HZ BOOKS 华章IT

Pearson

追溯数学原理，探求高效编程的奥秘

STL之父Alexander A. Stepanov力作，Amazon广泛好评

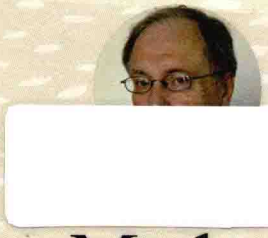
名家经典系列

数学与泛型编程

高效编程的奥秘

[美] 亚历山大 A. 斯捷潘诺夫 (Alexander A. Stepanov)
丹尼尔 E. 罗斯 (Daniel E. Rose) 著

爱飞翔 译



From Mathematics to Generic Programming

机械工业出版社
China Machine Press

名家经典系列

数学与泛型编程

高效编程的奥秘

亚历山大 A. 斯捷潘诺夫
(Alexander A. Stepanov)
[美] 著
丹尼尔 E. 罗斯
(Daniel E. Rose)

爱飞翔 译

From Mathematics to Generic Programming

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

数学与泛型编程：高效编程的奥秘 /（美）亚历山大 A. 斯捷潘诺夫，（美）丹尼尔 E. 罗斯著；爱飞翔译 . —北京：机械工业出版社，2017.8
（名家经典系列）
书名原文：From Mathematics to Generic Programming

ISBN 978-7-111-57658-7

I. 数… II. ① 亚… ② 丹… ③ 爱… III. 程序设计 IV. TP311

中国版本图书馆 CIP 数据核字（2017）第 184219 号

本书版权登记号：图字：01-2015-1921

数学与泛型编程：高效编程的奥秘

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号　邮政编码：100037）
责任编辑：唐晓琳　　责任校对：李秋荣
印　　刷：北京文昌阁彩色印刷有限责任公司　　版　　次：2017 年 8 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：15.75
书　　号：ISBN 978-7-111-57658-7　　定　　价：79.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

译 者 序

本书是从数学到泛型编程的一次精彩之旅。

书中讲解了与乘法、素数及最大公约数有关的古老算法，并展示了历代数学家对这些算法所做的探索。他们既想对求解的方法做推广，以扩大其适用范围，又想对计算的步骤做简化，以提升其运算效率。

就前一方面来看，近代数学的发展（尤其是数论与抽象代数这两个分支）为算法的推广提供了很大的帮助。数论令人更加深刻地了解到数字中的奥妙，并给高速计算提供了灵感，而抽象代数则使得那些本来只能处理整数的算法，逐渐变得可以处理分数、实数、复数，乃至多项式。这种泛化还发生在数据之间的运算上面，比方说，通过连加可以实现求积，然而如果把其中的运算由加法换为乘法，那么效果就会从求积变成求幂。

就后一方面来看，计算机的出现使得算法的效率有了很大的提升空间。我们可以根据运算步骤及数据所具备的特征来发挥电脑硬件与软件所拥有的特性。

作为全书主题的泛型编程体现了上述两个方面的融合。它既借鉴了数学中的抽象思维及知识整理办法，又利用了计算机编程语言所提供的泛型与优化机制。这使得某些因实际问题而起的原始算法，在经过千百年的深化和抽象之后，终于可以变得更加普适、更加高效。

作者把这些对泛型编程有益的思维与技术提炼成许多条原则，并通过严谨的数学证明来培养读者的逻辑推理能力。此外，书中还有几篇简明的学者传记，使大家能够了解数学史及计算机技术的发展脉络。将这些内容与公钥加密系统等有趣的实例结合起来，可以指导我们把数学知识更好地转化为泛型编程的成果。

在翻译过程中，我得到了机械工业出版社华章公司诸位工作人员的帮助，在此深表谢意。

由于译者水平有限，书中难免出现错误与疏漏之处，请大家发邮件至 eastarstormlee@gmail.com，或访问 github.com/jeffreybaoshenlee/fm2gp-errata/issues 留言，给我以批评和指教。

爱飞翔

2017 年 4 月

致　谢

笔者要感谢令本书得以面世的每一个人。我们在 A9.com 的管理团队从一开始就积极协助这个项目。Bill Stasior 曾提议设立一门课程，并从我们所给出的选项中挑出了一些话题，本书正是以该课程为基础而撰写的。Brian Pinkerton 不仅出席了该课程，而且还强烈鼓励我们把课程素材写成一本书。感谢 Mat Marcus，他于 2004 ～ 2005 年同 Alex 合作，在 Adobe 开设了类似的课程。

Fundamental Data Structures and Algorithms for Search 团队的其他成员在成书过程中扮演了重要的角色。Anil Gangolli 帮我们确定了课程的内容，Ryan Ernst 对基础编程环境的搭建工作出力很多，Paramjit Oberoi 在我们写书时给出了宝贵的建议。笔者喜欢与他们一起工作，并感谢他们所提出的意见。

感谢我们的编辑 Peter Gordon 及 Greg Doench，也感谢 Addison-Wesley 所召集的专家团队，其中包括经理编辑 John Fuller、制作编辑 Mary Kesel Wilson、文稿编辑 Jill Hobbs 以及排版员 / LaTeX 专家 Lori Hughes，他们通过努力的工作把笔者的草稿制作成了精美的书籍。

还要感谢朋友及家人，同时感谢阅读本书初稿或给我们提供评论、修正、建议、意见或其他帮助的同事：Gašper Ažman、John Banning、Cynthia Dwork、Hernan Epelman、Ryan Ernst、Anil Gangolli、Susan Gruber、Jon Kalb、Robert Lehr、Dmitry Leshchiner、Tom London、Mark Manasse、Paul McJones、Nicolas Nicolov、Gor Nishanov、Paramjit Oberoi、Sean Parent、Fernando Pelliccioni、John Reiser、Robert Rose、Stefan Vargyas 及 Adam Young。他们的努力，令本书的品质大有提升。

最后，感谢许多细心的读者报告了阅读本书时所发现的错误，他们是：Abutalib Aghayev、Vladimir Burenkov、Greg Ives、Nitin Kumar、John Lakos、Andy Lawman、Jeremy Murphy、Anil Pal、Miguel Pinkas、Daniel Roldán、Alexander Slinkin、Saul Tamari 和 Boris Vassilev。如果你发现了错误可以发送邮件至 errata@fm2gp.com。可以在以下网址下载本书的勘误：www.fm2gp.com/errata.html。

作者简介

亚历山大A. 斯捷潘诺夫（Alexander A. Stepanov）于1967～1972年在莫斯科国立大学（Moscow State University）研习数学。他从1972年开始编程，1977年移民美国之后，继续从事编程工作。他参与了操作系统、编程工具、编译器与程序库的开发，其对编程基础的研究工作得到了通用电气（GE）、纽约理工大学（Polytechnic University）、贝尔实验室（Bell Labs）、惠普（HP）、SGI及Adobe的支持。2009年Amazon旗下的搜索技术公司A9.com开始支持这项工作。由于对C++标准模板库的设计有贡献，1995年他获得了《Dr. Dobb's Journal》的杰出编程奖。

丹尼尔E. 罗斯（Daniel E. Rose）是一位研究科学家，在Apple、AltaVista、Xigo、Yahoo及A9.com从事管理工作。他广泛地研究搜索技术，关注针对索引压缩的底层算法，以及Web搜索中的人机交互等问题。Rose曾在Apple公司带领团队为苹果电脑创建了桌面搜索机制。他拥有圣迭戈加利福尼亚大学（University of California, San Diego）的认知科学与计算机科学博士学位，以及哈佛大学（Harvard University）的学士学位。

作者附言

如果将计算机科学与数学分离，那么这两者的发展都会有很大困难。于是，我们就试图通过一些课程，把人类文明早期就有的数学活动与现代才有的计算机活动结合起来。本书正是基于这样一种课程而编写的。

能够与友人 Dan Rose 合作，我深感荣幸。他的管理工作令我们团队能够把泛型编程的原则运用到搜索引擎的设计上来，而且他愿意把我原来那些相当分散的课程内容集结成现在这样一本连贯的书籍。Dan 和我都希望读者能够喜欢我们这次合作的成果。

——A.A.S.

大家将要阅读的这本书是根据“Algorithmic Journeys”课程的笔记而撰写的，该课程于 2012 年由 Alex Stepanov 在 A9.com 讲授。当 Alex 与我合作把课程材料改编为书籍时，我们发现：其实可以将这些材料整理得更为集中一些，使其聚焦于泛型编程及其数学基础。这个想法促使我们对本书要讲解的话题进行了大幅度的调整，删掉了与集合论及逻辑有关的大段文字，因为那些内容似乎和本书要讲的内容无关。同时，我们还对一些细节有所增删，以便大家能够更加连贯地阅读本书，并使那些数学知识较缺乏的读者也能够顺畅地理解书中的内容。

Alex 是数学专业出身，但我不是。因此，我很努力地试着去读懂课程里的一些材料，并根据自身体会来确定那些需要加以解说的难点。如果你发现本书所用的一些描述方式及术语和专业数学家稍有不同，或是本书在解释某个问题时多写了几个简单的步骤，那么这应该归咎于我才对。

——D.E.R.

目　录

第 1 章
内容提要

不懂数学，就无法了解世界。

——罗吉尔·培根（Roger Bacon），《大著作》（Opus Majus）

这是一本谈编程的书，但是它与大多数的编程书都不太一样，因为除了算法和代码之外，本书还会给出数学证明和一些讲述从古代到 20 世纪各种数学发现的历史材料。

另一个更为具体的特色在于：这是一本谈论泛型编程（generic programming）的书。泛型编程是出现于 20 世纪 80 年代的编程方法，在 20 世纪 90 年代随着 C++ 标准模板库（Standard Template Library，STL）而变得流行起来。我们可以这样定义它：

定义 1.1　泛型编程是一种专注于对算法及数据结构进行设计的编程方式，它使得这些算法及数据结构能够在不损失效率的前提下，运用到最为通用的环境中。

用过 STL 的读者可能在想：“不对吧？泛型编程的概念只用这么简单的一句话就能定义出来？模板和迭代器（iterator）等特性怎么没有提到呢？”其实那些特性应该说是工具，它们使得编程语言能够支持泛型编程。程序员固然应该学会高效地使用那些工具，然而泛型编程主要谈的是编程态度（attitude），而不是某一套工具。

笔者认为，所有的程序员都应该抱持这种编程态度，也就是说，都应该试着以这种通用的方式来编写代码。如果能够写出高品质的泛型程序，那么很容易就能使用并修改其中的各个组件。这要比那种采用硬代码来编写的程序好很多，因为后者会针对具体的应用程序来给数据结构、算法以及接口施加一些毫无必要的限制。把程序写得通用一些可以令它变得更为简洁，也更为强大。

1.1　编程与数学

那么，这种泛型编程的想法是从哪里来的？我们又应该怎样来学习它呢？这种想法是从

数学中衍生出来的，尤其与抽象代数（abstract algebra）这个数学分支有关。为了使大家能够理解这种编程方式，本书会对抽象代数做一些介绍，并着重讲解怎样从抽象的运算属性来认识对象。这个话题一般是数学专业的大学生才去研究的，然而笔者认为，它对于我们理解泛型编程会起到关键的作用。

实际上，还有很多基本的编程概念也同样来自数学。对这些概念的产生及变化过程加以学习，能够促使我们更好地思考软件的设计问题。比方说，欧几里得（Euclid）的《几何原本》（Elements）虽然是两千多年前写成的书，但是其中的范例仍然具有很高的参考价值，它可以告诉我们，怎样用一些较小且较易理解的组件来构建一套复杂的系统。

尽管抽象是泛型编程的要义，但这个抽象却并不是本来就有的。我们必须从具体的事物入手，才能够获得更为抽象的认识，要想对某个领域进行正确的抽象，就必须理解该领域的细节。

抽象代数里面的抽象很大程度上是从另外一个数学分支的具体成果中得出的，那个分支比抽象代数更为古老，它叫做数论（number theory）。为此，我们还需要介绍数论中的某些关键概念，这些概念与整数的属性有关，其中尤其值得注意的是可除性（divisibility）。

我们在学习这些数学知识时所使用的思路可以对编程技巧起到提升作用，此外我们还会看到，对于当前某些软件来说，有一些数学成果本身就具有极其重要的意义，这尤其体现在给网络隐私与电子商务软件提供支持的加密协议上。本书最后会举例来说明某些数学知识在此类协议中的运用情况。

笔者会在数学和编程之间来回游走。当我们讲解某些重要的数学概念时，笔者会穿插一些对具体算法和通用编程技巧的讨论。笔者对其中某些算法只会给出简要的描述，而对另外一些算法则会在整本书里进行详细的讲解，并将其泛化。除了个别章节只谈数学或只谈编程之外，大部分章节都会同时涉及数学和编程这两个方面。

1.2 从历史的角度来讲解

如果能把某个话题融入故事中，那么我们学起来就会更容易一些，而且也会觉得更为有趣。我们想要知道：在某个时间发生了什么事件？都有谁与该事件相关？这些人是怎样产生某种想法的？他们是要在别人的基础之上进行研究，还是要驳斥别人的结论？为此，本书在介绍数学概念时，还会讲一些与此有关的故事，并谈一下提出这些概念的人。笔者通常用一份简要的传记来描述身为故事主角的数学家，这些小传不会像百科全书里面写得那样详尽，它只是想令你对这位数学家的情况有所了解而已。

尽管我们会从历史的角度来谈论某些概念，但这并不是一本讲数学史的书，而且也不会按照这些概念问世的先后顺序来进行讲解。我们可能会在空间和时间顺序上有所跳跃，因为笔者主要是想令大家了解每一个数学概念的历史背景。

1.3 阅读准备

由于书中的很多内容都和数学有关，因此你可能担心自己必须先具备丰富的数学知识，然后才能看懂这本书。其实你只要有逻辑思考能力就行（程序员应该很擅长逻辑思考），笔者并不会要求大家具备中学代数与中学几何之外的其他数学知识。某些章节可能会运用向量（vector）与矩阵（matrix）等线性代数（linear algebra）方面的概念，如果从前没有看过这方面的资料，那么把这些内容跳过去就可以了。若是对本书所用的记法不够熟悉，则请参考附录A。

数学中有一个很重要的部分就是对命题给出形式化证明。本书就包含了许多这样的证明过程。如果你在中学的几何课、计算机科学专业的自动机理论（automata theory）课以及逻辑课中做过一些证明，那么应该很容易就能理解本书所给出的证明。附录B描述了某些常用的证明技巧，并给出了范例。

笔者假设你已经是一名程序员了，而且对C、C++或Java等典型的命令式（imperative）编程语言相当熟悉。尽管书中的范例是用C++写的，但即便你原来没有用C++写过程序，也依然应该看得懂才对。附录C解释了一些C++特有的机制。虽说我们用的是C++语言，但笔者相信，书中所讲的原则能够适用于其他各种语言。

本书所谈的很多编程话题也同时出现在Stepanov与McJones所写的《编程原本》（Elements of Programming）一书中，而后者是从另外一种更加正式的角度来讲解这些话题的。想要深入研究这些话题的读者可以参考那本书，并将其与本书结合起来阅读。在本书里，我们偶尔也会提到《编程原本》中的相关章节。

1.4 各章概述

在详细讲解本书内容之前，我们先来简要叙述一下各章的概况：

- 第2章介绍一种古老的乘法算法，以及该算法的改进方式。
- 第3章初步讲解数字的某些性质，并给出一种寻找素数的高效算法。
- 第4章介绍一种寻找最大公约数（Greatest Common Divisor，GCD）的算法，后续的章节会以该算法为基础来讲述某些抽象思维及其运用方式。
- 第5章关注数学结论，我们会介绍几个重要的定理，这些定理在后续的章节中发挥着重要的作用。
- 第6章介绍数学中的抽象代数这一领域，泛型编程的核心思想正源自该领域。
- 第7章运用这些数学思想对乘法算法进行泛化，使它不仅能够执行简单的算数运算，而且还可以用来解决各种实际的编程问题。
- 第8章介绍一些新的抽象数学结构，并讲解怎样运用这些结构来解决一些新的问题。
- 第9章讲解公理系统、定理以及模型，这些都是泛型编程的基础组成部分。

- 第 10 章介绍泛型编程中的概念，并展示一些看似简单的编程任务中所蕴含着的微妙问题。
- 第 11 章继续研究某些基本的编程任务，并介绍怎样运用与该问题有关的理论知识，来实现各种实用的算法。
- 第 12 章讲解硬件方面的限制是怎样促使旧算法演化出新版本的，并针对 GCD 来展示一些新的运用方式。
- 第 13 章把数学结论与算法成果结合起来，以便在密码学上做一次重要的运用。
- 第 14 章总结本书所提到的某些基本观念。

编程与数学是两条贯穿于全书的线索，只不过在某些章里面，其中一条线索可能要比另一条更加明显。书中的每一章都体现了一段思路，这些思路合起来构成了全书的主旨，那就是：

> 要想成为优秀的程序员，就必须理解泛型编程的原则；要想理解泛型编程的原则，就必须学会抽象；要想学会抽象，就必须知道它所依据的数学基础。

以上就是笔者想要在本书中讲述的内容。

第 2 章
算法初谈

摩西很快就学会了算术与几何。
……这些知识是他从埃及人那里学来的，
埃及人最重视的研究科目是数学。
——亚历山大的斐洛（Philo of Alexandria），《摩西生平》（Life of Moses）

算法（algorithm）是用来完成计算任务的一系列有限步骤。由于算法与计算机编程的关系特别密切，因此很多人或许认为，算法是一个来自计算机科学专业的概念。其实算法这个词已经有几千年历史了。数学中充满了各种各样的算法，有些算法我们天天在用，就连小学生计算加法时所用的竖式（long addition）都可以说是一种算法。

尽管算法的历史很悠久，但它并不是天生就有的，而是由人发明出来的。虽然我们并不清楚第一个发明算法的人是谁，但我们知道，早在四千多年前埃及就已经有了某些算法。

* * *

古埃及文明以尼罗河为中心，尼罗河发洪水之后，土壤就变得肥沃起来，从而促进了古埃及的农业发展。可是问题在于，每次洪水过后，用来划分财产边界的那些标志也随之消失了。于是，埃及人就用绳子丈量距离，并拟定一套流程，使自己能够根据早前的书面记录来恢复当时的边界划分情况。有一批专职的祭司负责研究与此有关的数学技巧，这些人叫做“rope-stretcher”（拉绳者），希腊人后来把他们称为 geometer，也就是“Earth-measurer”（地形测量者）的意思。

我们很少发现与古埃及人的数学知识有关的文字记录。对于那个时期，目前只发现了两份数学文献，其中一份就是此处要提到的莱因德数学纸草书（Rhind Mathematical Papyru），它以苏格兰收藏家莱因德得名。莱因德在埃及所买的这份纸草书是公元前 1650 年左右由一位名叫阿姆士（Ahmes）的抄写员所写的。其中含有一系列算术与几何问题，而且还列有一些计算用的表格。这份卷轴里面所写的第一个算法是一种能够迅速计算乘法的技巧，此外还

有第二个算法，用来迅速地计算除法。我们首先来讲这个快速乘法算法，在本书后面的章节中大家会发现，该算法当今依然是一项重要的运算技巧。

2.1 埃及乘法算法

与所有的古文明一样，埃及的计数系统也没有按位置计数这一概念，而且无法表示 0。于是，乘法计算起来就特别困难，只有少数受过训练的专家才会做。（你可以想象一下：如果自己只能像使用罗马数字那样来做运算，而且要计算的数字又很大，那么算起来是相当困难的。）

怎样来定义乘法呢？宽泛地说，我们可以认为乘法就是“把某物多次加到它自己上面”，如果说得严谨一些，那么可以分两种情况来定义：一种情况是乘以 1，另一种情况是乘以一个大于 1 的数。

我们将乘以 1 的乘法运算，定义如下：

$$1a = a \tag{2.1}$$

接下来需要定义另一种情况，也就是怎样根据某数的 n 倍来计算其 n+1 倍。有些读者或许已经发现这是一个归纳的过程，本书稍后会以更为正规的形式来使用归纳法。

$$(n+1)\,a = na + a \tag{2.2}$$

有一种办法可以计算 n 与 a 的乘积，那就是把 n 个 a 连加起来。然而这种办法对于较大的数字来说相当乏味，因为总共要计算 n−1 次加法才行。此算法可以用 C++ 代码[⊖]表示为：

```
int multiply0(int n, int a) {
    if (n == 1) return a;
    return multiply0(n - 1, a) + a;
}
```

上面这个函数中的两行代码分别与等式（2.1）及等式（2.2）相对应。和古埃及人计算乘法时的情况一样，a 与 n 都必须是正数。

阿姆士所描述的乘法算法，古希腊人将其称为“埃及乘法”（Egyptian multiplication），而现在有很多人则把它叫做“俄罗斯农夫算法”[⊖]（Russian Peasant Algorithm）。这种算法所依据的原理是：

$$\begin{aligned} 4a &= ((a+a)+a)+a \\ &= (a+a)+(a+a) \end{aligned}$$

这个原理是根据加法结合律推算出来的：

⊖ 本书源代码请参见：www.fm2gp.com/downloads.html。——译者注

⊖ 很多计算机学者是从高德纳（Knuth）的《计算机程序设计艺术》（The Art of Computer Programming）一书中听到这个名字的，那本书说 19 世纪的旅行者在俄国看到有农夫使用这种算法。实际上，最先提到这个说法的人是 Sir Thomas Heath，他在 1911 年的一本书中写道：“我听说有一种办法至今依然在使用（有人说是在俄国，但我还无法核实）……”

$$a+(b+c)=(a+b)+c$$

如果采用这个办法，那么只需把 $a+a$ 的值计算一次就行了，这样可以降低加法运算的次数。

埃及乘法算法的思路是：反复地将 n 减半，并将 a 加倍，同时求出 a 的各种倍数，这些倍数与 a 的比值都是 2 的整数次幂。在那个时代，算法并不是用 a 或 n 这样的变量来描述的，而是以具体的数字举例，然后说："同样的操作还可以运用在其他数字上面"。阿姆士自然也不例外，他以 41×59（也就是说 $n=41$，$a=59$）为例，演示了怎样通过下面这种表格来执行该算法：

1	√	59
2		118
4		236
8	√	472
16		944
32	√	1888

表格左边那一列的数字都是 2 的整数次幂，而对于右边那一列来说，其中的每一个数字都是它上方那个数字的两倍（之所以要采用这种反复翻倍的办法来列表，是因为把同一个数字加到它自己上面计算起来要相对容易一些）。如果用二进制的形式来表示 41 这个数，那么值为 1 的那些二进制位就可以与表格中勾选的那些行分别对应起来。于是，这张表格实际上就相当于下面这个算式：

$$41\times59=(1\times59)+(8\times59)+(32\times59)$$

该等式的右侧出现了几个 59 的倍数值，这些倍数值都可以通过对 59 进行适当的翻倍而计算出来。

由于该算法在将 n 减半的时候需要判断 n 是奇数还是偶数，因此尽管没有直接的证据，但我们依然能够推测出：古埃及人已经知道了奇数和偶数之间的区别。因为古希腊人宣称他们是从埃及人那里学到数学的，所以这一点是可以肯定的。如果把埃及人定义奇数和偶数的办法[⊖]表示成现代的数学记号[⊜]，那就是：

$$n=\frac{n}{2}+\frac{n}{2}\Rightarrow \text{even}(n)$$

$$n=\frac{n-1}{2}+\frac{n-1}{2}+1\Rightarrow \text{odd}(n)$$

此外，我们还要依赖下面这个关系式：

⊖ 这个定义出现在《Introduction to Arithmetic》这本书的第1卷第7章，此书由杰拉什的尼科马库斯（Nicomachus of Gerasa）于公元1世纪撰写。他在书中写道："偶数就是可以分成两等份且中间不会有多余单位的数，而奇数则是无法分成两等份的数，因为它中间会多出来一个单位。"

⊜ 箭头符号"⇒"读作"蕴含"(imply)。附录A总结了本书所用的数学记法。