Предисловие

Приходилось ли вам когда-нибудь

тратить уйму времени на программирование неправильного алгоритма?
пользоваться структурой данных, сложность которой превосходила всякое понимание?
пропускать ошибку, лежащую на поверхности, даже при самом тщательном тестировании программы?
возиться целый день с исправлением ошибки, на которую следовало бы потратить пять минут?
дорабатывать программу так, чтобы она работала втрое быстрее и при этом потребляла меньше памяти?
в муках переносить программу с рабочей станции на персональный компьютер или наоборот?
вносить "косметические" изменения в программы, написанные кем-то другим?
переписывать программу начисто, потому что она была совершенно неудобочитаема?

Скорее всего, вам это не понравилось. Что ж, подобное случается с программистами постоянно. Решение этих проблем часто оказывается более трудоемким, чем следовало бы, потому что такие темы, как тестирование, отладка, переносимость, быстродействие, стиль, проектирование программ — т.е. *практика программирования*, — вовсе не являются центральными в вузовских курсах программирования и вычислительной техники. Большинству программистов приходится осваивать все это хаотически, по мере накопления опыта, а некоторым из них вообще не судьба этому научиться.

В мире объемных и изощренных интерфейсов, постоянно изменяющихся языков, сред и утилит разработки, под давлением необходимости осваивать все новые и новые знания нетрудно потерять основной ориентир — такие базовые принципы, как простоту, ясность, общность, которые образуют фундамент для разработки высококачественных программ. Увы, людям свойственно переоценивать роль стандартов и автоматизированных сред программирования, которые механизируют значительную часть разработки программ, так что компьютер в известной степени становится сам себе программистом.

Подход, принятый нами в этой книге, основан именно на глубоких и взаимосвязанных принципах, применимых в программировании любого уровня. Эти принципы следующие. *Простота* означает, что программа должна быть короткой и легкой для доработки; *ясность* состоит в том, что ни у программиста при чтении програм-

мы, ни у компьютера при ее выполнении не должно возникать сомнений и ощущения двусмысленности; *общность* заключается в способности программы правильно реагировать на широкий диапазон ситуаций и легко адаптироваться к новой обстановке; наконец, *автоматизация* (ради которой, собственно, и пишутся программы) призвана освободить человека от трудоемких и однообразных операций. Рассматривая искусство программирования на самых разных языках и в широкой предметной области — от алгоритмов и структур данных до архитектуры, отладки, тестирования и повышения быстродействия, — можно проиллюстрировать универсальные инженерные концепции, независимые от конкретного языка, операционной системы или среды программирования.

Эта книга возникла как результат многолетнего опыта в области разработки и доработки самого разного программного обеспечения, преподавания курсов программирования и вычислительной техники, а также работы с огромным количеством программистов. Нам хочется поделиться усвоенными практическими уроками, предложить целый ряд методов и техник, которые бы помогли программистам всех уровней повысить свою квалификацию и производительность труда.

Мы пишем для нескольких категорий читателей. Если вы — студент, уже прошли несколько предметов из курса программирования и вычислительной техники и хотите лучше научиться программировать практические задачи, то в этой книге вы найдете более подробное изложение некоторых тем, недостаточно освещенных в учебной программе. Если вы пишете программы по роду своей деятельности, но скорее в качестве вспомогательных средств, а не основных продуктов производства, то приведенные в книге сведения помогут вам программировать на этом уровне более эффективно. Предлагаемый материал окажется полезным также в том случае, если вы — профессиональный программист, но в свое время не получили достаточных знаний по тем или иным вопросам из программы вуза (или же хотели бы освежить их в памяти). Наконец, немало интересного найдет здесь руководитель коллектива по разработке программного обеспечения, желающий научить своих подчиненных работать лучше.

Надеемся, что изучение материала этой книги пойдет на пользу вашим программам. Единственное требование к подготовке читателя — это некоторый опыт программирования, лучше всего на языках C, C++ или Java. Конечно, чем больше опыта, тем лучше; никто и ничто не может сделать специалиста из новичка за две-три недели. Программисты для систем Unix и Linux найдут в наших примерах больше знакомых мест и характерных черт, чем те, кто работал только в средах Windows и Macintosh. Тем не менее узнать что-то новое из книги и тем самым облегчить свою работу сможет практически каждый читатель, в какой бы среде он ни программировал.

Материал книги подразделяется на девять глав, каждая из которых посвящена одному большому и важному аспекту практического программирования.

В главе 1 рассматривается стиль программирования. Хороший стиль настолько важен для успешной работы программиста, что мы решили поставить его на первое место в книге. Хорошо написанные программы работают лучше, чем написанные плохо, — в них меньше ошибок, и их легче исправлять или отлаживать. Поэтому о хорошем стиле нужно помнить с самого начала. В этой же главе сделано введение в

такую важную тему, как использование устойчивых конструкций, характерных для того или иного языка.

Глава 2 посвящена алгоритмам и структурам данных, которые традиционно составляют основу вузовских учебных планов по программированию и вычислительной технике. Поскольку большинство читателей наверняка знакомы с этой тематикой, в нашем изложении мы дадим лишь краткий обзор основных алгоритмов и структур данных, используемых практически во всех программах. Более сложные конструкции обычно строятся их этих элементарных "кирпичиков", поэтому их знание имеет основополагающий характер.

В главе 3 рассматриваются архитектура и реализация небольшой программы, которая иллюстрирует вопросы алгоритмизации и структуризации данных в реалистической манере, приближенной к практической работе. Эта программа написана на пяти языках. Сравнение различных ее версий позволяет оценить, как одни и те же структуры данных реализованы в каждом из языков и как отличаются выразительность и эффективность аналогичных конструкций в различных средах программирования.

Способы взаимодействия (интерфейсы) пользователя, программ и отдельных частей программ составляют фундамент программирования, так что успех той или иной программы в значительной мере определяется тем, насколько хорошо спроектированы и реализованы ее интерфейсы. В главе 4 рассказывается о том, как эволюционировала небольшая библиотека для синтаксического анализа популярного формата данных. Хотя пример и невелик, в нем иллюстрируются многие важные вопросы проектирования программ: абстрагирование, сокрытие данных, управление ресурсами, обработка ошибок.

Как бы мы ни старались написать программу безошибочно с первой же попытки, все же возникновение ошибок в ней неизбежно. В главе 5 излагаются общая стратегия и тактические приемы, помогающие отлаживать программы и исправлять ошибки более систематически и эффективно. Среди рассмотренных в этой главе тем — формы проявления типичных ошибок и анализ статистики, позволяющий выявить источник проблемы по некоторым типичным шаблонам в отладочных выходных данных.

Тестирование представляет собой попытку дать достаточное доказательство того, что программа работает правильно и остается правильной по мере ее доработки. Поэтому в главе 6 мы сосредоточимся на вопросах тестирования — как выполняемого вручную, так и автоматизированного. Например, потенциально опасные или слабые места в программах можно обнаружить проверкой граничных условий. С помощью различных средств автоматизации (наподобие специально написанных для этого программ или сценариев) можно легко выполнить большие объемы тестов сравнительно небольшими усилиями. Совершенно другая типичная разновидность тестирования, предназначенная для выявления других видов ошибок, называется *стрессовым тестированием* и также рассматривается в этой главе.

Компьютеры стали такими быстрыми, а компиляторы — настолько качественными, что многие программы вполне удовлетворяют требованиям к их быстродействию с первого же дня их работы. Но бывают и программы, работающие слишком медленно или потребляющие слишком много памяти, а некоторые из них отличаются сразу обоими этими недостатками. В главе 7 представлен систематический подход к эффективной организации ресурсов, используемых программой, который позволяет

сохранить правильность и рациональность организации программы в ходе оптимизации ее работы.

В главе 8 рассматриваются вопросы переносимости. Действительно успешные программы существуют достаточно долго, так что их выполняющая среда успевает неоднократно измениться. Может также возникнуть необходимость переноса таких программ на новую аппаратную платформу, в новую операционную или языковую среду. Хорошая переносимость программы в этом контексте означает минимум труда и времени, которые необходимо затратить на ее приспособление к новой среде.

В программировании существует множество языков, причем не только языков общего назначения, используемых для реализации разнообразных универсальных задач, но и узкоспециализированных, предназначенных для ограниченной области приложений. В главе 9 демонстрируется важность знаковых систем или систем обозначений, которые можно использовать для упрощения программ, структуризации стоящих перед программистом задач и даже для написания программ, которые сами способны генерировать другие программы.

Говоря о программировании, поневоле приходится демонстрировать большие объемы кода. Большинство примеров кода было написано специально для этой книги, и лишь некоторые небольшие фрагменты заимствованы из других источников. Мы приложили все усилия к тому, чтобы этот код был качественным, и протестировали его в пяти-шести системных средах непосредственно в электронной форме. Более подробную информацию об этом можно найти на Web-сайте, посвященном данной книге:

```
http://tpop.awl.com
```

Большая часть приведенных примеров написана на языке C; имеется некоторое количество кода на C++ и Java, а также совсем небольшие фрагменты на языках разработки сценариев. На самом низком уровне организации кода языки C и C++ почти ничем не отличаются, так что программы на C являются в то же время и программами на C++. Языки C++ и Java являются прямыми потомками C, поэтому в значительной мере совпадают с ним по синтаксису и предлагают столь же эффективные и выразительные средства, в то же время обладая более развитыми наборами типов и библиотек. В нашей повседневной работе мы широко используем все три языка, а также множество других. Выбор языка зависит от конкретной задачи: операционные системы лучше всего писать на скоростном высокоэффективном языке, лишенном всяких ограничений, таком как C или C++; короткие одноразовые задачки прекрасно решаются с помощью языков разработки сценариев или командных интерпретаторов наподобие Awk или Perl; для реализации пользовательских интерфейсов практически нет равных языкам Visual Basic, Tcl/Tk и Java.

Выбор языка для примеров представляет собой нетривиальную методическую задачу. Точно так же, как ни один язык не может справиться со всеми задачами одинаково хорошо, нельзя и проиллюстрировать все темы программирования на одном и том же языке с одинаковой выразительностью. Языки высокого уровня непосредственно диктуют программисту ряд технических решений. На более низком уровне необходимо выбирать между альтернативными подходами к решению. Учитывая больше деталей, можно обсудить вопрос более подробно. Опыт показывает, что даже если мы пользуемся только средствами языков высокого уровня, всегда бывает

полезно знать, как они соотносятся со средствами более низкого уровня. Не обладая этими знаниями, легко наткнуться на непреодолимый барьер в виде непонятного поведения программы или необъяснимого падения быстродействия. Поэтому для наших примеров мы часто выбирали C, хотя на практике предпочли бы какой-нибудь другой язык.

Тем не менее в основном наше изложение можно считать независимым от конкретного языка программирования. Выбор структуры данных диктуется тем языком, который есть "под рукой"; одни языки не предоставляют почти никакого выбора, тогда как другие предлагают широкий круг альтернатив. Но критерии, в соответствии с которыми делается выбор, должны оставаться одними и теми же. Так, мелкие подробности процедур тестирования или отладки в разных языках сильно отличаются, а вот стратегия и тактические приемы практически одинаковы. Большая часть способов и рекомендаций, как сделать программу эффективнее, одинаково применима во всех языках.

На каком бы языке ни писал программист, его задача состоит в том, чтобы выжать максимальный эффект из имеющихся у него в наличии средств. Хороший программист способен перебороть недостатки как неуклюжего языка, так и неповоротливой операционной системы. В то же время даже самая лучшая среда программирования не спасет плохого программиста. Мы надеемся, что эта книга поможет вам программировать лучше и получать больше удовольствия от этой работы независимо от нынешнего уровня вашей квалификации.

Мы хотели бы выразить искреннюю благодарность нашим друзьям и коллегам, которые прочитали рукопись книги и высказали множество ценных советов и пожеланий. Джон Бентли (Jon Bentley), Расс Кокс (Russ Cox), Джон Лакош (John Lakos), Джон Линдерман (John Linderman), Питер Мемишьян (Peter Memishian), Ян Лэнс Тейлор (Ian Lance Taylor), Ховард Трики (Howard Trickey) и Крис Ван Вик (Chris Van Wyk) прочли рукопись, причем неоднократно, с необыкновенной тщательностью и дотошностью. Мы испытываем глубокую признательность к таким людям, как Том Карджилл (Tom Cargill), Крис Клиленд (Chris Cleeland), Стив Дьюхерст (Steve Dewhurst), Эрик Гросс (Eric Grosse), Эндрю Херрон (Andrew Herron), Джерард Хольцман (Gerard Holzmann), Даг Макилрой (Doug McIlroy), Пол Макнами (Paul McNamee), Питер Нельсон (Peter Nelson), Деннис Ритчи (Dennis Ritchie), Рич Стивенс (Rich Stevens), Том Шиманski (Tom Szymanski), Кентаро Тояма (Kentaro Toyama), Джон Уэйт (John Wait), Дэниел Ван (Daniel C. Wang), Питер Вайнбергер (Peter Weinberger), Маргарет Райт (Margaret Wright) и Клифф Янг (Cliff Young), за их бесценные комментарии на различных стадиях производства книги. Мы также благодарим за добрый совет и вдумчивые предложения таких лиц, как Эл Ахо (Al Aho), Кен Арнольд (Ken Arnold), Чак Бигелоу (Chuck Bigelow), Джошуа Блох (Joshua Bloch), Билл Кафрэн (Bill Coughran), Боб Фландрена (Bob Flandrena), Рене Франш (Renée French), Марк Керниган (Mark Kernighan), Энди Кёниг (Andy Koenig), Сэйп Мюллендер (Sape Mullender), Эви Немет (Evi Nemeth), Марти Рабинович (Marty Rabinowitz), Марк Шэйни (Mark V. Shaney), Бьорн Страуструп (Bjarne Stroustrup), Кен Томпсон (Ken Thompson) и Фил Уодлер (Phil Wadler). Спасибо всем вам!

Брайан Керниган

Роб Пайк

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:	115419, Москва, а/я 783
Украины:	03150, Киев, а/я 152