УДК 004.4
ББК 32.973.26-018
С74

V. Anton Spraul
THINK LIKE A PROGRAMMER

Спрол, Антон.

С74 Думай как программист: креативный подход к созданию кода. С++ версия / Антон Спрол. — Москва : Эксмо, 2018. — 272 с. — (Мировой компьютерный бестселлер).

При помощи этой книги любой программист, особенно начинающий, может усовершенствовать свои навыки программирования. Автор разработал собственную программу, позволяющую получить навыки креативного решения разнообразных задач. Эти навыки необходимы в первую очередь тем, кто хочет создавать собственный код и действительно понимать и чувствовать основы программирования. Живой язык, множество примеров на языке С++ и уникальное авторское видение сделают чтение этой книги настоящим удовольствием.

УДК 004.4
ББК 32.973.26-018

Научно-популярное издание

МИРОВОЙ КОМПЬЮТЕРНЫЙ БЕСТСЕЛЛЕР

Антон Спрол

ДУМАЙ КАК ПРОГРАММИСТ
КРЕАТИВНЫЙ ПОДХОД К СОЗДАНИЮ КОДА
С++ ВЕРСИЯ

Директор редакции *Е. Капьёв*. Ответственный редактор *Е. Истомина*
Младший редактор *Е. Минина*. Художественный редактор *А. Гусев*

ООО «Издательство «Эксмо»
123308, Москва, ул. Зорге, д. 1. Тел.: 8 (495) 411-68-86.
Home page: **www.eksmo.ru** E-mail: **info@eksmo.ru**

Өндіруші: «ЭКСМО» АҚБ Баспасы, 123308, Маскеу, Ресей, Зорге көшесі, 1 үй.
Тел.: 8 (495) 411-68-86.
Home page: www.eksmo.ru E-mail: info@eksmo.ru.
Тауар белгісі: «Эксмо»
Қазақстан Республикасында дистрибьютор және өнім бойынша
арыз-талаптарды қабылдаушының
өкілі «РДЦ-Алматы» ЖШС, Алматы қ., Домбровский көш., 3«а», литер Б, офис 1.
Тел.: 8(727) 2 51 59 89,90,91,92, факс: 8 (727) 251 58 12 вн. 107; E-mail: RDC-Almaty@eksmo.kz
Өнімнің жарамдылық мерзімі шектелмеген.
Сертификация туралы ақпарат сайтта: www.eksmo.ru/certification

Сведения о подтверждении соответствия издания согласно законодательству РФ о техническом регулировании можно получить по адресу: http://eksmo.ru/certification/

Өндірген мемлекет: Ресей. Сертификация қарастырылмаған

Подписано в печать 18.12.2017. Формат 70x100$^{1}/_{16}$.
Печать офсетная. Усл. печ. л. 22,04.
Тираж экз. Заказ

ISBN 978-5-04-089838-1

ISBN 978-5-04-089838-1

Содержание

Об авторе

Антон Спрол преподавал введение в программирование и информатику более 15 лет. Эта книга представляет собой квинтэссенцию методов, которые он использовал и оттачивал на протяжении множества занятий с испытывающими сложности программистами.

ВВЕДЕНИЕ

Написание программ вызывает у вас сложность, несмотря на то, что вы, вроде бы, понимаете языки программирования? Вы киваете головой, читая главу какой-нибудь книги по программированию, но не можете применить на практике то, что прочитали? А может быть, вы понимаете листинг программы, увиденный в Интернете, так хорошо, что можете объяснить кому-то, что делает каждая из строк, но все равно смотрите в пустой экран редактора, когда сталкиваетесь с задачей по программированию, и ощущаете, что ваш мозг отказывается работать?

Вы не одиноки. Я преподаю программирование уже более 15 лет, и большинство моих студентов подошли бы под это описание в определенные моменты процесса обучения. Я бы назвал это *навыком решения задач,* способностью воспринять описание поставленной задачи

и написать оригинальный программный код для ее решения. Не все аспекты программирования требуют решения большого количества задач. Если вы просто вносите незначительные изменения в код рабочей программы, выполняете отладку, добавляете код тестирования, процесс программирования может быть настолько механическим по своей сути, что вашим творческим возможностям в итоге никогда не будет брошен вызов. Но факт в том, что все программы в тот или иной момент требуют решения задач, и все хорошие программисты умеют эти задачи решать.

Делать это непросто. Конечно, есть люди, справляющиеся с различными задачами легко и непринужденно. Их называют the naturals — эквивалент одаренного атлета, наподобие Майкла Джордана в мире программирования. Для этих избранных любые высокоуровневые идеи легко трансформируются в исходный код. Если воспользоваться метафорой языка Java, то получится, что их мозг выполняет программы «нативно», а все остальные, включая нас с вами, вынуждены запускать виртуальную машину, интерпретирующую код по мере выполнения.

Однако нужно понимать, что программист может не относится к «the naturals», и это не фатально. Иначе в нашем мире было бы очень мало программистов. Тем не менее я видел слишком много хороших учеников, отчаянно бьющихся над решением задач на протяжении слишком долгого времени. В худшем случае они отказывались от программирования вовсе, убежденные в собственной неспособности стать программистами и считая, что единственные хорошие программисты — это люди с природным даром.

Почему научиться решать задачи по программированию так сложно?

Отчасти из-за того, что решение задач — это деятельность, отличная от изучения синтаксиса языка программирования, чтения кода программ и запоминания элементов интерфейса приложений; вышеперечисленные процессы в основном аналитические, относящиеся к деятельности левого полушария. Написание оригинальной программы с использованием ранее изученных инструментов и навыков — это творческий процесс, относящийся к правому полушарию головного мозга.

Предположим, что вам нужно убрать ветку, попавшую в один из дождевых желобов вашего дома, однако стремянка недостаточно высока и вы не можете дотянуться до ветки. Вы направляетесь в гараж в поисках чего-то или сочетания чего-то с чем-то, что поможет вам убрать ветку из желоба. Есть ли что-то такое, что может увеличить длину стремянки, чтобы вы смогли схватить или сбить ветку? Возможно, вы можете просто взобраться на крышу и достать ветку оттуда? Вот это и называется решением задач, и это деятельность творческая. Хотите верьте, хотите нет, но при проектировании оригинальной программы мыслительный процесс у вас в голове похож на то, что происходит

в голове человека, пытающегося придумать, как убрать ветку из желоба, и довольно сильно отличается от мыслительного процесса человека, занятого отладкой существующего цикла for.

Впрочем, внимание авторов большинства книг по программированию сосредоточено на синтаксисе и семантике. Изучение таких норм любого языка необходимо, но это лишь первый шаг на пути изучения программирования. На самом деле, большинство книг по программированию для начинающих учат, как читать код программ, но не как его писать. Книги, которые сосредоточены на написании, зачастую представляют собой «кулинарные книги», обучающие конкретным «рецептам» и их использованию в определенных ситуациях. Такие книги могут быть очень ценными для экономии времени, но не для обучения написанию оригинального кода. Подумайте о кулинарных книгах в исходном значении этих слов. Несмотря на то, что у замечательных поваров есть кулинарные книги, никто из тех, кто полагается исключительно на кулинарные книги, не может быть по-настоящему хорошим поваром. Замечательный повар понимает ингредиенты, методы готовки и знает, как их сочетать для получения вкусных блюд. Все, что нужно такому повару — это кухня с нужными ингредиентами и оборудованием. Таким же образом замечательный программист понимает синтаксис языка, среду разработки приложения, алгоритмы, принципы разработки и знает, как все это сочетать для создания отличных приложений. Дайте такому программисту технические спецификации, развяжите ему руки полнофункциональной средой программирования — и случится нечто удивительное.

Современное обучение программированию, по большей части, не предлагает много руководств по решению задач. Наоборот, предполагается, что если программисту дан доступ ко всем инструментам и от него требуют написать большое количество программ, то рано или поздно он научится писать такие программы и будет делать это хорошо. Путь от инициации до просветления может быть полон отчаяния и растерянности — и слишком многие из тех, кто его начинает, не доходят до конца.

Вместо того чтобы учиться методом проб и ошибок, вы можете научиться решать задачи, используя системный подход. Именно этому и посвящена данная книга. Вы изучите техники организации ваших мыслей, процедуры поиска решений и стратегии, которые могут быть применены к определенным классам задач. Изучив эти подходы, вы сможете высвободить свой потенциал. Он неизвестен, и никто не может с точностью сказать, как работает творческое сознание. Однако если мы можем разучить музыкальное произведение, воспринять совет по творческому написанию текстов или научиться рисовать, значит, мы можем также научиться творчески решать задачи. Это книга не скажет вам, что именно нужно делать, она поможет вам развить скрытые способности решения задач, так, что вы будете

знать, как следует поступить. Предназначение этой книги — помочь вам стать таким программистом, каким вы должны стать.

Цель, которую я ставлю перед вами и перед каждым читателем этой книги, — научиться системному подходу к выполнению каждой задачи по программированию и быть уверенным в том, что в конце концов вы сможете решать любые из них. Когда вы закончите читать эту книгу, я хочу чтобы *вы думали как программист* и *верили в то, что вы программист.*

Об этой книге

Объяснив необходимость этой книги, я должен также прокомментировать то, чем эта книга является, а чем нет.

Предварительная подготовка

Эта книга предполагает, что вы уже знакомы с базовыми синтаксисом и семантикой языка C++ и уже начали писать программы. В большинстве глав предполагается, что вам уже известны конкретные основы C++ — они будут начинаться с обзора этих основ. Если вы только постигаете язык, не переживайте: существует много замечательных книг по синтаксису C++, и вы можете параллельно изучать решение проблем и задач и синтаксис. Просто убедитесь, что вы изучили нужный синтаксис прежде, чем попытаетесь одолеть задачи в главе.

Выбор тем

Темы, обсуждаемые мной в этой книге, представляют те аспекты, в которых, по моему опыту, у начинающих программистов больше всего проблем. Они также представляют широкий срез различных аспектов программирования начального и среднего этапов.

Я должен, однако, подчеркнуть, что это не «кулинарная книга» с готовыми алгоритмами и шаблонами для решения конкретных задач. Несмотря на то, что в последних главах книги обсуждается применение широко известных алгоритмов или шаблонов, вам не следует использовать эту книгу как «шпаргалку», которая поможет вам решить конкретные задачи или сконцентрироваться лишь на главах, имеющих непосредственное отношение к вашим текущим задачам. Вместо этого я бы посоветовал вам проработать всю книгу и пропускать тот или иной раздел только в том случае, если вам не хватает знаний для усвоения материала.

Стиль программирования

Небольшое замечание о стиле программирования в данной книге: эта книга не посвящена высокопроизводительному программированию или запуску наиболее компактного и эффективного кода. Выбранный мной стиль, в первую очередь, отвечает задаче удобочитаемости. В некоторых случаях для четкой иллюстрации описанного

правила я даже делаю несколько шагов, несмотря на то, что это же действие можно было выполнить за один шаг.

Некоторые вопросы стиля программирования будут описаны в этой книге, но они относятся к большим вопросам, как, например, что должно, а что не должно быть включено в класс, а не к маленьким вопросам, таким как отступы строк программного кода. Как развивающийся программист, вы, разумеется, захотите применить наиболее удобочитаемый стиль для всех выполняемых работ.

Упражнения

В эту книгу я включил большое количество упражнений по программированию. Но это не учебник, поэтому вы не найдете ответов к упражнениям в конце книги. Упражнения лишь дают вам возможность применить на практике подходы, описанные в главе. Выполнять упражнения или нет — решать вам, однако практическое применение этих подходов очень важно. Простое чтение книги не даст вам ничего. Помните, что эта книга не скажет вам, что именно нужно сделать в той или иной ситуации. Применяя техники, показанные в этой книге, вы самостоятельно поймете, что нужно делать. Более того, повышение вашей самооценки — еще одна из целей этой книги — подразумевает достижение успеха. На самом деле, когда вы поймете, что можете решить любую задачу в той или иной области, это будет означать, что вы проработали достаточное количество упражнений. В конце концов, выполнение упражнений по программированию должно приносить вам удовольствие. Выполнение этих задач должно быть не рутиной, но вызовом вашим возможностям.

Воспринимайте эту книгу, словно полосу препятствий в вашей голове. Прохождение полосы препятствий увеличивает вашу силу, выносливость, ловкость и дает вам уверенность в собственных способностях. По мере прочтения этой книги и применения описанных в ней подходов на практике вы разовьете уверенность в себе и приобретете навыки решения задач, которые сможете использовать в любой ситуации при программировании. Встретив сложную задачу в будущем, вы будете знать подход к ее решению.

Почему C++?

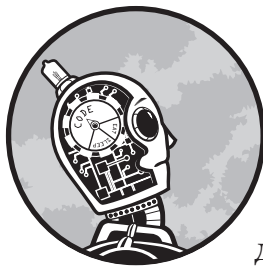
Программный код примеров в данной книге написан на языке C++. При этом следует отметить, что книга посвящена решению проблем, не обязательно характерных для этого языка. В книге вы не найдете большого количества советов и приемов, подходящих только для C++. Общие концепции, которым она обучает, могут быть реализованы с помощью любого языка программирования. Тем не менее мы не можем обсуждать программирование, не затрагивая программы, поэтому для примеров я выбрал конкретный язык для примеров.

C++ был выбран по многим причинам. Во-первых, этот язык популярен во многих проблемных областях. Во-вторых, так как C++

произошел от строго процедурного языка C, код на C++ может быть написан с использованием как процедурной, так и объектно ориентированной парадигмы. Объектно ориентированное программирование применяется сегодня настолько повсеместно, что не может быть исключено из обсуждения решения задач. Однако многие фундаментальные подходы к решению задач могут быть реализованы в строго процедурной парадигме, что упрощает как написание, так и изменение кода. В-третьих, будучи низкоуровневым языком с высокоуровневыми библиотеками, C++ позволяет мне обсудить оба уровня программирования. При необходимости лучшие программисты могут объединять низкоуровневые части с высокоуровневыми библиотеками и компонентами для сокращения времени разработки. Наконец, C++ — это отличный выбор, так как, научившись решать задачи на этом языке, вы сможете в принципе решать задачи на любом другом языке программирования. Изучение одного языка программирования облегчает последующее изучение других. Это особенно характерно для языка C++, так как он довольно сложен и основан на двух парадигмах одновременно. Возможно, это может вас испугать. Но преуспев в C++, в сможете не просто писать программы — вы по-настоящему станете программистом.

1

Стратегии решения задач



Эта книга посвящена решению задач, но что это значит на самом деле? Когда люди используют этот термин в разговоре, они зачастую имеют в виду нечто очень отличное от того, о чем я говорю в этой книге. Если из выхлопной трубы вашего старого автомобиля идет сизый дым, двигатель то и дело глохнет, а расход топлива вырос, то эта проблема, как задача, может быть решена с применением знаний автомобиля, диагностикой и заменой деталей при помощи инструментов автомеханика. Впрочем, если вы расскажете о своей проблеме друзьям, то один из них может вам ответить: «Эй, ты должен заменить свою старую машину на что-то поновее, и проблема будет решена». Однако предложение вашего друга на самом деле не будет *решением задачи*: это будет лишь способом ее *избежать*.

Задачи подразумевают ограничения, незыблемые правила, касающиеся задачи или способа ее решения. В случае со сломанной машиной одно из ограничений заключается в том, что вам нужно отремонтировать имеющийся автомобиль, а не купить новый. Ограничения также должны подразумевать общую стоимость ремонта, время на ремонт и невозможность приобретения инструмента для ремонта только этой поломки.

При решении проблемы в программе у вас также есть определенные ограничения. Среди общих ограничений язык программирования, платформа (будет ли программа работать на ПК, iPhone или ином устройстве?), производительность (для игровой программы может требоваться обновление графики до 30 кадров в секунду, бизнес-приложение должно иметь ограничение по максимальному времени отклика на ввод пользователя) или объем требуемой памяти. Иногда ограничения также подразумевают код, на который вы можете ссылаться: возможно, программа не может включать определенный открытый код или, наоборот, должна использовать только открытый код.

Таким образом, термин *решение задач* для программистов я могу определить как написание оригинальной программы, выполняющей определенный набор заданий и соответствующей всем установленным ограничениям.

Начинающие программисты зачастую так хотят выполнить первую часть определения — написать программу, выполняющую определенное задание — что забывают про выполнение второй части определения — соответствовать установленным ограничениям. Я называю такую программу, выдающую правильный результат, но нарушающую одно или несколько заявленных правил, *«Кобаяси Мару»*. Если это название вам незнакомо, значит вы не смотрели культовый в узких кругах кинофильм *«Звездный Путь 2: Ярость Хана»*. В этом фильме есть сюжетная линия про тест для начинающих офицеров Академии Звездного флота. Кадетов помещают на мостик симулятора звездного корабля или поручают роль капитана, выполняющего миссию, в ходе которой они встречаются с невозможным выбором. Гибель грозит людям на подбитом корабле *«Кобаяси Мару»*, однако, чтобы добраться до них, вам нужно пойти на самоубийственное сражение с врагами. Цель этого упражнения — проверить смелость и отвагу кадета, когда он находится под огнем. В данном сражении победить невозможно, и любой выбор ведет к провалу. Ближе к концу фильма мы обнаруживаем, что капитан Кирк модифицировал симулятор, сделав победу возможной. Кирк был хитер, но он не решил дилемму *«Кобаяси Мару»*, а лишь избежал ее.

К счастью, задачи, с которыми вы столкнетесь как программист, разрешимы, но многие программисты все еще прибегают к подходу Кирка. В некоторых случаях они делают это случайно. («О, черт! Мое решение работает только в том случае, если элементов данных

сто или меньше. Оно должен работать для неограниченного набора данных. Мне придется переосмыслить это решение».) В других случаях ограничения изменяют преднамеренно, прибегая к этому как к уловке, чтобы успеть к сроку, установленному начальником или преподавателем. В других случаях программист просто не знает, как соответствовать всем ограничениям. В худших случаях, которые я видел, студенты-программисты платили третьим лицам за написание программы. Независимо от мотивов, мы всегда должны проявлять максимум усердия во избежание «Кобаяси Мару».

Классические головоломки

По мере чтения книги вы заметите, что, несмотря на изменение частностей исходного кода от одной проблемной области к другой, некоторые схемы в моих подходах останутся неизменными. И это хорошо, потому что в конце концов позволит нам уверенно подходить к решению любой задачи, вне зависимости от того, есть ли у вас большой опыт в той или иной проблемной области. Эксперты в решении задач могут быстро распознать *аналогию*, т. е. пригодное для использования сходство между решенной и нерешенной задачей. Если мы распознаем, что некое свойство задачи А аналогично некоему свойству задачи Б, при условии, что мы уже решили задачу Б, у нас будет очень ценная наработка для решения задачи А.

В этом разделе мы обсудим классические задачи не из мира программирования, из которых мы можем вынести уроки, применимые при решении задач по программированию.

Лисица, гусь и кукуруза

Первая классическая задача, которую мы обсудим, — это загадка про крестьянина, которому нужно пересечь реку. Возможно, вы уже встречались с этой задачей в той или иной форме.

Задача: как пересечь реку?

Крестьянин с лисицей, гусем и мешком кукурузы должен пересечь реку. У крестьянина есть лодка, однако в ней есть место только для самого крестьянина и одного из перечисленных объектов. К сожалению, и лисица, и гусь голодны. Лисица не может быть оставлена наедине с гусем, так как она съест гуся. Точно так же гусь не может остаться наедине с мешком кукурузы, иначе он склюет кукурузу. Как крестьянину перевезти животных и кукурузу через реку?

Условие задачи проиллюстрировано на рис. 1.1. Если вы никогда ранее не встречались с этой задачей, сделайте небольшую паузу и потратьте несколько минут на попытки решить ее. Если же вы слышали эту загадку ранее, попытайтесь вспомнить решение и нашли ли вы его самостоятельно.