

C++在科学和工程应用 软件中的设计与实现

熊文杰 编译
范 峥
汪亚文 审



北京希望电脑公司

C++在科学和工程应用软件中的设计与实现

熊文杰等编译

924
3
7451

公司

C++在科学和工程应用 软件中的设计与实现

熊文杰 编译
范 峥

汪亚文 审

北京希望电脑公司
一九九二年

前 言

本书介绍了一个数值分析软件工具箱的设计、结构和用法。该软件工具是用 C++ 2.0 版写的，充分利用了该语言面向对象的程序设计的特点。它的开发环境是 Borland 公司 Turbo c++ 1.0 版，适用于 IBM 个人计算机及其兼容机。

这套软件工具及其背景描述覆盖了数值分析最主要的方面。讲述了在大部分科学和工程应用程序中使用的一些概念和技术。最基本的概念和技术包括浮点和复数的计算、C++ 函数库中的数学函数，和用于向量，多项式，矩阵代数的面向对象的框架结构。在此基础上，开发了求解非线性方程，求解线性和非线性方程组和求特征值问题的例行程序。

本书着重介绍软件开发。对用软件工具和面向对象程序设计技术处理向量，多项式和矩阵的着重介绍是本书新颖之处。本书构造了可重用的工具，用户通过这些工具能实现求解多种问题的许多技术；而不是介绍用特定的数值技术求解特定的应用问题。书中用了许多实例来说明这些工具的用法。这些工具是按层组织的。最底层是由 C++ 计算实数和复数的库函数组成的。整个软件包是结构化的，所以很容易选择实数或复数标量。在这些标量例程的基础上是向量代数软件。向量被当作面向对象程序设计的对象来处理；一个程序可以同时处理任意维的向量。这个特点对于处理多项式对象很重要。多项式对象继承了向量的特性，并有其自身的特性。例如，你可以把不同次数的多项式相加，把它们当作不同维数的向量处理：你可以乘多项式，不过这种操作对向量总的来说没有用。许多高水平的数值分析工具可以建立在向量和多项式代数的例程上。此外，另一个定义和处理矩阵对象的软件层建立在向量级之上。

本书强调了在处理向量，多项式和矩阵代数方面面向对象程序设计的特点。把高等数学组织到向量空间，多项式和矩阵代数等理论中去，是几十年来科学和工程应用程序的目标。这是一种有效的、优美的、非常成功的方法学。通过类似的策略，面向对象的程序设计开始让开发者获得与成套软件工具可比拟的通用性和好处。本书运用面向对象程序设计技术来组织数值应用程序并使它们看上去象它们所代表的数学形式。

完全可以把 C++ 数值软件整个写在一个 loop 框架内，其中每个计算实体都是某些类的一个对象。Turbo C++ 和它的类库(class library)支持这一过程。这些技术当前都在发展之中。C++ 的设计者 Bjarne Stroustrup 所写的 C++ 注释参考手册中论述了一些试验性的语言特点，也许会包括在下一个 c++ 版本中。我们的目标是：使程序易于建立、理解和维护。

目 录

第一章 引言	1
1.1 适用对象	2
1.2 使用软件工具	3
1.3 本书的结构	4
1.4 C、C++和 Turbo C++	5
1.5 MSP 软件工具箱	9
1.6 参考书目和进一步的讨论	10
第二章 软件组织	13
2.1 软件层次	13
2.2 C++辅助特征	18
2.3 抽象数据类型及类	21
2.4 成员函数和友员函数	24
2.5 构造函数和析构函数	25
2.6 重载	28
2.7 派生类	28
第三章 MSP 软件介绍	31
3.1 General 模块	32
3.2 Scalar 模块的 Doubwl 版本	38
3.3 处理向量对象	42
3.4 向量代数函数	49
3.5 矩阵对象的处理	55
3.6 矩阵代数函数	64
3.7 多项式	69
第四章 实数和复数的计算	82
4.1 整数运算	82
4.2 浮点运算	86
4.3 math.H 库函数	90
4.4 MSP 模块 RealFunc	96
4.5 Turbo C++算术和 Math.H 错误处理	98
4.6 使用复数标量	109
4.7 Complex 库函数	114

第五章 解实、复数方程	119
5.1 序言	119
5.2 二分法	120
5.3 不动点迭代法	125
5.4 Newton-Raphson 迭代法	129
5.5 实多项式的根	135
5.6 复数方程及其根	140
第六章 矩阵运算	145
6.1 线性方程组	145
6.2 高斯消元法	147
6.3 行列式	151
6.4 高斯消元法的软件实现	152
6.5 一般方程组	159
6.6 矩阵求逆	163
6.7 用 Hibert 矩阵检验	165
6.8 特征值与复线性方程组	167
第七章 方程组的迭代解	174
7.1 多维不动点迭代	175
7.2 线性方程组的迭代解	185
7.3 多维 Newton-Raphson 迭代	193
附 录	203
A.1 向量和矩阵显示输出函数	203
A.2 向量和矩阵的流输入/输出操作符	204
A.3 MSP 头文件	206
A.4 数学函数示例及中间输出	210
A.5 RowFinder 类成员函数	211
A.6 参考文献	212

第一章 引言

本书介绍了开发科学和工程应用软件时使用的 C++ 技术。它将告诉读者如何使用 C++ 面向对象的程序设计 (oop) 方法来定义和操作各种数学结构的类 (class) 或对象 (object)，通常还包括它们的应用——向量、矩阵、多项式及复数。C++ 语言将会比其它任何语言更易使你掌握这些对象，但要付出相当的代价：用户自己必须提供构成、支持、控制及分解这些结构的基本算法。本书详细解释了这些技术，并实现了向量、矩阵、多项式运算的基本算法。它将指出如何在此基础上构造功能强大的应用软件。

本书使用的语言环境是 Turbo C++ 1.00，由 Borland 国际有限公司出版(参见[5])。用于这里的大多数语言和 C++ 语言库的特点，均包括在 C++ 版本 2.0 中，由 AT&T 贝尔实验室开发(参见[13])。C++ 包括一个 C 语言的版本，只与在美国国家标准 X3.159-1989 中阐述的 ANSI C 语言稍有不同，Turbo C++ 与标准相异的地方均作了注释。

本书介绍如何利用 C++ OOP 技术来促进科学和工程应用软件的开发，它们如何增强软件的质量？此项研究需要对已考虑的软件作更加严格的限制。本书详细阐述了应用于下列数值分析领域的软件技术：

- 实数和复数的运算
- 基本函数
- 向量和矩阵代数
- 多项式代数
- 超越方程和多项式方程的解法
- 线性方程组的解法
- 特征值问题
- 非线性方程组的解法

其中大多数软件适用于实数和复数的标量运算。另外，本书还仔细研究了标准 C 语言和 Turbo C++ 对运算函数和库函数错误的处理方法。

对科学和工程应用软件的全面介绍需要涉及许多别的领域，但本书的重点是讲述软件开发的方法，而不是作面面俱到的介绍。当你在上述列举到的领域进行软件设计时，将会遇到主要的 OOP 软件开发问题。一旦能成功地解决这些问题，那么在其它领域基于 C++ OOP 方法的算法实现——例如，迭代和初始值问题——将迎刃而解。实际上，本书涉及了在这些领域的数值分析方面出现的大多数真正复杂的程序设计问题，但因时间和篇幅的关系，对这些问题的阐述也有所限制。

本书是围绕数学软件包（简称为 MSP）进行讲述的，这是作者为用 C++ OOP 方法实现数值分析算法而开发的一种软件包。典型的应用问题被列举，以证明为什么在科学和工程实践中需要这些算法。出于需要，本书中也讨论了基础数学，如果理解算法需要比通常的入门课程更多的数学知识的话。对 MSP 程序，书中有详细的讨论，并列出大量的文档和源代码。测试程序实例及输出结果显示 MSP 是如何应用于实际的。

这章详细介绍本书的内容、目的以及读者，还讨论了通用软件工具箱的方法学，以及 C++、Turbo C++ 对开发科学和工程应用软件的支持与限制。最后，列出一部分参考书及

C++、Turbo C++对开发科学和工程应用软件的支持与限制。最后，列出一部分参考书及进一步对此进行研究的资料来源。

1.1 适用对象

什么人需要这本书呢？如果你正在开发科学或工程应用软件，而且必须采用 C++语言以保证与其它一些软件的兼容性；或者你可能想用 C++OOP 技术促进软件开发；或者是想看看它们是如何在此领域中提高软件质量的，那么你就需要本书的帮助。

AT&T 贝尔实验室扩展了通常的系统程序设计语言 C 语言来构成 C++，这种语言是为了把 OOP 技术运用于各种不同的应用程序而设计的。因此，C++语言是一个折衷的方案。为了加强程序开发的效果，并保证程序的稳固性，OOP 技术在程序设计实践方面有所限制；而从另一方面来说，设计 C 语言的主要目的就是要取消这些限制。

OOP 技术在定义数据对象的类别方面与高等数学的组织方式类似。在研究数学结构时，你会把对数据的操作与这些数据本身放在一起考虑。例如，研究向量空间时，会考虑到向量加法和标量乘法运算及向量本身。而且，可能由于一些附加特征，你会把一些结构作为其它结构的特例。比如，一个多项式代数是一个带有一些附加运算的向量空间，包括多项式乘法和求值。为了解决在应用中产生的更进一步的数学和逻辑问题，经过几十年，这种数学的组织方式已得到改进。如果把 OOP 技术运用于程序设计的实践中，则会对数学软件的开发产生强烈的影响，特别是可以构造一些非常通用的例程，它们能轻易地适用于特定的环境中。而且软件与数学间的关系也将更为清楚了。

许多，但并非所有的 OOP 技术是采用 C++语言实现的。C++语言加上 Turbo C++编译程序的广泛普及以及基本 C 语言相当的灵活性，使之成为 OOP 技术在软件开发应用中的一个理想工具。本书的 MSP 软件包代表了这种运用的当前状况。

本书是对把 OOP 技术渗透到科学和工程应用软件开发中的一个讨论，但并不是要把 C++语言作为开发软件的标准语言。从今以后二十五年，下一代的程序员将把 OOP 技术当作是普遍而通用的程序设计方法，C++语言将被作为一种不同寻常的语言而被记住，这种语言把 OOP 特征融合进来，成为在商业环境中进行实验性开发的工具。本书将帮助读者进行这些尝试。

C++要求用户提供一定的软件来产生、保存、控制和分解应用中所需的数据结构，采用其它语言的应用程序员不会遇到这么多“低级”的工作。本书为向量、矩阵、多项式代数建立的 MSP 模块可以让其用户高效、轻松地完成这些工作。其它更高一级的模块是为解决方程式、方程组设置的，说明了如何在低级的 OOP 例程基础上建立更强大的应用软件。

在此仅仅是对 MSP 项目的掠影，这个软件以及用户自己对 MSP 软件包的改进和加强都将不断地进行。而且，C++语言可能会很快地发生变化。在《The Annotated C++ Reference Manual》（参见[13]）中，C++语言的设计者 Bjarne Stroustrup 用两章的篇幅介绍了试验性的技术——class templates（类模板）和 exception handling（例外处理）。这已超出了用于本书的语言和编译器的范围。这两种新的概念都是与科学和工程应用程序设计直接相关的。如果这两种概念可行，那么本书的 MSP 软件将会有不同的设计。最后，

OOP 本身也是当前计算机科学研究的一个领域，在今后的几年中，研究中产生的新思想将导致更新更好的语言，而且最终将在商业软件开发中成为主要的趋势。

因此，读者应怀着浓厚的兴趣学习 C++ 应用软件，但使用时也要持谨慎的态度。这种语言和它的 OOP 技术及本书将帮助你方便地实现某些算法，从另一方面讲，用户程序不会保存很长时间，因为不久会被更新的方法所取代。作为程序员会有更强的手段，为了更有效、更好地完成你的工作，你会重新开始。当然，这也不是终结。

尽管 C++ 语言是作为一种研究工具形成的，但它也可用于一些产品性的应用软件中。这将从 Turbo C++ 的广泛性和 Borland 公司对这种有力的、方便的编译程序不断的支持中得到确保。因此，为保持与其它 C++ 软件的兼容，需要用 C++ 来开发科学和工程应用软件。如果是这样，本书将阐明问题并帮助你开始一个令人兴奋的程序设计项目。

1.1.1 前提条件

在阅览本书之前读者必须知道些什么？本书假设读者在某些高级语言是一个熟练的程序员，而且熟悉 Turbo C 语言。本书中叙述了 C++ 语言大多数特有的功能，但并非是完全详尽的。读者需要一本 C++ 的手册。对通常的硬件概念只需了解：存贮管理、栈、寄存器。

为了掌握 Turbo C++，应熟悉 IBM PC 兼容机的各种硬件及 DOS 用户服务例程。最后，为了理解本书中涉及的数值分析算法，还应掌握大学程度的数学知识。

Turbo C++ 很容易充分发挥计算机的速度和存储能力。作者用来产生 MSP 软件和书写本书的系统在 1.4 节有叙述。

1.2 使用软件工具

C++ 要求用户提供软件来产生、保存、控制和分解应用程序用到的数据结构，例如：向量，矩阵和多项式，采用其它语言的应用程序员不会遇到这么多低级别的工作。作为一个 C++ 程序员，需要一个完成这些工作的程序工具箱。它将包括实数和复数，基本函数，向量和矩阵及多项式运算的程序。这种低级别的工具箱可以用更高级别的程序进一步扩展，以便完成象计算行列式、解方程或方程组、计算积分等更为复杂的工作。这些更高级别的程序可以象在数值分析中出现的理论和方法一样进行组织。工具箱中有解决一般问题的算法，因此就可以集中精力解决在工作中出现的新问题了。

Turbo C++ 函数库是作为工具箱进行组织的，它在所有应用领域中给程序员提供服务，你能随时用到它的函数，尤其是输入/输出、字符串处理、数学函数等。读者应该熟悉它的结构——见其《参考指南》，参见[5]——否则就是在浪费时间去解决一个程序库中已有答案的问题。程序库包括下列程序设计领域的大约五百个函数：

控制存储区域	内存管理
文本字符串处理	执行分程序
文本方式屏幕处理	计时

键盘输入 数学函数
文件输入 / 输出 错误处理
目录管理 低级图形输出

由于上述例程中的大多数已在 ANSI C 标准中有定义, 因此它们也同样适用其它编译程序。本书则主要介绍函数库中的数学函数及它的错误处理能力。

虽然 Turbo C++ 程序库是既综合又带有专业性的, 但在某些领域并不能满足用户的需要。当然, 其中之一是数值分析。这就是写本书的主要原因。对你的工作, 可能很关键的其它方面是异步通讯、交互输入 / 输出及高级图形输出, 在这些领域里许多商用产品可以提供更进一步的支持, 这在 1.6 部分中有所提及。

什么原则能够决定工具箱的开发呢? 首先, 分隔数学问题和程序设计问题, 当它们吸引你的全部注意力时要把它们分割开来, 逐个击破。如果想建立有用的工具, 这是一种有效的方法。第二, 力求通用。每一种应用都会与以前的有所区别, 一种工具如果太特殊了就会一遍遍地修正, 而且每次都会失去一点稳固性。第三点, 测试代码, 尽可能在其它方面和各种不同情况下使用工具箱。最后, 写出可读的、易理解的、正确的代码和文档。如果一个程序员不懂得一种工具是如何使用的, 那他也就不会把它用于他工作中的产品上。

在 Turbo C++ 程序库中及其商用附本中的工具都是按以上准则编写的。本书的 MSP 软件包中的工具同样也是, 但有所区别。商用产品旨在协助建立产品软件, 而且经过相当的一段时间后已得到检测和修改。而 MSP 是一个实验性的工具箱, 没有这么长的历史。

1.3 本书的结构

本书的章节分成三部分:

介绍

1. 引言

2. 软件组织方式

MSP 和 Turbo C++ 程序库的基础

3. MSP 简介

4. 实数和复数的计算

高级 MSP 程序

5. 解实数和复数方程

6. 矩阵运算

7. 方程组的迭代法

第二章讲述程序设计基础, 这章首先讨论在应用程序执行过程中同时运行的软件层。从程序本身讲到象本书 MSP 软件一样的软件层, 讲到 Turbo C++ 和 DOS 层, 最后到底层的 BIOS。第二章的其余部分是叙述 OOP 和 C++ 原理, 这些原理都是本书用到的数据结构——复数、向量、矩阵和多项式——的基础。

中间的一章讲述低级 MSP 程序以及 Turbo C++ 程序库中数学和错误处理的功能。低级 MSP 程序构造、保存、控制和分解基础结构, 并对复数、向量、矩阵和多项式进行代

数运算。第三章和第四章的顺序可以颠倒。现有的顺序是为了保持第二章和第三章的连续性。

最后几章更为典型地介绍了科学和工程应用程序设计。为了解决在应用中经常出现的数学问题，这几章讲述了几种不同的算法。这些算法都是以微积分和线性及多项式代数为基础的。在中间章节中讲到的低级程序的基础上，这些方法均可用 C++ 语言来完成实现。在最后三章中选择这几种算法有两个原因：

- 它们是数值分析中非常基本的概念；
- 它们提供了一个对低级程序的测试：设计和实现的主要问题都会在此暴露出来。

由于篇幅和时间的限制，高级程序在这几章中未作详细叙述。

大多数的科学和工程应用程序设计书都把重点放在高级算法和实现上，而不重视或者忽略低级方面。但采用 C++ 语言的主要原因是为了在应用中运用 OOP 技术，必须由数据结构类型和低级程序来组成。因此，本书必须强调那些方面。一旦掌握了在最后三章提及的高级算法的实现方法，那么把它们运用到数值分析的其它领域也是易如反掌的。

本书下一步是什么？

在本书中，你将看到 MSP 软件包的缩影。如果此书还将继续讲述高级算法，则应包括下面几章：

- 插值法和近似法；
- 数值积分和微分；
- 普通微分方程初值问题；

你可以看到并没有什么显著的地方需要强调，而且，等到上述提及的内容写出来时，C++ 语言可能已更进一步发展了，那就需要对基本的东西重新编写，以此改进某些对当前版本来说并非理想的 MSP 特征。

可选择的磁盘：

当在书中详尽地讨论大多数 MSP 程序时，并非包括了所有的源代码，由于这个原因，并为用户在使用时修改和改进 MSP 方便起见，所有的源代码都在一张可选择的磁盘上。其目录列于附录 A.6。

1.4 C, C++ 和 Turbo C++

这一节讨论采用 C 语言和 C++ 语言进行程序设计的益处和代价，以及采用 Turbo C++ 编译器的优点，并对用于本书及 MSP 的软件和硬件作了简要的小结。

1.4.1 为什么采用 C++

由于 C++ 是 C 语言的扩展，所以那些支持或反对在软件开发中采用这些语言的理由都是相关的。在程序设计的文献中，有几种支持 C 语言的论述。

- C 语言是一种高级语言，很适合表示各类算法。
- C 语言便于编写出易读的程序。
- C 语言程序对许多计算机系统都是可移植的。

- 有很好的编译程序。
- 在许多领域中有很好的软件工具。
- C 编译程序可以产生许多小的，可快速执行的代码。
- C 语言可以访问和控制硬件特征。
- C 语言程序设计使你考虑低级细节。

那么上述这些论断如何运用在 C++ 语言中呢？下面它们被逐一考虑，作者对如上描述的并不都持赞同态度，第一点和最后一点甚至是矛盾的。

1.4.2 C++ 是高级语言吗？

原来的 C 语言并非高级语言，用 C 语言时必须不时考虑到低级语言的地址细节。由于那些细节极易弄错，因此，用 C 语言编程时很不顺利，除非是一个 C 语言专家。即使在一个简单的程序中避免了绝大多数寻址问题，在调试过程中你还是会遇到它们，以便找出程序为什么没有按照预期的执行。C 语言并非能够很好地表示大多数算法规则，除了那些明显与寻址有关的问题，低级内容通常起决定作用，而且 C 语言程序看起来并不象算法，因此，第一点与最后一点的矛盾就解决了，最后一点是对的。

C 语言程序的确便于编写优美易读的程序。但它是持中立的立场：因为它也会写出粗糙、不可读的程序。

大多数把 C 语言转化成 C++ 语言的改进形式都支持高级语言的特征。这里有一些例子：

- 类(class)的概念。就象在高等数学中，可以把运算定义以及一个数据结构的定义融合在一起。
- 把特殊的 K 类作为一般 G 类的一个实例。可以编写控制 G 中的所有对象的程序，并把它们运用到 K 中的特定实例，K 中可能会有更具体的数据结构和附加的或更为特殊的运算。
- 函数名和操作符重载。可以使用相似的、直观的记号，对不同类的对象作相似的操作。
- 引用参量。可以通过引用代替值来传送函数参数，而无需明确指明其地址。
- 参数初始化。可以把缺省值分配给一些函数参数，并在调用序列中省略它们，除非要给予特殊的值。
- 输入/输出流。包含输入/输出操作的 C++ 程序库比 C 程序库中的输入/输出函数更灵活。

如果能系统地运用这些特征，就可以用 C++ 语言编写出比 C 语言更精巧、更易读的程序。事实上，本书 MSP 软件工具箱的主要目的就是使你编写的应用程序与进行运算的数学描述更接近。利用 C++ 的这些特征的代价是必须提供具体的基础例程，就象低级 MSP 程序一样，这些例程来组成、保存、控制、分解所用的数据结构。

C++ 语言解决了 OOP 数据封装技术方面的问题。一个类定义可以声明数据结构的某些元素成为私有(private)的，只有指定的成员(member)函数和友员(friend)函数可以访问它们。由于只有有限的函数可以改变私有的数据，这就使调试更为容易。然而，通过运用低级语言 C 语言的设计可以取消这种限制。

C++语言的另外两个特征使用起来并不方便。首先，类概念的内容之一是对成员函数的基本参数作强制性语法限制，这也许会与标准数学记号冲突，而不利於它们的使用。例如，数据结构 `complex`（复数）的实部(`real`)和虚部是私有的。`real`是一个返回 `complex` 变量 `z` 的实部的函数。如果 `real` 是成员函数，那必须以 `z.real()` 的形式调用它；为了保护标准记号 `real(z)`，它被声明成 `friend`（友员）。但是友员函数的定义不能放到数据结构的定义内。其次，参量初始化功能会导致与重载的冲突，产生难以查明的错误。而且，这些设计问题难以解决。

正如 1.1 部分提到的，C++高级语言的某些方面发展还不完善。例如，已有两种不同输入/输出流的版本。类模板(`classtemplate`)的概念为了包括在 C++语言的后续版本中，正在不断地进行推敲，这样就可以更好地定义。一个一般的向量类，和象整数向量、复数向量和多项式向量这样的向量类。如果增加了新的特征，则 MSP 需要重新修订。同样地，C++语言的错误处理方法在下一个版本中也尚未完善，而且也可能引起 MSP 的另一次重新修订。

用当今正在发展的语言，可以很快地获取不断改进的程序设计方法，但却缺乏一定的稳固性。

总的来说，C++语言包含了正在发展的、但很强的高级语言的特征，同时这些特征又融合了基本的低级语言的特征。如果没有这些特征，就没有必要使用 C++语言。如果能适当地编写低级语言为基础的程序，就可以编写能够清晰地反映基本数学的、优美易读的高级应用程序。虽然如此，在调试和维护程序过程中，仍会不断地迫使你面对低级语言的细节，以便能够找出发生了什么程序错误操作。

1.4.3 软件工具

对 C 程序员来说，有许多可用的软件工具。在 1.2 节中已简要介绍了 Turbo C++程序库和一些商用辅助软件。《Advanced Turbo C》一书（参见[3]）详细介绍了大多数 Turbo C 程序和一个商用辅助软件。

当然，所有的 C 语言工具都可以在 C++中工作。现在又开始出现了专为 C++语言设计的软件工具箱。本书中的 MSP 软件工具箱就是一例。但必须谨慎地使用这些软件工具。一般来说，这些软件工具是类定义及成员函数，友员函数，相关函数的代码组成的程序库。由于 OOP 还是一个正在研究的领域，而且 C++语言也正在发展中，所以任何这样的软件工具——特别是 MSP——都是实验性的。用 OOP 技术解决一般程序设计问题的最佳途径至今还未发现。

1.4.4 局部的考虑

你可以编写可移植到许多环境去的 C 语言程序，而且许多机器都有很好的 C 语言编译器。从某种程度上讲，C++语言程序就没有那么易于移植，而且 C++语言编译程序就更少。原因很简单，因为它还是一种刚刚兴起的语言。正如前所述，C++语言是作为研究程序语言设计的工具而编写的，这就可能限制编译程序的数量。（从另一方面讲，对 C 语

言也可能作出同样的陈述)。Turbo C++对 c 和 c++语言都是很好的编译程序，以后还会提到它特别的优点。

C 语言编译程序可以产生小的、快速执行的代码。很少哪个有 c++编译程序为了有关 C++语言的论断而被详尽地研究过。对 C++语言作出评价很困难，因为 C++语言是为了加强程序开发和维护而设计的，而这两者之间还存在着矛盾。本书着重介绍开发过程，并介绍软件设计的实验结果。在此，程序的运行速度和大小是不重要的。

以前选择 C 语言作为一种开发语言的最后一个原因是让用户访问和控制硬件特征。当然这对 C++语言（和一些其它的高级语言）同样适用。这对大多数科学和工程的应用程序设计来说是无关紧要的，除非你的设计项目要从传感器或通讯电缆上读数据，或者需要控制一些设备。

1.4.5 Turbo C++

为了什么要采用 Turbo C++编译程序？下述的讨论是令人信服的：

- Turbo C++有一个很好的用户接口。
- Turbo C++大多数特点符合用户的习惯。
- 在编辑程序中可以立即标出格式错误。
- 编辑程序可以模仿通常的字处理程序。
- 通过编辑程序用户能够控制源代码调试器。
- Turbo C++的程序库是全面的、可靠的，并有大量的文档。
- Turbo C++具备支持较大软件系统开发的标准特征。
- Turbo C++可以利用扩展存储器来编辑，编译，连接较大的文件。
- C++的最新版本 Borland C++可以在 Microsoft Windows 下运行，而且包括开发 Windows 应用程序的工具箱。

最新的版本对这个研究项目不适用。作者对 Turbo C++还有一些批评的意见：

• 用来生成本书的早期版本有一些错误。这些错误 Borland 已通过 Compuserve 信息服务网承认了，而且正在纠正。

• 用引用传递参数既不是很准确的又不能很好地实现，它使作者无法发现问题，这些或者会暴露出编译程序错误，或者带来 C++设计错误，使 C++的这个特征不能很好地加以利用。在任何情况下，错误的程序都会破坏 Turbo C++系统，而且通常需要重新启动。

• 调试员不允许对临时变量进行检验，也不允许引用传递大多数参数。

• 从某种程度上来说编译是很慢的。部分原因是由于极长的头文件。编译的新版本对此已作明显的改进。

总的来说，Turbo C++是很好用的。

1.4.6 本书的产生：

一个 33MHz 的台湾产 80386 系统用来开发本书的软件并编写文本，它有 80MB 的硬

盘。没有装数字协处理器。有两倍于原来 4MB 的存储器，这个存储器是由 QuarterdeckSoftware 公司的 QEMM 和 DESQview 软件管理的。六个 DESQview 窗口可以同时运行：

IBM	4.01 版本
F	一个共享目录管理程序
WordPerfect	5.1 版本
CC	一个 Prentice-Hall 数学台式程序
PC File+	2.0 版本，一个 ButtonWare 数据库产品——作者的个人信息管理器。

Turbo C++ 1.00 版本

在 DESQview 之前已装了 1MB 的磁盘缓冲器。并且，在 CC 窗口中作为内存驻留工具，首先装进 Word Perfect 屏幕控制器。而 DESQview 则用于把 Turbo C++或 DOS 窗口的文件拷贝到 Word Perfect。CC 是图形软件，WordPerfect 可以拷贝这些图形。对于编辑来说 Super VGA 很重要。

对 Turbo C++和 Word Perfect 来说，这个系统的运行速度是很关键的，作者不希望在低于这个速度的系统上进行 Turbo C++的软件开发。用 C++小存储模式编译作为低级 MSP 函数的测试程序是必需的，因为运行在此级别时极易发生下标和其它指针的错误，如果程序是用小型模式编译的，则通常会引起程序发生 NULL pointer assignment“空指针赋值”错误。在测试程序终结后，Turbo C++会识别出这些错误。而对于高级程序的测试程序，则需数据和代码空间大于 64K，所以用大模式进行编译。

1.5 MSP 软件工具箱

本书介绍：如何用 C++ Object Oriented Programming(OOP)技术来加强科学和工程应用软件开发？全书围绕 MSP（数学软件包）软件工具箱来构成这些思想，其主要设计目的是建立所需的软件基础，使用户程序员可以用这个工具箱编写出漂亮的，易读的程序，并使这些程序能够反映对所实现的算法的数学描述。

正如高等数学的组织一样，OOP 技术允许把类定义和操作定义封装在一起。而且，可以把一些类作为更为一般的类的特例，就象多项式代数是向量空间代数的特例一样。最后，可以使运算符和函数名超出它们原先的意义，这样不同结构的类似运算可以写成相似的，正如同一个数学符号可以表示多项式乘法和向量乘法一样。

在 C++语言中采用这种方法，必须比应用程序更为注意低级语言的细节。你必须提供软件构造、保存、控制和分解所要用的数据结构，如复数，向量，矩阵，和多项式。MSP 的首要任务是定义那些数据结构的类型，提供低级别的支持。其次，MSP 还包括在这些类型的基础上进行基本代数运算的程序，并以此为基础，建立高级 MSP 程序，进行特殊的数值分析运算，以解决应用中的问题。

MSP 包括九个模块，前两个为其它模块提供通用的和数学上的支持：

General 为其它程序模块单元提供支持。

RealFunc 实数函数，作为对 Turbo C++程序库的补充。

RealFunc 实数函数，作为对 Turbo C++程序库的补充。

在 Turbo C++程序库中本应包含 General 和 RealFunc 的绝大多数功能。但实际上没有。下一个程序模块决定是否用实数或复数标量进行计算：

Scalar 支持标量运算的基本函数。

对于实数和复数标量，有两个版本。可以采用任一版本，其它的程序模块可自动调整进行实数或复数标量运算，同一代码对两种情况均适用。下面三个程序模块单元完成 MSP 的低级部分。

Vector Matrix Polynom

它们每个都包含支持相应数据结构和基本代数操作的程序。例如，Vector 包含构造、拷贝及输出向量的程序；而 Polynom 包含乘、除多项式的程序。最后四个程序单元是高级别的：

Equatcl 求解单个实数或复数未知量的方程。

GaussE1 通过高斯消元法解线性方程组，计算行列式和逆矩阵

Eigenval 构造矩阵的特征多项式并找出其特征值。

Equatc2 用迭代法解线性和非线性方程组。

这些程序实现了数值分析的基本算法。

一个综合的数值分析软件工具箱应包含更进一步的程序模块来解决插值和近似问题、数值积分和微分，初值等问题。尽管如此，本书主要强调软件开发，同时也包括在组成高级的程序模块时所遇到的主要程序设计问题。一旦了解了这些，实现更进一步的数值分析算法将是易如反掌的。因时间和篇幅的关系只到此结束。

Scalar 程序模块单元两个版本的运用都是简洁的，同时也显出运算符重载(operator overloading)的优点。但同时也存在不足。在 MSP 基础上建立的程序必须采用一个或者另一个，而很难同时处理两种类型的标量。若要执行单个复数计算，需要把所有数据结构用复数标量来存储，而且需要两倍于实数标量的空间。虽然可以让 MSP 更为有效地利用存储空间，但那样会使用户程序不易读，而且使用户程序员不得不更注意低级语言的细节。合适的解决方法是 C++语言的 template (模板) 概念，它还并不完善，而且还未包含在 Turbo C++中。(参见[13,第 14 节]) 如果可以利用这个特点，MSP 就该重新修订了。

MSP 也应该有更为系统的方法来报告和处理数学错误的情况，象溢出，或除零错误。MSP 具有某些能力，本书详细叙述了通过 Turbo C++程序库可以采用的方法，但错误处理的技术并未完善 (参见[13,第 15 节])，而且现在可以用 Turbo C++实现的并不一定在将来也是可能的，在此领域内一个主要的变化也可能引起 MSP 的重新修订。

1.6 参考书目和进一步的讨论。

为了用 Turbo C++对科学和工程软件进行开发，必须找到时常产生的硬件、软件和数学问题的答案。本书在这方面涉及的较少，尽管如此，本书在这节指出一些参考书目供读者参考，一些是与程序设计有关的，甚至是必不可少的，另外一些提供一些背景信息，

其中有一些是关于在 1.2 部分中提到的把工具箱概念扩展到商用软件包的信息，书后括号中的号码表示书刊目录。

本书假设你基本上熟悉 Turbo C 及其在程序设计中的应用。这包括对 C 语言的了解及对其 Borland 实现方法的熟悉。你需要使用 Turbo C++，但无需以前有什么经验。不过必须对 IBM 兼容硬件和 DOS 操作系统很熟悉。

Kernighan 和 Ritchie 写的 [27] 原始的 C 语言参考书，对学习 C 语言来说，并不太好，但却经常提到，似乎你应该有一本才对。由 Harbison 和 Steche [18] 写的参考著作很有用。最初的 C++ 参考书 Stroustrup [39]，已经被他带有权威性的著作《Annotated Reference Manual》所取代了。有许多介绍 C 语言、C++ 语言和 OOP 技术的文献和商用书籍，但作者发现它们中的大多数对一个有经验的程序员来说，是冗长而又近乎无用的。它们常常避免讨论可能在程序设计中引起麻烦的棘手问题。其中最好的一本是 Lippman 的书 [29]，但就连他都没有涉及一些有用的信息，而去研究一些不相关的情况。

有两本商用书籍，Norton [33] 及 Sargent 和 Shoemaker [36] 较全面地讨论 IBM PC 兼容硬件。你需要查阅系统技术参考手册来确定其组成是如何与这些书上的标准一致或背离的。在 IBM 技术参考手册 [24] 里，甚至对非 IBM 系统来说，都有很有用的信息。

如果需要有关 CPU 的机器语言或数学协处理器的详尽信息，可查阅 Intel 80286-80287 手册 [22]。对于 80386 和 80486 也有类似的手册。80287 手册上涉及了有关协处理器的信息。

了解有关 DOS 的信息，其手册当然是很重要的，随软件提供的手册是非常必需的，但对高级程序设计应用来说却是不合适的。高级 IBM 指南过分昂贵，而现在几乎被价廉物美且必不可少的 Microsoft MS-DOS Encyclopedia [32] 所取代了，包括 DOS 3.3；你可以找到有关后来版本特殊细节方面的信息，由 Norton [34] 和 Duncan 所著的商用书籍中用不太正式的方式对此有所提及。（Duncan 是一个百科全书的编辑）。

Turbo C++ 有一套四册的指南，其“专业性”的版本还有针对 Turbo Assembler [3] 的另外三册，针对 Turbo Profiler [7] 的一册，针对单独 Turbo Debugger [6] 的一册。Turbo C++ 的后一个版本是 Borland C++，它把上述这些结合在一起。你必须逐渐熟悉 Turbo C++ 指南，其它三本与本书无直接的关系。

有一本书 Advanced Turbo C [37] 对 Turbo C 语言及其大部分程序库作了完整的叙述。高昂的 Turbo C 程序库源代码可能是“太”有用了。如果你不能通过文件理解一个程序，你就必须进行实验或分析其源代码。作者花了“太多”时间在这两个方面。对于本书，这种类型的问题是 4.5 部分讨论的错误处理程序；那些程序会随着 Turbo C++ 以后版本的变化而改变。

可以采用与 Turbo C++ 兼容的好的软件工具。科学和工程应用的程序员时常进行大胆尝试，特别是组成交互的用户接口，用数据传感器、计算机或其它设备处理串行通讯，以及显示图形输出。Blaise (Complting) 有限公司的产品，Turbo C TOOLS [2] 是一种综合而且性能可靠的工具箱，尤其适用于建立接口设备。另一种 Blaise 公司的产品，CASYNCH MANAGER [1]，提供了可靠的低级 C 语言支撑来实现中断驱动的串行通讯，还有几个工具箱用于扩展 Turbo C++ 低级图形能力。如想得到有关这方面产品的进一步信息，请查阅广告和商业周刊上的介绍文章。

科学和工程应用程序员碰到数学问题就象碰到有关硬件或软件问题一样频繁。可以利用 Courand[n]和 Hille[9]的经典著作作为微积分及其有关论题的参考。对于高等的微积分和代数问题, Wylie[45]和 Uspensky[40]的著作则是丰富的来源。Kuuth 的书[28]在有关算法的所有数学书中是一个很好的信息来源。在许多数值分析的著作中, 有两本特别好的参考书: Burden 和 Faires[8], 及 Maron 和 Lopea[30]所著的两本书, 前一本文字流畅, 后一本有大量实例。

最后, 你应该了解在科学和工程计算中一种新开发的软件: 数学台式软件。现在有几种软件可以按照要求进行计算。你可以通过键盘键入问题, 在表示法上与标准数学表示法近似, 并且从显示器上迅速读出答案。有些软件显示二维和三维图形并用图形技术显示实际数学表示法中的公式。数学台式软件能够处理以前需要用高级语言编写程序的复杂问题, 大多数这类软件都是可编程的, 用专门的数学台式语言编程比用象 C++这样为了一般问题而设计的语言编程简单多了。