Видите там воздушный шар? Вот так и мы должны подняться над городом и увидеть его весь, во всей перспективе — красоту и изящество одних строений, простоту и надежность других, понять, что стиль и суть взаимосвязаны, и они не просто сосуществуют, но и взаимодействуют и дополняют друг друга. Поднявшись над городом, мы видим не отдельные дома, но весь город в его красоте и неповторимости, и выработать именно этот взгляд на C++ — во всей его красоте и целостности — должна помочь нам данная книга.

Метод Сократа

Греческий философ Сократ обучал своих учеников, задавая им вопросы, которые были разработаны таким образом, чтобы направлять мышление учеников и помогать

[1] Вышедшие в издательстве "Вильямс" объединенными в одну книгу: *Саттер Г. Решение сложных задач на C++. Серия C++ In-Depth, т.4. М.: Издательский дом "Вильямс", 2004. — Прим. ред.*

им сделать верные выводы из того, что они уже знают, а также показать им взаимосвязь изучаемого материала с другими знаниями. Этот метод обучения стал так популярен, что сегодня мы называем его "методом Сократа". С точки зрения учащихся подход Сократа включает их в процесс обучения, заставляет думать и помогает применить уже имеющиеся знания к новой информации.

Эта книга вполне следует методу Сократа, как и ее предшественницы [Sutter00] и [Sutter02]. Предполагается, что вам приходится заниматься написанием промышленного программного обеспечения на языке C++; в книге используются вопросы и ответы для обучения эффективному применению стандарта C++ и его стандартной библиотеки, причем особое внимание уделяется разработке надежного программного обеспечения с использованием всех возможностей современного C++. Многие из рассмотренных в книге задач появились в результате работы автора и других программистов над своими программами. Цель книги — помочь читателю сделать верные выводы, как из хорошо известного ему материала, так и из только что изученного, и показать взаимосвязь между различными частями C++.

Данная книга не посвящена какому-то конкретному аспекту C++. Нельзя, однако, сказать, что она охватывает все детали C++ — для этого потребовалось бы слишком много книг, — но, тем не менее, в ней рассматривается широкая палитра возможностей C++ и стандартной библиотеки и, что немаловажно, демонстрируется, как кажущиеся на первый взгляд несвязанными между собой вещи могут совместно использоваться для получения новых решений старых и хорошо известных задач. Здесь вы найдете материал, посвященный шаблонам и пространствам имен, исключениям и наследованию, проектированию надежных классов и шаблонам проектирования, обобщенному программированию и магии макросов, — и не просто винегрет из этих вопросов, а задачи и решения, выявляющие взаимосвязь всех этих частей современного C++.

Эта книга продолжается с того места, где заканчивается изложение материала в [Sutter00] и [Sutter02], и следует той же традиции: Материал книги подается в виде задач, сгруппированных по темам. Читатели первых книг найдут здесь наполненные новым содержанием уже знакомые им темы — безопасность исключений, обобщенное программирование, методы оптимизации и управления памятью. Основное внимание уделяется вопросам обобщенного программирования и эффективного использования стандартной библиотеки C++.

Большинство задач первоначально были опубликованы в Internet и некоторых журналах, в частности, это расширенные версии задач 63–86, которые можно найти на моем узле *Guru of the Week* [GotW], а также материалы, опубликованные мною в таких журналах, как *C/C++ User Journal*, *Dr. Dobb's Journal*, бывшем *C++ Report* и др. Последние исправления и дополнения к книге можно найти на Web-узле по адресу `www.gotw.ca`.

Как читать данную книгу

Предполагается, что читатель уже хорошо знаком с основами C++. Если это не так, начните с хорошего введения и обзора по C++. Для этой цели могу порекомендовать вам такие книги, как [Stroustrup00], [Lippman98], а также [Meyers96] и [Meyers97].

Каждая задача в книге имеет заголовок, который выглядит, как показано ниже.

Задача №. Название задачи **Сложность: X**

Название задачи и уровень ее сложности подсказывают вам, для кого она предназначена. Обычно в задаче есть вопрос для новичка, что позволяет размяться прежде

чем приступить к главной части — вопросу для профессионала. Замечу, что указанный уровень сложности задач — это мое субъективное мнение относительно того, насколько сложно будет решить ту или иную задачу большинству читателей, так что вы вполне можете обнаружить, что задача с уровнем сложности 7 решена вами гораздо быстрее, чем задача с уровнем сложности 5. Со времени выхода в свет предыдущих книг я получил немало писем, в которых читатели утверждали, что задача *M* сложнее (проще) задачи *N*. Это только подтверждает мой тезис о субъективности оценок и их зависимости от конкретных знаний и опыта читателя.

Чтение книги от начала до конца — неплохое решение, но вы не обязаны поступать именно так. Вы можете, например, читать только интересующие вас разделы книги. За исключением того, что я называю "мини-сериями" (связанные между собой задачи с одинаковым названием и подзаголовками "Часть 1", "Часть 2" и т.д.), задачи в книге практически независимы друг от друга, и вы можете читать их в любом порядке. Единственная подсказка: мини-серии лучше читать вместе; во всем остальном — выбор за вами.

Все примеры кода — всего лишь отрывки программ, если не оговорено иное, и не следует ожидать, что эти отрывки будут корректно компилироваться при отсутствии остальных частей программы. Для этого вам придется самостоятельно дописывать недостающие части.

И последнее замечание — об URL: нет ничего более изменчивого, чем Web, и более мучительного, чем давать ссылки на Web в печатной книге: зачастую эти ссылки устаревают еще до того, как книга попадает в типографию. Так что ко всем приведенным в книге ссылкам следует относиться критически, и в случае их некорректности — пишите мне. Дело в том, что все ссылки даны через мой Web-узел `www.gotw.ca`, и в случае некорректности какой-либо ссылки я просто обновлю перенаправление к новому местоположению страницы (если найду ее) или укажу, что таковой больше нет (если не смогу ее найти).

Благодарности

В первую очередь я хочу поблагодарить мою жену Тину (Tina) за ее терпение, любовь и поддержку, а также всю мою семью за то, что они всегда со мной, как при написании книг, так и в любое другое время. Без их безграничного терпения и участия книга бы не получилась такой, какой вы ее видите.

Я выражаю особую признательность редактору серии Бьерну Страуструпу (Bjarne Stroustrup), а также редакторам Питеру Гордону (Peter Gordon) и Дебби Лафферти (Debbie Lafferty), Тирреллу Альбах (Tyrrell Albaugh), Бернарду Гафни (Bernard Gaffney), Курту Джонсону (Curt Johnson), Чанда Лири-Куту (Chanda Leary-Coutu), Чарльзу Ледди (Charles Leddy), Мелинде Мак-Кейн (Malinda McCain), Чати Прасерсиху (Chuti Prasertsith) и всем остальным членам команды Addison-Wesley за их помощь и настойчивость в работе над данным проектом. Трудно представить себе лучшую команду для работы над данной книгой; их энтузиазм и помощь помогли сделать эту книгу тем, чем она, я надеюсь, является.

Еще одна группа людей, заслуживающих особой благодарности, — это множество экспертов, чья критика и комментарии помогли сделать материал книги более полным, более удобочитаемым и более полезным. Особую благодарность я хотел бы выразить Бьерну Страуструпу (Bjarne Stroustrup), Дэйву Абрамсу (Dave Abrahams), Стиву Адамчику (Steve Adamczyk), Андрею Александреску (Andrei Alexandrescu), Чаку Аллисону (Chuck Allison), Мэтту Остерну (Matt Austern), Йоргу Барфурту (Joerg Barfurth), Питу Беккеру (Pete Becker), Брендону Брею (Brandon Bray), Стиву Дьюхарсту (Steve Dewhurst), Джонатану Кейвзу (Jonathan Caves), Питеру Димову (Peter Dimov), Хавьеру Эстрада (Javier Estrada), Аттиле Фехеру (Attila Fehér), Марко Далла Гасперина (Marco Dalla Gasperina), Дугу Грегору (Doug Gregor), Марку Холлу (Mark Hall), Кэвлину Хен-

ни (Kevlin Henney), Говарду Хиннанту (Howard Hinnant), Кэю Хорстману (Cay Horstmann), Джиму Хайслопу (Jim Hyslop), Марку Камински (Mark E. Kaminsky), Дэннису Манклу (Dennis Mancl), Брайену Мак-Намара (Brian McNamara), Скотту Мейерсу (Scott Meyers), Джеффу Пейлу (Jeff Peil), Джону Поттеру (John Potter), П. Плагеру (P. J. Plauger), Мартину Себору (Martin Sebor), Джеймсу Слотеру (James Slaughter), Николаю Смирнову (Nikolai Smirnov), Джону Спайсеру (John Spicer), Яну Кристиану ван Винклю (Jan Christiaan van Winkel), Дэвиду Вандевурду (Daveed Vandevoorde) и Биллу Вейду (Bill Wade). Все оставшиеся в книге ошибки, описки и неточности — только на моей совести.

Еще одна благодарность — нашему щенку Франки (Frankie), который оттаскивал меня от стола и тащил дышать свежим воздухом, без чего, конечно, моя работа никогда не была бы закончена. Замечу, что Франки ничего не знает о программировании, оптимизации, архитектуре программного обеспечения, и при этом он всегда выглядит счастливым и довольным. Над этим стоит задуматься...

Герб Саттер (Herb Sutter)
Сиэтл, май 2004

От издательства

Вы, читатель этой книги, и есть главный ее критик и комментатор. Мы ценим ваше мнение и хотим знать, что было сделано нами правильно, что можно было сделать лучше и что еще вы хотели бы увидеть изданным нами. Нам интересно услышать и любые другие замечания, которые вам хотелось бы высказать в наш адрес.

Мы ждем ваших комментариев и надеемся на них. Вы можете прислать нам бумажное или электронное письмо, либо просто посетить наш Web-сервер и оставить свои замечания там. Одним словом, любым удобным для вас способом дайте нам знать, нравится или нет вам эта книга, а также выскажите свое мнение о том, как сделать наши книги более интересными для вас.

Посылая письмо или сообщение, не забудьте указать название книги и ее авторов, а также ваш обратный адрес. Мы внимательно ознакомимся с вашим мнением и обязательно учтем его при отборе и подготовке к изданию последующих книг. Наши координаты:

E-mail: `info@williamspublishing.com`

WWW: `http://www.williamspublishing.com`

Информация для писем из:

России: 127055, Москва, ул. Лесная, д. 43, стр. 1

Украины: 03150, Киев, а/я 152

Обобщенное программирование и стандартная библиотека C++

Одной из наиболее мощных возможностей C++ является поддержка обобщенного программирования. Эта возможность находит непосредственное отражение в гибкости стандартной библиотеки C++, в особенности в контейнерах, итераторах и алгоритмах, известных под названием стандартной библиотеки шаблонов (Standard Template Library — STL).

Так же, как и предыдущая книга [Sutter02], эта книга начинается с задач, которые привлекают наше внимание к некоторым хорошо знакомым частям STL, в частности к векторам и строкам. Сможете ли вы избежать широко распространенных ловушек при использовании такого базового контейнера STL, как вектор? Как вы выполните обычные манипуляции со строками в C++? Какие уроки в плане конструирования библиотек вы сможете извлечь для себя из STL?

После того как мы разберемся с предопределенными шаблонами STL, мы обратимся к более общим вопросам, связанным с шаблонами и обобщенным программированием на C++. Как можно избежать при разработке собственного шаблонного кода его *не*обобщенности? Почему специализация шаблонов функций — не лучшая идея, и что следует делать вместо этого? Как корректно и переносимо добиться тех же результатов, которые дают отношения дружественности? И что нам дает ключевое слово `export`?

Эти и другие вопросы будут рассмотрены нами в разделе, посвященном обобщенному программированию и стандартной библиотеке C++.

Задача 1. Вектор: потребление и злоупотребление — Сложность: 4

Почти все используют `std::vector`, и это хорошо. К сожалению, многие не всегда верно понимают его семантику и в результате невольно применяют его странными, а порой и опасными способами. Сколько из перечисленных ниже проблем можно найти в ваших программах?

Вопрос для новичка

1. В чем разница между строками *A* и *B*?

```
void f( vector<int>& v ) {
    v[0];        // A
    v.at(0);     // B
}
```

Вопрос для профессионала

2. Рассмотрим следующий код.

```
vector<int> v;

v.reserve( 2 );
assert( v.capacity() == 2 );
v[0] = 1;
v[1] = 2;
for(vector<int>::iterator i = v.begin(); i<v.end(); i++){
    cout << *i << endl;
}

cout << v[0];
v.reserve( 100 );
assert( v.capacity() == 100 );
cout << v[0];

v[2] = 3;
v[3] = 4;
// ...
v[99] = 100;
for( vector<int>::iterator i = v.begin(); i<v.end(); i++){
    cout << *i << endl;
}
```

Раскритикуйте этот код, как с точки зрения стиля, так и с точки зрения корректности.

Решение

Обращение к элементу вектора

1. В чем разница между строками *A* и *B*?

```
void f( vector<int>& v ) {
    v[0];        // A
    v.at(0);     // B
}
```