Нашим студентам,
благодаря которым мы поняли,
как нужно учить других

Введение

Новый подход к программированию на C++

Если вы читаете эти строки, то вам, вероятно, хотелось бы быстро научиться писать сто*я*щие программы на языке C++. Посему сразу же возьмем "быка за рога" и рассмотрим самые полезные разделы C++. Такая стратегия, возможно, станет понятной в процессе осуществления; основной ее принцип состоит в том, что мы не будем начинать с изучения языка C, несмотря на то что C++ строится на C. Мы сразу же начнем с использования структур данных высокого уровня и только позднее рассмотрим детали, на которых они базируются. Такой подход позволит вам немедленно приступить к написанию правильно организованных программ на C++.

Данный подход имеет еще одну необычную особенность. Мы делаем акцент не на изучении возможностей самого языка и библиотек, а на решении проблем. Конечно же, мы не уходим от разъяснения применяемых возможностей, но делаем это для поддержки программ, а не для того, чтобы использовать программы в качестве демонстрации этих возможностей.

Цель этой книги — научить программированию на C++, а не просто изложить средства языка; она особенно полезна читателям, которые уже знакомы с C++ и хотят использовать этот язык более эффективно. Ведь довольно часто приходится сталкиваться с тем, что новички в C++ пытаются освоить язык чисто механически, т.е. не пытаясь узнать, как применить его к решению каждодневных проблем.

Наша книга полезна как для новичков, так и для опытных программистов

Каждое лето мы практиковали недельный курс интенсивного изучения языка C++ в Станфордском университете (Stanford University). Для этого курса изначально был принят традиционный подход. Предполагая, что студенты уже знают C, мы начинали с демонстрации использования классов, а затем систематически "проходили" остальные разделы языка. Мы обнаружили, что наши студенты пребывали в растерянности в течение почти двух дней — до тех пор, пока не узнавали достаточно для того, чтобы приступить к написанию программ. С этого момента обучение продвигалось быстрее.

При переходе к C++-среде, которая поддерживала совершенно новую на тот момент стандартную библиотеку, мы тщательно пересмотрели свой курс. В новом курсе с самого начала использовалась эта библиотека, и внимание студентов акцентировалось именно на написании программ, а углубленное рассмотрение деталей мы практиковали только после того, как студенты получали достаточно информации, чтобы продуктивно их использовать.

Результаты оказались просто ошеломляющими. Через день после начала занятий наши студенты могли писать программы, в отличие от предыдущего курса, когда они приходили к этому лишь к концу недели. Более того, от растерянности слушателей наших курсов не осталось и следа.

Абстракция

Наш подход стал возможен только благодаря усовершенствованию C++ и нашему пониманию этого языка, что позволило нам игнорировать многие второстепенные темы, которые ставились во главу угла программистами раннего периода "жизни" C++.

Способность игнорировать детали — характерная особенность "зрелости" любых технологий. Например, первые автомобили ломались так часто, что каждому автомобилисту одновременно приходилось быть и автомехаником. Было полным безрассудством в то время отправляться в дорогу, не зная деталей устройства автомобиля. Современным водителям не нужны глубокие знания автомеханика, чтобы использовать автомобиль в качестве транспортного средства. Они, конечно, могут вникать в детали своего "железного коня" по каким-то иным причинам, но это совершенно другая история.

Мы определяем абстракцию как избирательное неведение, которое выражается в сосредоточении внимания на идеях текущей задачи и игнорировании всего остального. На наш взгляд, это самое важное в современном программировании. Ключ к написанию успешно работающей программы лежит в знании того, какие части проблемы следует принять во внимание, а какие — игнорировать. Каждый язык программирования предлагает инструменты создания полезных абстракций, и каждый успешно работающий программист знает, как использовать эти инструменты наилучшим образом.

Мы считаем абстракции настолько полезными, что буквально заполнили ими эту книгу. Безусловно, мы не называем их абстракциями в прямом смысле этого слова, поскольку они принимают различные формы. Мы оперируем такими понятиями, как функции, структуры данных, классы и наследование; ведь все это абстракции, которые мы не просто упоминаем здесь — мы их активно используем.

Если абстракции хорошо разработаны и удачно выбраны, то их можно использовать даже в том случае, когда вы не понимаете все их детали. Ведь нам необязательно быть автомеханиками, чтобы водить автомобиль, и точно так же нам не нужно понимать до конца работу C++-среды, чтобы использовать ее.

Охват материала

Если у вас серьезные намерения в отношении программирования на C++, вам просто необходимо прочесть эту книгу, несмотря на то что она не включает всего, что вам нужно знать.

Это утверждение вовсе не так уж парадоксально, как может показаться на первый взгляд. Ни одна книга такого объема не сможет уместить всей информации о C++, поскольку разным программистам для создания различных программ потребуются различные знания. Поэтому любая более полная книга о C++, например *Язык программирования C++* Бьярни Страуструпа (Addison-Wesley, 2000), при необходимости поможет вам уточнить интересующие детали.

С другой стороны, многие разделы C++ настолько важны (в универсальном смысле), что без их понимания просто невозможно написать эффективно работающие программы. Нашу книгу составляют именно такие разделы. Используя информацию только из этой книги, можно написать огромное множество программ. Один из наших рецензентов, ведущий программист по обслуживанию большой коммерческой системы, написанной на C++, сказал, что в этой книге описаны все основные средства, которые он использует в своей работе.

С помощью этих средств вы можете писать настоящие C++-программы, а не C++-программы в стиле C или любого другого языка. Освоив материал этой книги, вы поймете, *что еще* вам нужно узнать и как приступить к работе. У любителей создавать телескопы есть поговорка, что легче вначале сделать 3-дюймовый рефлектор, а затем 6-дюймовый, чем с самого начала приступить к созданию 6-дюймового отражателя.

Мы описываем только стандарт языка C++ и игнорируем всяческие расширения. Преимущество такого подхода в том, что программы, которые вы научитесь писать, будут работать везде. Кроме того, мы не уделяем внимание тому, как писать программы, предназначенные для работы в оконных средах, поскольку такие программы обязательно привязаны к конкретной среде. Постойте! Не спешите откладывать книгу в сторону! Поскольку наш подход универсален, в будущем вы сможете использовать все, что узнаете из этой книги, в *любых* средах. И потом, вы еще успеете прочитать, к примеру, о GUI-приложениях, но только *после того*, как освоите нашу книгу.

Несколько слов для опытных C- и C++-программистов

При изучении нового языка программирования часто приходится наблюдать, как многие пытаются писать программы в стиле языков, которые уже им знакомы. Наш подход позволяет избежать подобного, благодаря использованию абстракций высокого уровня из стандартной библиотеки C++ с самого начала обучения. Если вы имеете опыт работы на C или C++, то при таком подходе вас ожидает ряд хороших и не очень хороших новостей (причем это одни и те же новости).

Вероятно, новостью окажется для вас уже то, сколь небольшой уровень ваших знаний позволит понять язык C++ так, как он представлен в нашей книге. Возможно, на первых порах вам придется узнать больше, чем вы того ожидали (плохая новость), но вы освоите незнакомый материал быстрее, чем ожидали (хорошая новость). В частности, если вы уже знакомы с C++, то, скорее всего, вы сначала научились писать программы на C; это означает, что ваш стиль программирования построен на фундаменте языка C. В этом нет ничего дурного, но наш подход настолько отличается от упомянутого выше, что, как нам кажется, с нашей помощью вы увидите C++ с совершенно иной точки зрения, с которой вам еще не приходилось на него смотреть.

Конечно, многие детали синтаксиса вам будут уже знакомы, но ведь это всего лишь детали! Важные понятия языка представлены в книге, возможно, в совершенно неожиданном для вас порядке. Например, мы не касаемся массивов и указателей аж до главы 10 и даже не собираемся рассматривать такие любимцы многих программистов, как функции `printf` и `malloc`. Однако прямо в главе 1 мы рассматриваем библиотечный класс `string`. Заявляя о совершенно новом подходе в изучении C++, мы знаем, что говорим!

Структура книги

Нам кажется, что эту книгу удобно разделить на две части. В первой части (главы 0–7) акцент делается на программах, которые используют абстракции стандартной библиотеки. Во второй части (главы 8–16) речь идет об определении собственных абстракций.

Безусловно, такое раннее знакомство с библиотечными средствами может показаться неожиданным, но нам такое решение кажется правильным. Значительная часть синтаксиса языка C++ — особенно более трудные его разделы — обязана своим существованием авторам библиотеки. Пользователям библиотеки совсем не обязательно знать эти разделы языка. Игнорируя их до начала второй части книги, мы позволяем нашим читателям приступить к написанию C++-программ гораздо быстрее, чем при более традиционном подходе.

Если вы поймете, как использовать библиотеку, то будете готовы усвоить материал, посвященный средствам низкого уровня, на которых построена библиотека, и научитесь применять эти средства для написания собственных библиотек. Более того, вы начнете понимать, когда библиотека будет полезной, а когда стоит избегать написания нового библиотечного кода.

Хотя эта книга по объему меньше многих других книг о C++, мы все же попытались использовать в ней каждую важную идею, по крайней мере, дважды, а ключевые идеи — и того чаще. В результате многие части нашей книги ссылаются на другие части. Употребляя (и поясняя) впервые какой-нибудь важный термин, мы выделяем его ***полужирным курсивом***, чтобы обратить на него ваше внимание. Кроме того, ***так*** его будет легче отыскать в тексте, если вы захотите вернуться к нему, чтобы при необходимости перечитать приведенные пояснения.

Все главы (за исключением последней) завершаются разделами "Резюме". Эти разделы служат двум целям. Они позволяют вспомнить идеи, изложенные в данной главе, а также ознакомиться с дополнительным материалом, который, по нашему мнению, вам рано или поздно необходимо узнать. Мы предполагаем, что при первом чтении достаточно беглого ознакомления с этими разделами, но они вам еще сослужат хорошую службу, если вы захотите вернуться и освежить в памяти некоторые моменты.

В двух приложениях изложены важные аспекты языка и библиотеки на таком уровне детализации, который, мы надеемся, окажется для вас весьма полезным при написании собственных программ.

Как получить максимальную пользу от этой книги

Любая книга по программированию содержит примеры программ, и эта не является исключением. Чтобы понять, как работают эти программы, у вас нет другого выхода, как выполнить их на компьютере. Компьютерный парк постоянно обновляется, поэтому все, что мы могли бы сказать о нем сейчас, окажется неточным к тому времени, когда вы прочтете эти слова. Поэтому, если вы еще не знаете, как скомпилировать и выполнить C++-программу, обратитесь, пожалуйста, по адресу: `http://www.acceleratedcpp.com` и прочитайте все, что мы приготовили для вас. Время от времени мы будем обновлять этот Web-сайт, и на нем вы всегда сможете получить актуальную информацию и советы по выполнению C++-программ. На этом сайте вы также найдете "готовые к употреблению" версии некоторых примеров программ и другую информацию, которая может оказаться для вас интересной.

Благодарности

Мы хотели бы поблагодарить тех людей, без помощи которых создание этой книги было бы невозможно. Особо мы признательны рецензентам, которые внесли свой вклад в написание этой книги: Роберту Бергеру (Robert Berger), Дагу Бруку (Dag Brück), Адаму Бучсбауму (Adam Buchsbaum), Стефану Клеймеджу (Stephen Clamage), Джону Кэлбу (John Kalb), Джеффри Олдхэму (Jeffrey Oldham), Дэвиду Слейтону (David Slayton), Бьярни Страуструпу (Bjarne Stroustrup), Альберту Тенбушу (Albert Tenbusch), Брюсу Тителману (Bruce Tetelman) и Кловису Тондо (Clovis Tondo). В издании этой книги принимали участие многие сотрудники издательства Addison-Wesley, поэтому огромное спасибо Тиреллу Албаху (Tyrrell Albaugh), Банни Эймс (Bunny Ames), Майку Хендриксону (Mike Hendrickson), Деборе Лаферти (Deborah Lafferty), Кэси Охала (Cathy Ohala) и Симоне Пеймент (Simone Payment). Александр Цирис (Alexander Tsiris) проверил греческую этимологию, используемую в разделе 13.2.2. Наконец, идея начать книгу с создания высокоуровневых программ зрела в наших головах в течение многих лет, но окончательно сформировалась благодаря сотням студентов, которые посещали наши курсы, и тысячам желающих научиться программировать, которые слушали наши лекции.

Эндрю Коэнинг (Andrew Koenig)
Барбара Му (Barbara E. Moo)

Жилетт, Нью-Джерси
июнь 2000 года