Обращение к читателям

“Если карта не соответствует местности,
доверяй местности.”

Швейцарская армейская поговорка

Эта глава содержит разнообразную информацию; ее цель — дать представление о том, что можно ожидать от остальной части книги. Пожалуйста, пролистайте ее и прочитайте то, что найдете интересным. Для преподавателей полезной будет б*о*льшая часть книги. Если же вы читаете книгу без помощи хорошего преподавателя, то не пытайтесь прочитать и понять все, что написано в этой главе; просто взгляните на раздел "Структура книги” и первую часть раздела “Педагогические принципы”. Возможно, вы захотите вернуться и перечитать эту главу еще раз, когда научитесь писать и выполнять свои собственные программы.

В этой главе...

0.1. Структура книги

Книга состоит из четырех частей и нескольких приложений.

- В части I, “Основы”, описаны фундаментальные концепции и методы программирования на примере языка C++ и библиотек, необходимых для начала разработки программ. К этим концепциям относятся система типов, арифметические операции, управляющие конструкции, обработка ошибок, а также разработка, реализация и использование функций и пользовательских типов.
- В части II, “Ввод и вывод”, описаны способы ввода числовых и текстовых данных с клавиатуры и из файлов, а также вывода результатов на экран и в файлы. Кроме того, в ней показано, как вывести числа, текст и геометрические фигуры в виде графической информации, а также как ввести данные в программу с помощью графического пользовательского интерфейса (GUI).
- Часть III, “Данные и алгоритмы”, посвящена контейнерам и алгоритмам из стандартной библиотеки C++ (standard template library — STL). В ней продемонстрирована реализация и использование контейнеров (таких как **vector**, **list** и **map**) с помощью указателей, массивов, динамической памяти, исключений и шаблонов. Кроме того, описаны разработка и использование алгоритмов из стандартной библиотеки (таких как **sort**, **find** и **inner_product**).
- Часть IV, “Расширение кругозора”, посвящена изложению идей и истории программирования на примерах матричных вычислений, обработки текста, тестирования, а также встроенных систем управления на основе языка C.
- *Приложения* содержат полезную информацию, которая была пропущена в тексте по причинам педагогического характера. В частности, приводится краткий обзор языка C++ и возможностей стандартной библиотеки, а также продемонстрированы принципы работы с интегрированными средами разработки (integrated development environment — IDE) и библиотекой графического пользовательского интерфейса (graphical user interface — GUI).

К сожалению, программирование нельзя так просто разделить на четыре четко разделенные области. По этой причине предложенная классификация является довольно грубой, хотя мы считаем ее полезной (иначе не стали бы ее предлагать). Например, операции ввода в книге используются намного раньше детального описания стандартных потоков ввода-вывода в языке C++. Как только для описания какой-то идеи нам требуется упомянуть несколько тем, мы предпочитаем изложить минимум информации, а не отсылать читателя к подробному изложению темы в другом месте. Строгая классификация больше нужна для справочников, чем для учебников.

Порядок изложения определяется методами программирования, а не языковыми конструкциями (см. раздел 0.2). Обзор свойств языка содержится в приложении A.


Для облегчения работы читателей, впервые читающих книгу и еще не знающих, какая информация является действительно важной, мы используем три вида пиктограмм, которые должны привлечь внимание.


- Метка: концепции и методы (как в данном разделе).


- Метка: совет.


- Метка: предупреждение.

0.1.1. Общие принципы

В книге я обращаюсь к вам непосредственно. Это проще и понятнее, чем принятое в научных работах косвенное обращение в третьем лице. Под местоимением “вы” я подразумеваю вас, читатель, а под местоимением “мы” — себя и преподавателей или нас с вами, работающих вместе над решением задачи, как если бы мы сами находились в одной комнате.


Эту книгу следует читать главу за главой от начала до конца. Довольно часто у вас будет появляться желание вернуться в какое-то место и перечитать его во второй или в третий раз. На самом деле это единственное разумное поведение, так как со временем некоторые детали стираются в памяти. В таких случаях вы обязательно рано или поздно постараетесь их освежить. Однако, несмотря на предметный указатель и перекрестные ссылки, это не та книга, которую можно открыть на любой странице и начинать читать, рассчитывая на успех. Каждый раздел и каждая глава требуют от вас твердого знания материала, изложенного в предыдущих разделах и главах.

Каждая глава является вполне самодостаточной единицей, т.е. ее можно прочесть за один присест (что, конечно, не всегда возможно из-за напряженного расписания занятий). Это один из основных критериев разделения текста на главы. Кроме того, каждая глава содержит упражнения и задачи, а также посвящена конкретной концепции, идее или методу. Некоторые главы получились слишком длинными, поэтому не следует понимать выражение “за один присест” слишком буквально.

В частности, поразмышляв над контрольными вопросами, разобрав примеры и выполнив несколько упражнений, вы почти наверняка поймете, что вам следует еще раз перечитать какие-то разделы, и на это может уйти несколько дней. Мы объединили главы в части, посвященные основным темам, например вводу-выводу. Эти части удобны для проведения контрольных опросов.

Об учебниках часто говорят: "Он ответил на все мои вопросы сразу, как только я о них подумал!" Это типично для простых вопросов, и первые читатели рукописи этой книги заметили это. Однако этот принцип не может быть всеобщим. Мы поднимаем вопросы, которые новичку вообще не могут прийти в голову. Наша цель — поставить вопросы, необходимые для написания качественных программ, предназначенных для других людей, и ответить на них. Научить задавать правильные (часто сложные) вопросы необходимо для того, чтобы студент стал думать как программист. Задавать простые и очевидные вопросы очень удобно, но это не поможет стать программистом.

Мы стараемся уважать ваш интеллект и учитываем затраты вашего времени. В изложении мы ценим профессионализм, а не красивости, поэтому некоторые вещи недоговариваем, а не разжевываем. Мы стараемся не преувеличивать важность методов программирования или языковых конструкций, но не следует также недооценивать такие простые утверждения, как, например: "Это свойство часто оказывается полезным". Если мы подчеркиваем, что некий материал является важным, то это значит, что рано или поздно вы потеряете много дней, если не освоите его. Мы шутим намного меньше, чем хотели бы, но опыт показывает, что у людей совершенно разное чувство юмора и попытки шутить могут лишь запутать изложение.

Мы не претендуем на то, что наши идеи или инструменты идеальны. Ни один инструмент, ни одна библиотека и ни один метод не может решить все проблемы, возникающие у программиста. В лучшем случае они помогут разработать и реализовать ваше решение. Мы очень старались избегать "святой лжи", т.е. отказались от упрощенных объяснений, которые легко и просто понять, но которые на самом деле неверны в контексте реальных языков и задач. С другой стороны, эта книга — не справочник; более точное и полное описание языка C++ изложено в книге Страуструп Б. *Язык программирования C++*. — М.; СПб. — "Издательство БИНОМ" – "Невский диалект", 2001. — 1099 с., и в стандарте ISO C++ .

0.1.2. Упражнения, задачи и т.п.

Программирование — это не просто интеллектуальная деятельность, поэтому для овладения этим искусством необходимо писать программы. Мы предлагаем два уровня практического программирования.

- *Задания*. Простые задачи, предназначенные для отработки практических, почти механических навыков. Задания обычно подразумевают последовательность модификаций простой программы. Вы должны выполнить каждое задание.

Задания не требуют глубокого понимания, ума или инициативы. Мы рассматриваем их как очень важную часть книги. Если вы не выполните задания, то не поймете материал, изложенный в книге.

- *Упражнения*. Одни упражнения тривиальны, другие очень сложны, но большинство из них предназначено для того, чтобы разбудить у вас инициативу и соображение. Если вы серьезный человек, то выполните хотя бы несколько упражнений. Попробуйте это сделать хотя бы для того, чтобы понять, насколько это трудно для вас. Затем выполните еще несколько упражнений. Так постепенно вы справитесь с большинством из них. Эти упражнения требуют не столько выдающих умственных способностей, сколько изобретательности. Однако мы надеемся, что они достаточно трудны, чтобы стимулировать ваше самолюбие и занять все ваше свободное время. Мы не рассчитываем, что вы решите все задачи, но советуем попытаться

Кроме того, рекомендуем каждому студенту принять участие в разработке небольшого проекта (или крупного, если будет время). Эти проекты предназначены для того, чтобы написать законченную полезную программу. В идеале проекты должны создаваться небольшими группами разработчиков (например, тремя программистами), работающих вместе около месяца и осваивающих главы части III. Большинство студентов получают удовольствие именно от работы над проектом, который связывает людей друг с другом. Одни люди предпочтут отложить книгу в сторону и решать задачи, еще не дойдя до конца главы; другие захотят дочитать до конца и лишь затем приступить к программированию. Для того чтобы поддержать студентов, желающих программировать сразу, мы предлагаем простые практические задания, которые озаглавлены “**Попробуйте**”. Эти задания являются естественными составными частями книги. По существу, эти задания относятся к упражнениям, но сфокусированы на узкой теме, которая изложена перед их формулировкой. Если вы пропустите это задание — например, потому, что поблизости нет компьютера или вас слишком увлекло чтение книги, — вернитесь к нему, когда начнете разбирать упражнения; задания “**Попробуйте**” либо являются частью упражнений, либо дополняют их. В конце каждой главы вы найдете контрольные вопросы. Они предназначены для закрепления основных идей, объясняемых в главе. Эти вопросы можно рассматривать как дополнения к задачам. В то время как задачи посвящены практическим аспектам программирования, контрольные вопросы позволяют сформулировать идеи и концепции. Этим они напоминают интервью.

Раздел “Термины” в конце каждой главы представляет собой часть словаря по программированию и языку C++. Если хотите понимать, что люди говорят о программировании, и свободно выражать свои собственные идеи, вам следует знать значение этих слов.

Повторенье — мать ученья. Идеальный студент должен повторить каждую важную идею как минимум дважды, а затем закрепить ее с помощью упражнений.

0.1.3. Что потом?

Станете ли вы профессиональным программистом или экспертом по языку C++, прочитав эту книгу? Конечно, нет! Настоящее программирование — это тонкое, глубокое и очень сложное искусство, требующее знаний и технических навыков. Рассчитывать на то, что за четыре месяца вы станете экспертом по программированию, можно с таким же успехом, как и на то, что за полгода или даже год вы полностью изучите биологию, математику или иностранный язык (например, китайский, английский или датский), или научитесь играть на виолончели. Если подходить к изучению книги серьезно, то можно ожидать, что вы сможете писать простые полезные программы, читать более сложные программы и получите хорошие теоретическую и практическую основы для дальнейшей работы.

Прослушав этот курс, лучше всего поработать над реальным проектом. Еще лучше параллельно с работой над реальным проектом приступить к чтению какой-нибудь книги профессионального уровня (например, Bjarne Stroustrup, *The C++ Programming Language, Special Edition* (Addison-Wesley, 2000), более специализированной книги, связанной с вашим проектом (например, документация по библиотеке Qt для разработки графического пользовательского интерфейса GUI, или справочник по библиотеке ACE для параллельного программирования, или учебник, посвященный конкретному аспекту языка C++, например Кёниг Э., Му Б. *Эффективное программирование на C++*. — М.: Издательский дом "Вильямс", 2002. — 384 с.; Саттер Г. *Решение сложных задач на C++*. — М.: Изд-во Вильямс, 2002. — 400 с.; Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. *Приемы объектно-ориентированного проектирования. Паттерны проектирования*. — Спб.: Питер, 2004. — 366 с.)[1]. Полный список рекомендуемых книг приведен в разделе 0.6 и в разделе "Библиография" в конце книги.

В конце концов, можете приступить к изучению другого языка программирования. Невозможно стать профессионалом по программному обеспечению, даже если программирование не является вашей основной специальностью, зная только один язык программирования.

0.2. Педагогические принципы

Чему мы хотим вас научить и как собираемся организовать процесс обучения? Мы попытались изложить минимальный объем концепций, методов и инструментов, необходимых для эффективного программирования. Их список приведен ниже.

- Организация программ.
- Отладка и тестирование.
- Разработка классов.
- Вычисления.

[1] Приведены русскоязычные переводы рекомендуемых автором книг. — *Примеч. ред.*

- Разработка функций и алгоритмов.
- Графика (только двумерная).
- Графические пользовательские интерфейсы.
- Обработка текста.
- Сопоставление регулярных выражений.
- Файлы и потоки ввода-выводы (I/O).
- Управление памятью.
- Научные/числовые/инженерные вычисления.
- Принципы проектирования и программирования.
- Стандартная библиотека языка C++.
- Стратегии разработки программного обеспечения.
- Приемы программирования на языке C.

Эти темы охватывают процедурное программирование (его типичным представителем является язык C), а также абстракцию данных, объектно-ориентированное и обобщенное программирование. Основным предметом книги является именно *программирование*, т.е. идеи, методы и средства выражения этих идей с помощью программ. Нашим основным инструментом является язык C++, поэтому мы довольно подробно описываем его многочисленные возможности. Однако следует помнить, что язык C++ — это просто инструмент, а не основной предмет изучения этой книги. Иначе говоря, книга посвящена программированию с помощью языка C++, а не языку C++ с небольшим количеством теории.

Каждая тема, которую мы разбираем, преследует две цели: описать метод, концепцию или принцип, а также практический язык программирования или свойство библиотеки. Например, для иллюстрации классов и концепции наследования мы используем систему двумерной графики. Это позволит сэкономить место (и ваше время), а также продемонстрировать, что программирование не сводится к простому связыванию фрагментов кода друг с другом, чтобы как можно быстрее получить результат. Основным источником таких “примеров двойного назначения” является стандартная библиотека языка C++. Некоторые из этих примеров имеют даже тройное назначение. Например, мы рассматриваем класс **vector** из стандартной библиотеки, используем его для иллюстрации полезных методов проектирования и демонстрируем многочисленные приемы программирования, позволяющие его реализовать. Одна из целей — показать, как реализованы основные возможности библиотеки и как они отражаются на аппаратном обеспечении. Мы настаиваем на том, что профессионал должен понимать устройство инструментов, с помощью которых он работает, а не считать их волшебной палочкой.

Одни темы покажутся некоторым программистам более интересными, чем другие. Однако мы советуем не предвосхищать свои потребности (как вы можете знать,

что вам понадобится в будущем?) и хотя бы просмотреть каждую главу. Если вы используете книгу как учебник, а не самоучитель, то ваш преподаватель сам определит выбор глав.

Наш подход можно назвать глубинным, конкретным или концептуальным. Вначале, в главах 1–11, мы быстро (ну, хорошо, относительно быстро) описываем набор навыков, необходимых для написания небольших практических программ. При этом мы описываем много инструментов и приемов, не вдаваясь в детали. Мы акцентируем внимание на простых конкретных программах, поскольку конкретное усваивается быстрее, чем абстрактное. Большинство людей используют именно такой способ обучения. Не рассчитывайте на то, что уже на ранних стадиях обучения вы поймете все до малейших деталей. В частности, пытаясь сделать что-то, отличающееся от того, что только что работало, вы обнаружите "загадочные" явления. Впрочем, попытайтесь! И, пожалуйста, не забывайте выполнять упражнения и решать задачи, которые мы предлагаем. Помните, что на первых порах у вас просто еще нет достаточно знаний и опыта, чтобы понять, что является простым, а что сложным; выявляйте недостатки и учитесь на них.

☑ Первый этап мы пройдем в быстром темпе. Мы хотим как можно быстрее достичь пункта, после которого вы сможете писать свои собственные интересные программы. Некоторые говорят: "Мы должны двигаться медленно и осторожно; прежде чем научиться бегать, мы должны научиться ходить!" Но где вы видели ребенка, который учился бы именно ходить, а не бегать? На самом дели дети бегают, пока не научатся контролировать свою скорость. Точно так же мы сначала быстренько, иногда ненадолго останавливаясь, научимся программировать, а уж потом притормозим, чтобы глубже разобраться и понять, как все это работает. Мы должны научиться бегать раньше, чем ходить!

☒ Ни в коем случае не следует зацикливаться на попытках досконально изучить какие-то детали языка или метода. Разумеется, вы можете заучить все встроенные типы данных в языке C++ и все правила их использования. Конечно, после этого вы можете чувствовать себя знатоком. Однако это не сделает вас программистом. Пренебрежение деталями может вызвать у вас ощущение недостатка знаний, но это быстрейший способ, позволяющий научиться писать хорошие программы. Обратите внимание на то, что именно наш подход, по существу, используется при обучении детей иностранным языкам. Если вы зайдете в тупик, советуем искать помощи у преподавателей, друзей, коллег и т.п. Не забывайте, что в первых главах нет ничего принципиально сложного. Однако многое будет незнакомым и поэтому может показаться сложным.

Позднее мы углубим ваши первоначальные навыки, чтобы расширить базу ваших знаний и опыта. Для иллюстрации концепций программирования мы используем упражнения и задачи.

Основной упор в книге делается на идеи и причины. Людям нужны идеи, чтобы решать практические задачи, т.е. находить правильные и принципиальные решения. Необходимо понимать подоплеку этих идей, т.е. знать, почему именно этими, а не другими принципами следует руководствоваться, а также чем это может помочь программистам и пользователям программ. Никого не может удовлетворить объяснение “потому что потому”. Кроме того, понимание идей и причин позволит вам обобщить их в новых ситуациях и комбинировать принципы и средства для решения новых задач. Знание причин является важной частью программистских навыков. И наоборот, формальное знание многочисленных плохо понятых правил и конструкций языка программирования является источником многих ошибок и приводит к колоссальной потере времени. Мы ценим ваше время и не хотим его тратить понапрасну.

Многие технические детали языка C++ изложены в приложениях и справочниках, где их можно при необходимости найти. Мы считаем, что вы способны самостоятельно найти заинтересовавшую вас информацию. Используйте для этого предметный указатель и содержание. Не забывайте также об Интернете. Однако помните, что не каждой веб-странице следует слепо доверять. Многие веб-сайты, выглядящие авторитетными источниками знаний, созданы новичками или просто пытаются что-то кому-то продать. Некоторые веб-сайты просто устарели. Мы собрали коллекцию полезных ссылок и фактов на нашем веб-сайте **`www.stroustrup.com/Programming`**.

Пожалуйста, не придирайтесь к “реалистичности” примеров. Идеальный пример — это максимально короткая и простая программа, ярко иллюстрирующая свойство языка, концепцию или прием. Большинство реальных примеров являются намного более запутанными, чем наши, и не содержат необходимых комбинаций идей, которые мы хотели бы продемонстрировать. Успешные коммерческие программы, содержащие сотни тысяч строк, основаны на технических приемах, которые можно проиллюстрировать дюжиной программ длиной по 50 строк. Самый быстрый способ понять реальную программу сводится к хорошему знанию ее теоретических основ.

С другой стороны, мы не используем для иллюстрации своих идей красивые примеры с симпатичными животными. Наша цель — научить вас писать реальные программы, которые будут использоваться реальными людьми. По этой причине каждый пример, не относящийся к технической стороне языка программирования, взят из реальной жизни. Мы стараемся обращаться к читателям как профессионалы к будущим профессионалам.

0.2.1. Порядок изложения

Существует множество способов обучения программированию. Совершенно очевидно, что мы не придерживаемся популярного принципа “способ, которым я научился программировать, является наилучшим способом обучения”. Для облегчения процесса обучения мы сначала излагаем темы, которые еще несколько лет назад

считались сложными. Мы стремились к тому, чтобы излагаемые темы вытекали из поставленных задач и плавно переходили одна в другую по мере повышения уровня ваших знаний. По этой причине книга больше похожа на повествование, а не на словарь или справочник.

Невозможно одновременно изучить все принципы, методы и свойства языка, необходимые для создания программ. Следовательно, необходимо выбрать подмножество принципов, методов и свойств, с которых следует начинать обучение. В целом любой учебник должен вести студентов через набор таких подмножеств. Мы понимаем свою ответственность за выбор этих тем. Поскольку невозможно охватить все темы, на каждом этапе обучения мы должны выбирать; тем не менее то, что останется за рамками нашего внимания, не менее важно, чем то, что будет включено в курс.

Для контраста, возможно, будет полезно привести список подходов, которые мы отвергли.

- "*Сначала следует изучить язык C*". Этот подход к изучению языка C++ приводит к ненужной потере времени и приучает студентов к неправильному стилю программирования, вынуждая их решать задачи, имея в своем распоряжении ограниченный набор средств, конструкций и библиотек. Язык C++ предусматривает более строгую проверку типов, чем язык C, а стандартная библиотека лучше соответствует потребностям новичков и позволяет применять исключения для обработки ошибок.
- "*Снизу-вверх*". Этот подход отвлекает от изучения хороших и эффективных стилей программирования. Вынуждая студентов решать проблемы, ограничиваясь совершенно недостаточными языковыми конструкциями и библиотеками, он приучает их к плохим и слишком затратным способам программирования.
- "*Если вы что-то описываете, то должны делать это исчерпывающим образом*". Этот подход подразумевает изложение по принципу "снизу-вверх" (заставляя студентов все глубже и глубже погружаться в технические детали). В результате новички тонут в море технических подробностей, на изучение которых им потребуются годы. Если вы умеете программировать, то техническую информацию найдете в справочниках. Документация хороша сама по себе, но совершенно не подходит для первоначального изучения концепций.
- "*Сверху-вниз*". Этот подход, предусматривающий переход от формулировки принципа к его техническим подробностям, отвлекает читателей от практических аспектов программирования и заставляет концентрироваться на высокоуровневых концепциях еще до того, как они поймут, зачем они нужны. Например, никто просто не в состоянии правильно оценить принципы разработки программного обеспечения, пока не поймет, как легко делать ошибки и как трудно их исправлять.

- "*Сначала следует изучать абстракции*". Фокусируясь лишь на основных принципах и защищая студентов от ужасной реальности, этот подход может вызвать у них пренебрежение реальными ограничениями, связанными с практическими задачами, языками программирования, инструментами и аппаратным обеспечением. Довольно часто этот подход поддерживается искусственными "учебными языками", которые в дальнейшем нигде не используются и (вольно или невольно) дезинформируют студентов о проблемах, связанных с аппаратным обеспечением и компьютерными системами.
- "*Сначала следует изучить принципы разработки программного обеспечения*". Этот подход и подход "сначала следует изучить абстракции" порождают те же проблемы, что и подход "сверху-вниз": без конкретных примеров и практического опыта, вы просто не сможете оценить важность абстракций и правильного выбора методов разработки программного обеспечения.
- "*С первого дня следует изучать объектно-ориентированное программирование*". Объектно-ориентированное программирование — один из лучших методов организации программ, но это не единственный эффективный способ программирования. В частности, мы считаем, что сначала необходимо изучить типы данных и алгоритмы и лишь потом переходить к разработке классов и их иерархий. Мы с первого дня используем пользовательские типы (то, что некоторые люди называют объектами), но не углубляемся в устройство класса до главы 6 и не демонстрируем иерархию классов до главы 12.
- "*Просто верьте в манию*". Этот подход основан на демонстрации мощных инструментов и методов без углубления в технические подробности. Он заставляет студентов угадывать — как правило, неправильно, — что же происходит в программе, с какими затратами это связано и где это можно применить. В результате студент выбирает лишь знакомые ему шаблоны, что мешает дальнейшему обучению.

Естественно, мы вовсе не имеем в виду, что все эти подходы совершенно бесполезны. Фактически мы даже используем некоторые из них при изложении некоторых тем. Однако в целом мы отвергаем их как общий способ обучения программированию, полезному для реального мира, и предлагаем альтернативу: конкретное и глубокое обучение с упором на концепции и методы.

0.2.2. Программирование и языки программирования

В первую очередь мы учим программированию, а выбранный язык программирования рассматриваем лишь как вспомогательное средство. Выбранный нами способ обучения может опираться на любой универсальный язык программирования. Наша главная цель — помочь вам понять основные концепции, принципы и методы. Однако эту цель нельзя рассматривать изолированно.

Например, языки программирования отличаются друг от друга деталями синтаксиса, возможностями непосредственного выражения разных идей, а также средствами технической поддержки. Тем не менее многие фундаментальные методы разработки безошибочных программ, например простых и логичных программ (главы 5-6), выявления инвариантов (раздел 9.4.3) и отделения интерфейса от реализации (разделы 9.7 и 14.1–14.2), во всех языках программирования практически одинаковы.

Методы программирования и проектирования следует изучать на основе определенного языка программирования. Проектирование, программирование и отладка не относятся к навыкам, которыми можно овладеть абстрактно. Вы должны писать программы на каком-то языке и приобретать практический опыт. Это значит, что вы должны изучить основы какого-то языка программирования. Мы говорим “основы”, так как времена, когда все основные промышленные языки программирования можно было изучить за несколько недель, ушли в прошлое. Для обучения мы выбрали подмножество языка C++, которое лучше всего подходит для разработки хороших программ. Кроме того, мы описываем свойства языка C++, которые невозможно не упомянуть, поскольку они либо необходимы для логической полноты, либо широко используются в сообществе программистов.

0.2.3. Переносимость

Как правило, программы на языке C++ выполняются на разнообразных компьютерах. Основные приложения на языке C++ выполняются на компьютерах, о которых мы даже представления не имеем! По этой причине мы считаем переносимость программ и возможность их выполнения на компьютерах с разной архитектурой и операционными системами одним из самых важных свойств. Практически каждый пример в этой книге не только соответствует стандарту ISO Standard C++, но и обладает переносимостью. Если это не указано явно, представленные в книге программы могут быть выполнены с помощью любого компилятора языка C++ и были протестированы на разных компьютерах и под управлением разных операционных систем.

Процесс компилирования, редактирования связей и выполнения программ на языке C++ зависит от операционной системы. Было бы слишком неудобно постоянно описывать детали устройства этих систем и компиляторов каждый раз при ссылке на выполнение программы. Наиболее важная информация, необходимая для использования интегрированной среды разработки программ Visual Studio и компилятора Microsoft C++ под управлением операционной системы Windows, приведена в приложении B.

Если вы испытываете трудности при работе с популярными, но слишком сложными интегрированными средами разработки программ, предлагаем использовать командную строку; это удивительно просто. Например, для того чтобы скомпилировать, отредактировать связи и выполнить простую программу, состоящую из двух

исходных файлов, **my_file1.cpp** и **my_file2.cpp**, с помощью компилятора GNU C++ **g++** под управлением операционной системы Unix или Linux, выполните две команды:

```
g++ -o my_program my_file1.cpp my_file2.cpp
my_program
```

Да, этого достаточно.

0.3. Программирование и компьютерные науки

Можно ли свести компьютерные науки к программированию? Разумеется, нет! Единственная причина, по которой мы поставили этот вопрос, заключается в том, что люди часто заблуждаются по этому поводу. Мы затрагиваем множество тем, связанных с компьютерными науками, например алгоритмы и структуры данных, но наша цель — научить программировать, т.е. разрабатывать и выполнять программы. Это изложение и шире, и уже, чем общепринятая точка зрения на компьютерные науки.

- *Шире*, так как программирование связано с множеством технических знаний, которые, как правило. не относятся ни к какой научной дисциплине.
- *Уже*, т.е. мы не стремились к систематическому изложению основ компьютерных наук.

Цель этой книги — частично охватить курс компьютерных наук (если вы собираетесь стать специалистом в этой области), изложить введение в методы разработки и эксплуатации программного обеспечения (если вы планируете стать программистом или разработчиком программного обеспечения) и, вообще, заложить основы более общего курса.

Тем не менее, несмотря на то, что изложение опирается на компьютерные науки и их основные принципы, следует подчеркнуть, что мы рассматриваем программирование как совокупность практических навыков, основанных на теории и опыте, а не как науку.

0.4. Творческое начало и решение задач

Основная цель книги — помочь вам выражать свои идеи в программах, а не научить придумывать эти идеи. Кроме того, мы приводим множество примеров решения задач, как правило, с помощью анализа, за которым следует последовательное уточнение решения. Мы считаем, что программирование само по себе является формой решения задач: только полностью поняв задачу и ее решение, можно написать правильную программу; и только через конструирование и тестирование программ можно прийти к полному пониманию задачи. Таким образом, программирование является неотъемлемой частью процесса познания. Однако мы стараемся продемонстрировать это на примерах, а не путем “проповеди” или подробного описания процесса решения задач.

0.5. Обратная связь

Идеальных учебников не существует; потребности разных людей очень отличаются друг от друга. Однако мы старались сделать эту книгу и сопровождающие ее материалы как можно лучше. Для этого нам необходима обратная связь; хороший учебник невозможно написать в изоляции от читателей. Пожалуйста, сообщите нам об ошибках, опечатках, неясных местах, пропущенных объяснениях и т.п. Мы также будем благодарны за постановку более интересных задач, за формулировку более ярких примеров, за предложения тем, которые следует удалить или добавить, и т.д. Конструктивные комментарии помогут будущим читателям. Все найденные ошибки будут опубликованы на веб-сайте **`www.stroustrup.com/Programming`**.

0.6. Библиографические ссылки

Кроме публикаций, упомянутых в главе, ниже приведен список работ, которые могут оказаться полезными.

Austern, Matthew H. Generic Programming and the STL: Using and Extending the C++ Standard Template Library. Addison-Wesley, 1999. ISBN 0201309564.

Austern, Matthew H. (editor). “Technical Report on C++ Standard Library Extensions.” ISO/IEC PDTR 19768.

Blanchette, Jasmin, and Mark Summerfield. *C++ GUI Programming with Qt 4*. Prentice Hall, 2006. ISBN 0131872493.

Gamma, Erich, Richard Helm, Ralph Johnson, and John M. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley, 1994. ISBN 0201633612.

Goldthwaite, Lois (editor). “Technical Report on C++ Performance.” ISO/IEC PDTR 18015.

Koenig, Andrew (editor). *The C++ Standard.* ISO/IEC 14882:2002. Wiley, 2003. ISBN 0470846747.

Koenig, Andrew, and Barbara Moo. *Accelerated C++: Practical Programming by Example.* Addison-Wesley, 2000. ISBN 020170353X.

Langer, Angelika, and Klaus Kreft. *Standard C++ IOStreams and Locales: Advanced Programmer’s Guide and Reference.* Addison-Wesley, 2000. ISBN 0201183951.

Meyers, Scott. *Effective STL: 50 Specific Ways to Improve Your Use of the Standard Template Library.* Addison-Wesley, 2001. ISBN 0201749625.

Meyers, Scott. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs (3rd Edition).* Addison-Wesley, 2005. ISBN 0321334876.

Schmidt, Douglas C., and Stephen D. Huston. *C++ Network Programming, Volume 1: Mastering Complexity with ACE and Patterns.* Addison-Wesley, 2002. ISBN 0201604647.

Schmidt, Douglas C., and Stephen D. Huston. *C++ Network Programming, Volume 2: Systematic Reuse with ACE and Frameworks.* Addison-Wesley, 2003. ISBN 0201795256.

Stroustrup, Bjarne. *The Design and Evolution of C++*. Addison-Wesley, 1994. ISBN 0201543303.

Stroustrup, Bjarne. “Learning Standard C++ as a New Language.” *C/C++ Users Journal,* May 1999.

Stroustrup, Bjarne. *The C++ Programming Language (Special Edition).* Addison-Wesley, 2000. ISBN 0201700735.

Stroustrup, Bjarne. “C and C++: Siblings”; “C and C++: A Case for Compatibility”; and “C and C++: Case Studies in Compatibility.” *C/C++ Users Journal,* July, Aug., Sept. 2002.

Sutter, Herb. *Exceptional C++: 47 Engineering Puzzles, Programming Problems, and Solutions.* Addison-Wesley, 2000. ISBN 0201615622.

Более полный список библиографических ссылок приведен в конце книги.

0.7. Биографии

Вы можете вполне резонно спросить: “Кто эти люди, которые собираются нас учить программировать?” Для того чтобы вы поняли это, мы приводим некоторую биографическую информацию. Я, Бьярне Страуструп, написал эту книгу и вместе с Лоуренсом “Питом” Петерсеном на ее основе разработал университетский вводный курс программирования.

Бьярне Страуструп

Я разработал и впервые реализовал язык программирования C++. В течение последних тридцати лет я использовал этот и многие другие языки программирования для решения многочисленных задач. Я люблю элегантные и эффективные программы, предназначенные для сложных приложений, таких как управление роботами, графические системы, игры, анализ текста и компьютерные сети. Я учил проектированию программирования и языку C++ людей с разными способностями и интересами. Кроме того, я являюсь основателем и членом комитета ISO по стандартизации языка C++, в котором возглавляю рабочую группу по эволюции языка.

Это моя первая книга, представляющая собой вводный курс. Мои другие книги, такие как “Язык программирования C++” и “Дизайн и эволюция C++”, предназначены для опытных программистов.

Я родился в рабочей семье в Архусе, Дания, и получил магистерскую степень по математике и компьютерным наукам в местном университете. Докторскую степень по компьютерным наукам я получил в Кембридже, Англия. Более двадцати пяти лет я работал в компании AT&T, сначала в знаменитом Исследовательском компьютерном центре лабораторий Белла (Computer Science Research Center of Bell

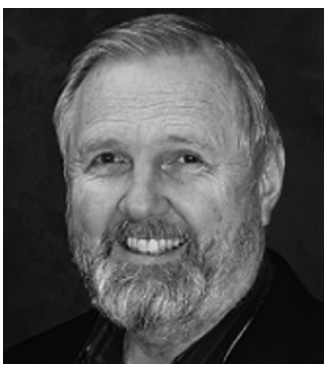
Labs) — именно там были изобретены операционная система Unix, языки C и C++, а также многое другое, — а позднее в подразделении AT&T Labs–Research.

Я являюсь членом Национальной технической академии США (U.S. National Academy of Engineering), Ассоциации по вычислительной технике (Association for Computing Machinery — ACM), Института инженеров по электротехнике и электронике (Institute of Electrical and Electronics Engineers — IEEE), а также сотрудником компаний Bell Laboratories и AT&T. Я был первым специалистом по компьютерным наукам, получившим в 2005 году премию Уильяма Проктера за научные достижения (William Procter Prize for Scientific Achievement), которую присуждает научное общество Sigma Xi.

Работа не занимает все мое время. Я женат, у меня двое детей. Один из них стал врачом, а другой учится в аспирантуре. Я читаю много книг (исторические повести, фантастику, детективы и труды по дипломатии) и люблю музыку (классику, рок, блюз и кантри). Застолья с друзьями составляют существенную часть моей жизни. Я люблю посещать интересные места по всему миру. Для того чтобы застолья проходили без последствий, я бегаю трусцой.

За дальнейшей информацией обращайтесь на мои персональные страницы **`www.research.att.com/~bs`** и **`www.cs.tamu.edu/people/faculty/bs`**. В частности, там вы узнаете, как правильно произносится мое имя[2].

Лоуренс “Пит” Петерсен



В конце 2006 года Пит представлялся так: “Я — учитель. Почти двадцать лет я преподаю языки программирования в Техасском университете агрокультуры и машиностроения (Texas A&M). Студенты пять раз выдвигали меня на присуждение премий за успехи в преподавании (Teaching Excellence Awards), и в 1996 году я получил премию за достижения в преподавании (Distinguished Teaching Award) от ассоциации выпускников Технического колледжа (Alumni Association for the College of Engineering). Я участвую в программе усовершенствования преподавания (Wakonse Program for Teaching Excellence), а также являюсь членом Академии усовершенствования учителей (Academy for Educator Development).

---

[2] На веб–странице **`http://www.research.att.com/~bs/bs_faq.html`** автор очень подробно объясняет, что его норвежское имя правильно произносится как Беарне или, в крайнем случае, Бьярне, а не Бьорн и не Бьёрн, а фамилия читается как Стровструп, а не Страуструп. Однако по историческим причинам мы придерживаемся принятой в русскоязычной литературе транскрипции. В этом нет ничего необычного. Было бы странно, руководствуясь формальными рассуждениями, переделывать фамилии Эйлер на Ойлер, Эйнштейн на Айнштайн и т.д. — *Примеч. ред.*

Будучи сыном офицера, я легок на подъем. Получив степень по философии в университете Вашингтона (University of Washington), я двадцать два года прослужил в армии полевым артиллерийским офицером и аналитиком-исследователем по опытной эксплуатации. С 1971-го по 1973-й год я прошел Высшие курсы полевых артиллерийских офицеров в Форт-Силле, Оклахома (Field Artillery Officer's Advanced Course at Fort Sill, Oklahoma). В 1979 я помог организовать Учебный центр офицеров-испытателей и с 1978-го по 1981-й год и с 1985-го по 1989-й год работал ведущим преподавателем на девяти разных должностях в разных регионах США.

В 1991 году я создал небольшую компанию, разрабатывавшую программное обеспечение для университетов вплоть до 1999 года. Мои интересы сосредоточены в области преподавания, проектирования и разработки программного обеспечения, предназначенного для реальных людей. Я получил магистерскую степень по техническим наукам в Технологическом институте штата Джорджия (Georgia Tech), а также магистерскую степень по педагогике в Техасском университете агрокультуры и машиностроения. Я также прошел программу подготовки магистров по микрокомпьютерам. Моя докторская диссертация по информатике и управлению написана в Техасском университете агрокультуры и машиностроения.

С женой Барбарой мы живем в г. Брайан, штат Техас. Я люблю путешествовать, ухаживать за садом и принимать гостей. Мы стараемся проводить как можно больше времени с нашими сыновьями и их семьями, особенно с внуками Ангелиной, Карлосом, Тесс, Эйвери, Николасом и Джорданом.

К несчастью, в 2007 году Пит умер от рака легкого. Без него этот курс никогда не достиг бы успеха.

Послесловие

Большинство глав завершается коротким постскриптумом, в котором излагается определенная точка зрения на предшествующую главу. Мы сделали это, хотя понимали, что она может ошеломить читателей (и часто на самом деле приводит их в замешательство) и что полностью уяснить ее можно, лишь выполнив упражнения и прочитав следующие главы (в которых будут применяться указанные идеи). Не паникуйте, расслабьтесь. Это вполне естественно и понятно. Вы не можете стать экспертом за один день, но, проработав книгу, можете стать вполне компетентным программистом. Кроме того, вы найдете в книге много фактов, примеров и приемов, которые многие программисты считают чрезвычайно интересными и поучительными.