深入 C++ 系列

Applied C++
Practical Techniques for Building Better Software

Applied C++ 中文版
——构建更佳软件的实用技术

[美] Philip Romanik, Amy Muntz 著
陈学峰 杨健康 林琪 译

包括书中项目全部源代码
和多种软件

PEARSON
Addison Wesley
Pearson Education
中国电力出版社
www.infopower.com.cn

深入 C++ 系列

Applied C++
Practical Techniques for Building Better Software

Applied C++ 中文版
——构建更佳软件的实用技术

[美] Philip Romanik, Amy Muntz 著
陈学峰 杨健康 林琪 译

中国电力出版社
www.infopower.com.cn

Applied C++: practical techniques for building better software (ISBN 0-321-10894-9)
Philip Romanik & Amy Muntz

图书在版编目（CIP）数据
Applied C++中文版 /（美）曼特兹著；陈学峰 杨健康 林琪译. 北京：中国电力出版社，2004
（深入 C++系列）
ISBN 7-5083-2185-5
Ⅰ.A... Ⅱ.①曼...②陈...③杨...④林... Ⅲ.C 语言－程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字（2004）第 022598 号

丛 书 名：深入C++系列
书　　名：Applied C++中文版
编　　著：（美）Philip Romanik & Amy Muntz
翻　　译：陈学峰　杨健康　林琪
责任编辑：牛贵华
出版发行：中国电力出版社
地址：北京市三里河路6号　邮政编码：100044
电话：（010）88515918　传　真：（010）88518169
印　　刷：北京丰源印刷厂
开　　本：787×1092　1/16　印　张：18.5　字　数：418千字
书　　号：ISBN 7-5083-2185-5
版　　次：2004 年 5 月北京第 1 版　2004 年 5 月第 1 次印刷
定　　价：35.00 元（1CD）

前 言

本书将介绍如何应用 C++来解决开发商业软件时所固有的问题。参加过复杂软件开发小组的读者，对于这里提到的所谓商业软件的含义，应当有非常确切的理解。

商业软件（commercial software）要交付给用户（可以是内部用户，也可以是外部用户），用户就要依赖于你所提供的接口。商业软件可能置于嵌入式系统中，也可能是一个软件库或面向标准平台的应用程序。无论最终在什么环境运行，软件都必须在特定的时间发布，并且具备在市场中取得成功的所有要素。软件往往会由一个工程小组进行开发，但由另外一些工程师来对其进行扩展和维护。这些对软件进行扩展和维护的工程师很可能不是最初的开发小组成员，他们可能必须从网站听取客户的建议，并按照客户的要求为软件增加新的功能，或者纠正一些问题。

软件工程最大的挑战之一是组织一个工程小组来开发一个复杂的软件，而且要按照要求准时完成全部功能以交付使用。不过还存在另一个更为艰巨的挑战，即在该软件开发的同时，还要能够将其移交给其他的工程小组进行扩展和维护。本书所汇集的 C++技术和实践技巧正是要满足这一要求，而且这些技术和技巧也已经得到了反复使用。对于许多情况，我们对理想解决方案和实际解决方案进行了比较。在此提供了对这两者加以权衡的讨论，以便你做出更全面的决定，而且在选择某个方法（而不是另一些方法）时，我们还给出了相应的指导原则。至于在你的应用中采用哪种方法最佳则要由你自己来决定。我们在开发商业软件中积累了一些实践技巧，这些技巧的使用使我们的软件更加成功，而本书的目的就是与你分享这些经验，希望能对你有所帮助。

如果你习惯于通过阅读代码来学习，那么你会发现本书中提供了大量的示例，供你学习之用。我们使用了一个贯穿全书的例子，以此来介绍所有相关技术。由于我们编写本书的想法来源于开发图像软件的经验，所以尽管 C++技术适用于任何技术领域，但书中的例子仍大多取自于图像处理领域。

本书以一个缩略图生成器的简单应用程序（尽管并不完备）作为起点。在原型阶段我们将使用这个应用来验证不同的 C++设计和实现技术。这个应用非常简单，因此很易于理解，采用不同 C++技术的效果也比较明显，因此很适合用来建立原型。

在这个简单的缩略图生成器中，存在一些固有问题，书中最后的图像框架程序将对这些问题加以解决。该应用程序具有如下特性：

- 内存占用大（memory intensive）。图像处理需要高效地使用内存，因为图像可能会十分庞大且难于处理。内存管理对于整个应用程序的性能起着十分关键的作用。
- 性能要求高（performance intensive）。虽然生成缩略图是一种简单易懂的图像处理技

术，但在书中后面所介绍的技术（例如边界锐化和降低噪声）却需要通过周密细致的设计才能真正可用。能有好的图像函数来处理数字图像当然很不错，但如果这些函数需要耗费很长的运行时间，那就没有什么实际用处了。

完成了本书的学习后，你将得到一个可以用来处理你自己的数字图像的图像处理框架，以及一个实用的 C++应用工具包。此图像处理框架提供了高效的图像存储和内存使用功能，还提供了处理数字图像的例程（例如边界锐化、图像大小调整、降低噪声、边界检测、图像缩减等等）、第三方软件的接口以及很多用来优化性能的工具。该框架是一个很有用的软件，它具备实用的设计和实现特性，因此甚至可以将它作为开发商业软件产品的基础。

随书附带的光盘中包含：缩略图生成器、原型以及最终的图像框架的完整源代码。可以从以下站点获取相应的所有更新版本：http://www.appliedcpp.com。

适用读者

本书假设你对 C++很熟悉，这样当我们应用了该语言中的一些构造时，你会觉得似曾相识或曾经用过。同时假设你曾经开发过一些应用（可以是供个人使用，也可以是用于商业用途），而且熟悉标准模板库（Standard Template Library，STL）所提供的功能。我们非常希望能够与你一起细致地探讨一些特定 C++构造的优缺点。最后，希望你真心喜欢具体的代码示例，因为代码示例在本书中比比皆是。

本书不是一本 C++语言的参考书，不过我们也提供了一些入门知识，并对某些语法要求很严格或不常使用的内容做了相应介绍。关于 C++语言的基础，我们推荐《The C++ Programming Language, Special Edition》[Stroustrup00]；如果要深入讨论具体的 C++构造，例如引用计数，则可以参考《Effective C++, Second Edition》[①][Meyers98]；关于 STL 的内容，我们推荐《Effective STL》[②][Meyers01]；至于如何使用 C++模板，我们建议你阅读《C++ Templates》[③][Vandevoorde03]以了解有关内容。

至于我们所选择的数字图像领域，在此并不要求你开发过任何处理图像的软件。书中提供了一些内容可供你复习有关图像的基本知识，如果你对图像处理相当熟悉，则可以跳过这一部分。当谈到任何应用于图像的特定操作时，在介绍具体的代码示例前，我们都会给出一个简单的解释以及操作前后图像的对照。如果你想更深入、精确地讨论图像处理操作，我们推荐《Digital Image Processing》[Pratt01]一书。

如何使用本书

我们希望你能连续地阅读这本书，因为在第 2 章介绍了一个具体示例，并将由此开发出第

① 本书英文影印版《Effective C++（影印版）》已由中国电力出版社出版。——编者注
② 本书英文影印版《Effective STL（影印版）》已由中国电力出版社出版。——编者注
③ 本书英文影印版《C++Templates（影印版）》已由中国电力出版社出版。——编者注

5 章所示的最终图像框架设计。在整本书中，我们在标题以及每章第一页的概要框中都突出强调了将要探讨的 C++技术。

这本书的组织结构如下。

第 1 章，绪论。对于我们写这本书所要达到的目标，这一章将给出一个总体上的介绍，另外，对于我们所推荐的 C++技术，在此将揭示使用这些技术的背景以及一些个人喜好。我们还针对数字图像处理专门提供了一节可供选择的内容。如果你有开发图像应用的经验，那么可以跳过这一章的最后一节。

第 2 章，一个测试应用。我们将介绍一个简单而且尚不完备的应用，用于测试原型 C++技术。之所以选择这个简单应用，是因为它能够有效地展示对不同的设计和实现进行取舍时所涉及的权衡问题。

第 3 章，设计技术。从这一章开始 C++设计的讨论。在此用大量的代码示例来说明 C++的设计策略，由于在这本书中模板的应用相当多，因此这里提供了模板的入门知识。最后，我们为设计的各个方面建立原型，并建立了支持设计所需的通用实用程序。

第 4 章，设计考虑。在此探索设计遵循的指导原则以及设计中可能用到的其他策略。我们提供了一系列的编码原则、重用策略以及一个简单但很有效的调试策略。

第 5 章，系统考虑。我们将分析系统级的设计问题，例如多线程和多进程设计、异常处理（包括我们提供的一个异常处理的框架）、编译时和运行时问题、模板特殊化以及应用的国际化等问题。

第 6 章，实现考虑。这一章将我们所研究的 C++技术应用到图像框架各个部分的最终设计和实现中。另外，本章还介绍了一些全局图像处理函数，例如边界的锐化和噪声降低。我们介绍了这些技术及其 C++实现在视觉上产生的总体效果。本章还提供了一个库和其他第三方软件的高级接口。同时专门介绍了 IPP（Intel Integrated Performance Primitives，Intel 集成性能原语）库以及如何在高速图像应用程序中使用 IPP。

第 7 章，测试与性能。提供了一个把单元测试融入软件开发周期中的合理策略，其中包括一个完整的单元测试框架，并且讨论了如何扩展此框架以适应你的特定需求。这一章的另一个重点是性能，在此提供了一些可以使应用程序运行时性能迅速提升的具体技术。

第 8 章，高级主题。对一些有必要做更深入讨论的问题进行了研究，例如写拷贝（copy on write）、高速缓存（caching）、关键字 `explicit` 的使用、`const` 以及传引用（pass by reference，即传址）等。本章中有一节内容是关于该图像框架的扩展，以此可以指导你使用已有的框架，并基于此增加自己的处理函数。我们还强调了一些例程，它们对改善数字图像有很大帮助。

附录 A，有用的在线资源。这一部分提供了我们认为有用的软件工具和资源的链接。

附录 B，光盘信息。这一部分概要介绍了本书附带光盘的内容。在此包括书中出现的大量代码。我们的测试应用、原型和图像框架的所有源代码都可以在光盘中找到。另外，我们还提供了所有单元测试、单元测试框架以及在各种平台上运行软件所需的 makefile 文件。一些有用的第三方软件也包含在内，包括：Microsoft Windows SysInternals 开发的自由软件 DebugView 实用程序、Microsoft Windows 和 Linux 下 IPP 库的评估版、Microsoft Windows 下 Intel C++

Compiler 的评估版，以及建立 JPEG 文件委托和 TIFF 文件委托的源代码。

排版约定

该书中我们使用的约定如下：

斜体（italics） 强调新术语。

这是一个导航标记，以强调正在讨论的主题。

当对象声明出现在相应头文件中时，以此将其包围起来。

不建议采用的编码实践。带有此标记的代码在语法上可能是正确的，但是可能效率较低、不安全或者从另外的角度考虑是存在问题的，在相关的段落里会有讨论。

提示。指出某件事是应该做或者是不应该做的；明确标出是为了引起读者对这些重要信息的注意。

//注释 代码中的注释。为简短起见，书中的代码注释并没有包括源代码中的所有注释，光盘中则包含了所有源代码和完整的注释。

致谢

假如没有给予我们巨大鼓励、支持和帮助的人们，就不会有本书的出版，在这里感谢他们付出的时间与耐心。

我们要特别感谢 Donald D. Anderson、Louis F. Iorio 和 David Lanznar，他们花费了大量的个人时间反复审阅本书的原稿。他们富有洞察力的评论和技术经验使得本书更加完善。还要特别感谢 Don 在运行时问题领域的无价的贡献和独到的建议。

我们还要深深感谢 Mary Dageforde、Steve Vinoski 和 Jan Christiaan van Winkel，感谢他们全面、综合的技术审查和建议。他们在细节上细致的考虑和意见使本书的代码和原稿都得到了提高。

以下人员也为本书从多方面做出了技术上的贡献：Neil Jacobson、Benson Margulies、Neil Levine、Thomas Emerson、David Parmenter、Evan Morton 和 John Field。感谢你们！

我们也很幸运能够同 Addison-Wesley 这样一个杰出的团队一起工作，其中包括 Debbie Lafferty、Peter Gordon、Mike Hendrikson、Bernard Gaffney、John Fuller、Tyrrell Albaugh、Chanda Leary-Coutu、Melanie Buck、Curt Johnson 和 Beth Byers。他们以职业道德、奉献精神和严格的标准鼓励我们承担并完成了此书。

最后，要感谢督促我们前进的 Team Checkpoint。

Philip Romanik，2003

Amy Muntz，2003

目　录

1 绪 论

任何一个程序员，无论使用 C++还是其他编程语言，都会把自己的偏好和经验带到他们的软件开发工作中去。在本书中，将采用 C++来解决数字图像领域的特定问题。相对于处理数字图像（可能是数码相机拍出的照片），应用 C++构造并查看其直接效果是一种更好的做法。

本书以一个从数字图像生成缩略图的简单应用开始（该应用并不完备）起步。通过建立原型，我们将测试一组 C++技术以及有关此缩略应用的设计思想。通过应用适当的 C++技术，我们接下来就可以用所学到的内容来建立一个健壮的图像框架，如图 1-1 所示。

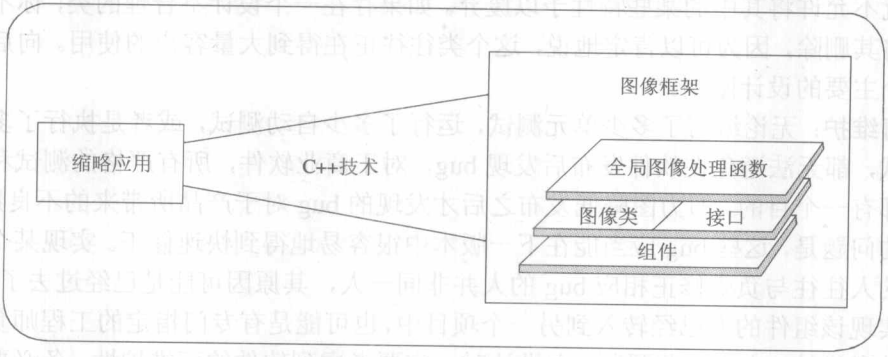

图 1-1 开发路线图

在此过程中，我们将研究以下问题：设计该应用的最佳途径是采用继承还是使用模板？是在静态的初始化时完成所有一切，还是使用一个单例对象？显式的模板实例化是否会带来语法或功能上的好处？是否在设计中加入使用引用（rep）对象和句柄的引用计数？如何划分全局函数和对象的功能？采用哪种框架更适合于异常处理？模板特殊化是否会对我们有所帮助？如何使例程运行得更快？除了讨论这些问题外，本书还编写了大量的代码来测试我们的设想，并检验在选择特定应用时所做的权衡。

我们的出发点是要开发商业软件，那具有以下特征的软件：由群体开发、可扩展、可维护、可理解且稳定。我们将这些特征作为标准，对所要探讨的 C++设计和应用技术做出评价。

■ **群体开发**：软件由一个团队来完成，从本质上讲，这说明没有哪一个人会对所有的代

码都了如指掌。同时，这也意味着团队成员之间交流的好坏直接决定着软件的成败。交流应当始于软件开发工作的开始；团队中的每个成员都必须对究竟要做什么、为谁做以及需要什么时间完成等问题达成共识。通过理解客户和他们的需求，就可以经常自我反省："所增加的特性是否确实必要？"。理解了商业和产品需求，就可以避免通常导致产品延期的因素，如产品中充斥着不必要的特性，或者是所面向范围过大等。必须在整个软件开发过程中自始至终地进行这种交流，其中包括交换接口设计思想，从而使组件间的集成能够平滑地完成；为代码建立准确而完整的文档；向 SQA（Software Quality Assurance，软件质量保证）小组和文档（Documentation）小组提交新的成员函数及其用途。

- **可扩展**：必须能够快速地在商业软件中增加新的特性，而且应当能够扩展其现有的特性。当某些重要的用户发现他们需要新的功能来解决问题时，经常会提出这样的要求。如果在产品已经发布之后才发现某种局限，那么情况将更为糟糕。可否容易地扩展代码来满足这些要求是软件设计的一个基本功能。尽管增加新特性可能如同增加一个新的成员函数一样简单，但是对于重要的新特性，往往要求设计也有所扩展。简单地说，就是在设计和编写代码前考虑周全。如果仅仅实现某个规范，而未考虑将来的扩展和维护性，会导致将来很难做出修改。还存在另一个问题，大多数软件一旦开始使用，就不允许将其中的某些特性予以废弃。如果存在一个设计不合理的类，你不能简单地将其删除，因为可以肯定地说，这个类往往正在得到大量客户的使用。向后兼容是一个主要的设计限制。
- **可维护**：无论编写了多少单元测试，运行了多少自动测试，或者是执行了多少功能测试，都无法避免在软件发布后发现 bug。对于商业软件，所有严格的测试和准备工作都有一个目的，即力图降低发布之后才发现的 bug 对于产品所带来的不良影响。关键的问题是，这些 bug 应当能在下一版本中很容易地得到快速修正。实现某个特定组件的人往往与负责修正相应 bug 的人并非同一人，其原因可能是已经过去了太长时间，实现该组件的人已经转入到另一个项目中，也可能是有专门指定的工程师负责提供支持和维护。出于这些限制，在设计时一定要考虑到软件的可维护性。务必要注意一些固定的设计做法，而且编码风格务必清晰，只有这样才能避免更大的复杂性，从而高效地完成软件的维护。
- **可理解**：商业软件通常具有可视的软件界面。如果开发的是一个嵌入式系统库，那么要保证公司里需要用到这个库的其他工程师都能够理解它。如果建立的是一个软件库，就要保证所提供的接口很容易被你的客户理解，以便他们用它来建立自己的应用。这不仅仅要求命名约定要有意义，更重要的是，要求你的设计和有关语言元素（如模板）的使用是清晰而准确的。
- **稳定**：商业软件必须很稳定，也就是说，能够正常运行足够长的时间，而不发生瘫痪、内存泄漏或其他无法解释的异常。

我们的偏好往往在构建软件的方法中最能体现出来。每次开始一项产品开发工作时，我们

都抱定着成功的信心，我们会尽可能使应用运转起来，从而降低风险。我们会从基础结构入手，进而开始具体的编码。通过以下几种途径，可以确保产品的成功：

- 建立一个代码基或主线，并保证从一开始就对其每晚进行构建。
- 建立一个简单的单元测试框架，并使用每晚构建系统确保其每天晚上都运行。
- 建立一个简化的原型并令其快速运行，从而可以应用不同的 C++技术和方法来得到适当的设计和最终实现。
- 保证主代码基不间断地运行，从而使其他工程人员以及支持小组（如 SQA 小组和文档小组）总能得到一个有效的工作基。
- 全组会议决不超过 30 分钟。如果在规定的时间内还不能将需要表达清楚，那么就要重新考虑这次会议的目的，或者召开一个相关人员的小型会议。

完成了这本书的学习后，除了能够更好地理解何时应用特定的 C++技巧以及相应的权衡考虑外，还会有什么收获呢？你将得到缩略应用和图像框架的全部源代码。图像框架提供了高效的图像存储和处理功能，并提供了图像处理的例程（包括边界锐化、噪声降低和图像缩减）以及到第三方图像处理和文件格式库的接口。可以使用图像框架来创建自己的健壮应用，在此可以直接使用图像框架与第三方图像处理库建立接口，也可以对我们所提供的图像处理例程加以扩展以创建自己的库。

书中还包括了全部的单元测试以及为运行这些测试而设计的单元测试要求。此外，书中还包括了一个简单的资源管理器，可以用来对你的软件进行国际化，以使用双字节语言（例如汉语和日语）。

在附录 A 中列出了一些有用的库和软件工具。

1.1 图像基础

这一节将简要介绍一些图像处理中的常用概念，如果你对数字图像及其特性比较熟悉，则可以跳过这一部分，直接开始第 2 章的学习。

图像处理 应用是指所有以图像为输入，然后对该图像进行一些处理，最后输出一个结果图像的程序，所输出的图像可能不同于输入的图像，也可能完全相同。

在图像处理中，图像具有特定的意义。本书最后部分提供的图像处理框架可以处理多种类型的图像，例如 8 位和 32 位灰度图像以及彩色图像。不过，为了介绍图像的特性，我们将从 8 位灰度图像开始。

灰度图像 可以理解为黑白胶卷所拍摄的照片。8 位的灰度图像是由*像素*（图片元素，也称为 pel）构成的，这些像素只有灰度级别而不包含颜色内容。一个像素包含了图像中一个点的图像信息，可以不连续地取 0（黑色）～255（白色）之间的整数值，0～255 之间的不同值代表了不同级别的灰度。

我们以像素为单位，用图像的宽度（x 轴）和高度（y 轴）来指定图像大小，而且图像原点（0，0）位于屏幕的左上角。之所以习惯上采用屏幕的左上角作为图像的原点，是因为这样

可以更容易地将像素的坐标映射为存储该像素的物理内存位置。一般地，像素位置的增长方向（从左到右）与内存的增长方向一致。这种性质如图 1-2 中所示。

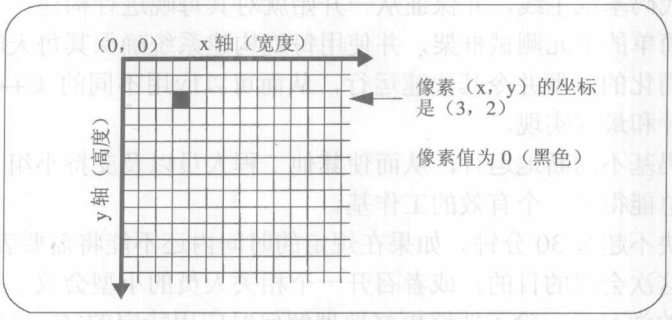

图 1-2 灰度图像性质

尽管大多数人都选择用数码相机来拍摄彩色图片，但灰度图像对于许多应用仍然是十分重要的。例如机器视觉、x 光以及医学成像等工业应用都有赖于灰度图像处理来提供信息。在机器视觉下，诸如宽度、高度以及周长等空间度量通常取自于灰度图像。这是因为，灰度图像的处理速度要比彩色图像的处理快三倍，这意味着生产线可以获得更大的吞吐量。灰度图像在使用 x 光技术进行物质探测和分析等领域也作为标准。

在最初的原型中，我们将一个灰度图像表示成一个二维的像素数组，并将每个像素表示为一个 8 位的无符号量（`unsigned char`）。

其他复杂的图像处理应用程序中则会需要彩色图像。精确分析就需要彩色图像，例如在基于颜色排序的应用中便是如此（例如制药业中采用不同的颜色来区分不同种类的药）。另外，由数码相机得到的通常主要是彩色图片。

1.1.1 RGB 图像

尽管*彩色图像*有多种表示方法，但在我们的原型里，它被表示成一个 24 位图像，其每个像素中 8 位表示红色（Red），8 位表示绿色（Green），8 位表示蓝色（Blue），每种颜色可以在 0～255 之间不连续地取值（0 代表没有颜色，255 代表纯色）。这称为 *RGB*（Red-Green-Blue）颜色空间，这三种基本颜色可以组合成人类能感知的各种颜色。因为用于灰度图像的算法可以简单地通过进行三重处理扩展为处理彩色图像，所以采用 RGB 颜色空间处理彩色图像很直观。

如前所述，RGB 值有多种表示方法，其中一些是以人类的感知为基础的。在这些颜色模型中，由于人的眼睛对绿色的感觉要比红色和蓝色敏感，所以通常采用更多的位来表示绿色，而用较少的位表示红色和蓝色。如果应用生成的图像是提供给人观看的，那么这个问题就显得尤为重要；而对于利用机器完成图像分析的应用，这个问题就不那么重要了。

通常，图像的表示需要在存储要求和所需分辨率之间取一个折衷。例如，一个 16 位 RGB 图像的存储空间只相当于 24 位 RGB 图像存储空间的三分之二。由于在 16 位的颜色空间中无

法将 16 个位平均分配给红、绿、蓝三种颜色，所以一般将多出的 1 位分给绿色通道。因此红色占 5 位，绿色占 6 位，蓝色占 5 位，这种分配方法被称作 5:6:5 颜色模型。

1.1.2 HSI 图像

另一种可以选择的颜色空间是 *HIS*（Hue-Saturation-Intensity，色调-饱和度-强度）。这种颜色空间模仿人类眼睛感知颜色的方式。一种颜色的*色调*就是它在电磁波谱可见部分中所在的位置。例如，一种颜色可能是浅绿色、蓝绿色或者草绿色。色调可以理解为*颜色轮*上的一个弧度，所有颜色都可映射至该颜色轮上。图 1-3 显示了一个红色位于 0 度的颜色轮，这是一种很常见的表示方法，其中箭头用以显示弧度，箭头的长度没有意义。

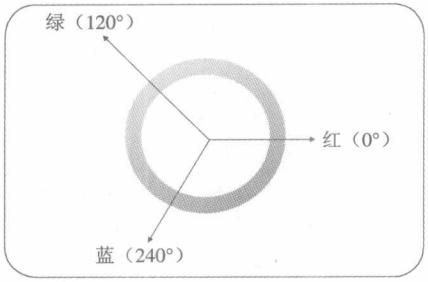

图 1-3　颜色轮上的色调

颜色的*饱和度*由这种颜色中混合了多少灰色和白色决定。一种完全饱和的颜色中除了该颜色本身以外，不包含灰色或白色。当灰色和白色加入到某种颜色后，该颜色的饱和度就会降低。例如，柠檬色是一种满饱和度的黄色，而香蕉色就是一种饱和度较小的黄色。

颜色的*强度*（在有些图形程序中也称为*亮度*）就是指这种颜色的亮度。

尽管用 HIS 颜色空间来描述颜色比较方便，但这种方法在处理速度上达不到我们的要求。与灰度图像类似，彩色图像的大小用其宽度和高度（以像素为单位）来表示。类似地，它的坐标原点（0，0）位于左上角。我们在原型中所使用的彩色图像性质如图 1-4 所示。

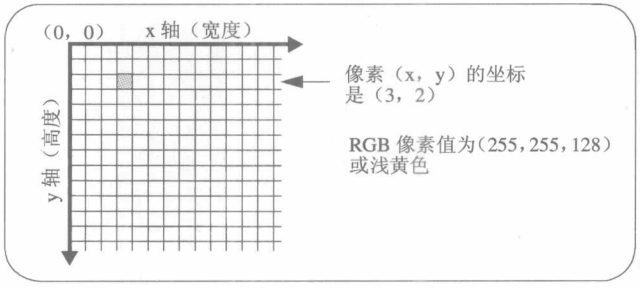

图 1-4　采用 RGB 颜色空间的彩色图像的性质

1.2 小结

在这一章中，我们对商业软件的基本特征进行了简单介绍，我们将以这些特征来衡量本书所提供的技术。另外还介绍了贯穿全书的一个图像框架应用，这个应用将作为各种 C++构造和设计技术的测试床。最后，我们对图像以及图像处理术语进行了基本的介绍，并以此作为结束。我们认为，所有这些内容对于理解书中的例程都是相当必要的；如果想使用这里所提供的框架，从而将图像处理技术充分地应用于自己的数字图像，那么这些概念也是必不可少的。

在第 2 章中，我们将通过建立一个生成缩略图的简单测试应用，开始 C++研究之旅。

本章包括以下内容：
C++概念
类设计
简单内存分配
赋值和复制构造函数

图像框架概念
图像对象设计
缩略图

2 一个测试应用

本书中所使用的测试应用是一个简单的图像处理应用，它以满分辨率的灰度图像作为输入，并生成相应图像的缩略。这是一个非常简单且不完善的测试程序，我们将以此作为测试床来分析后面章节所研究的 C++技术。本书的目标是通过有效地应用 C++技术，使用原型来逐步建立一个健壮的、具有商业质量的图像框架。

*缩略图*比原来的图像要小得多，但是它仍然包含足够的图像内容，使之看起来和原来的图像是一样的。例如，由于缩略的图像体积较小，传输速度快，所以在网站中经常使用缩略图。通常可以通过点击站点上的缩略图来显示原始的满分辨率图像。

由于这个测试应用所要做的具体工作很有限，因此它所要达到的设计目标也很少。这个应用将要完成以下工作：

- 以一个灰度图像作为输入。
- 按照用户指定的比例系数（例如原图像大小的 1/2、1/3 或 1/4）计算出缩略图。

设计和实现一个 C++类解决这个问题十分容易。书中这个测试程序的最初设计也很简单，只是根据以上设计目标和好的编程经验作为指导原则。在以后的章节中，随着需求的增加，设计也会不断地扩展。

之所以要使用反复的设计过程，其目的是为了说明：一开始满足目标的设计可能并不是最佳的解决方案。在后面的章节中还会介绍很多的技术，并给出不同的提示，这些技术和提示可确保可靠的、可扩展的设计能够最终满足得到一个完备的图像框架的目标。

2.1 图像类的设计

原始的灰度图像和所生成的缩略图有一些共同的属性。例如，两者都是二维图像，都能保存在一个文件或者在内存里对其进行操作。两者惟一的不同之处在于图像的大小和内容。例如，由类生成的一幅缩略图也可以用作为输入图像。对于这种设计，图像类具有以下属性：

- 这是一个灰度像素的二维数组。在这个例子中，一个像素可以表示为一个 8 位的无符号量（即 `unsigned char`），图像大小由图像宽度和高度确定。
- 允许对图像中的像素进行随机读/写访问。图像中的每个像素都由坐标（x，y）来描

述其位置。图像的原点坐标（0，0）指定为图像的左上角，坐标（width-1，height-1）表示图像右下角的像素。

- 该类为图像数据提供了一种简单的内存分配机制。对于一幅 x 维大小为 width、y 维大小为 height 的图像，其内存如下分配：

```
unsigned char* pixels = new unsigned char [width * height];
```

此图像中任意一个像素（x，y）的地址如下：

```
unsigned char* address = pixels + y*width + x;
```

- 当试图访问超出图像范围的像素时，将抛出一个 C++异常，即 rangeError。

这个图像类最初的设计就是这么简单。从表面看来，它达到了设计目标。图像类的定义如第 2.3.1 节所示。

2.2 缩略类

在我们的设计中，图像类只是图像像素的容器，并不包含其他的内容。由于这个应用的目的是创建缩略图，所以需要一个缩略类来处理文件 I/O 问题以及完成缩略的生成算法。一个缩略类具有以下属性：

- 读取来自文件的输入图像。
- 根据给定的输入文件和缩小系数（即要创建的缩略图到底有多小）来计算缩略图。
- 如果发生错误，则抛出一个 C++异常，即 invalid。如果图像类抛出了 rangeError 异常，此异常将被捕获，并代之以 invalid 异常。
- 把缩略图写入一个文件。

缩略类的完整定义如第 2.3.2 节所示。

2.2.1 缩略生成算法

对于缩略图的每一个像素，均通过对输入图像的大量像素取平均值获得。缩略图的像素值 T（x0，y0）采用以下公式计算，如图 2-1 所示，其中，factor 是期望的缩小系数。

$$T[x0,y0] = \frac{\sum_{x=0}^{factor-1}\sum_{y=0}^{factor-1} P[(x0*factor)+x,(y0*factor)+y]}{factor * factor}$$

图 2-1 计算缩略像素的值

下面用一幅图片来说明这个公式。使用图 2-1 中的公式，对原始图像 P 中的像素取平均值，