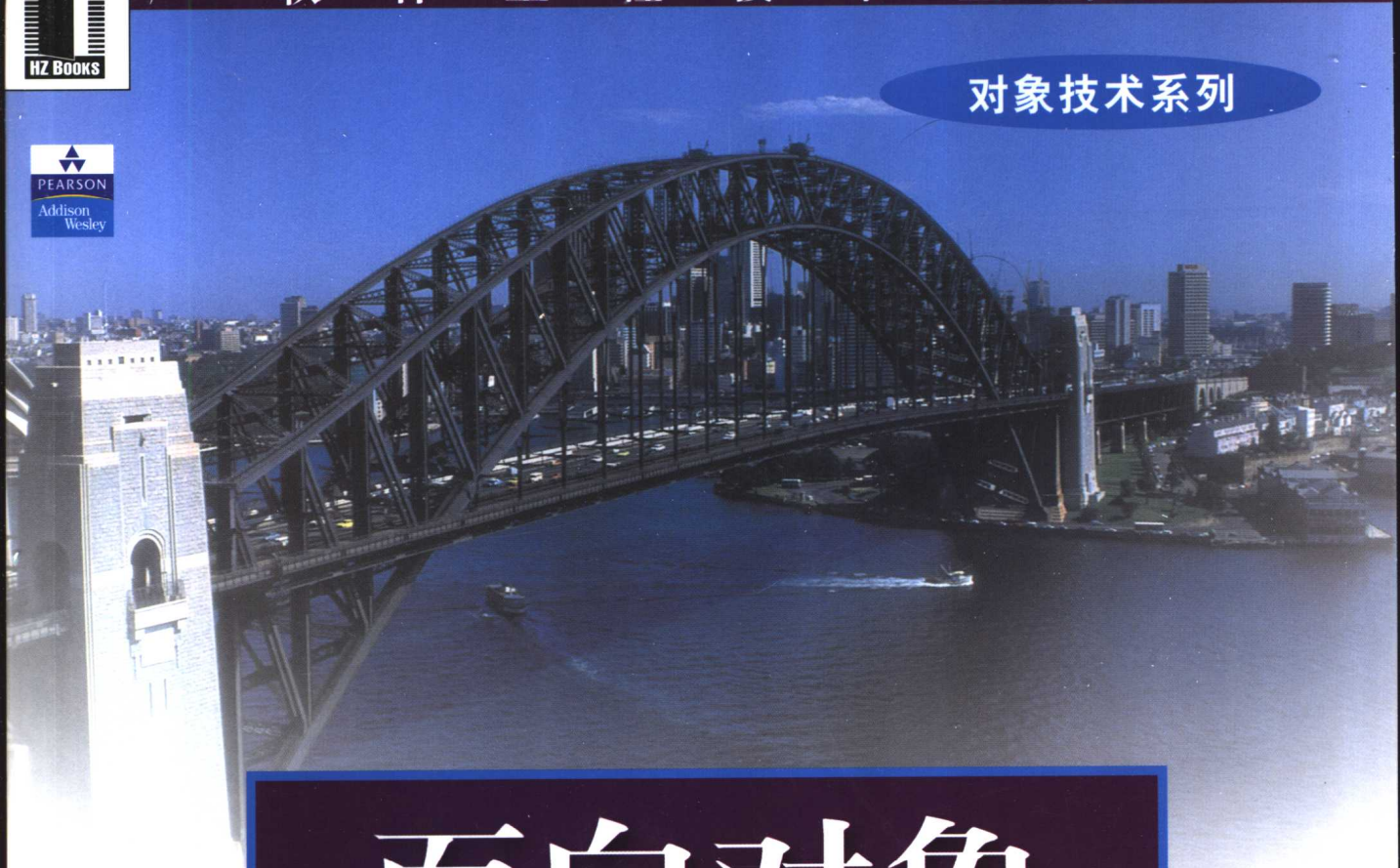
软 件 工 程 技 术 丛 书

HZ BOOKS

对象技术系列

PEARSON
Addison Wesley



面向对象编程导论

（原书第3版）

An Introduction to Object-Oriented Programming

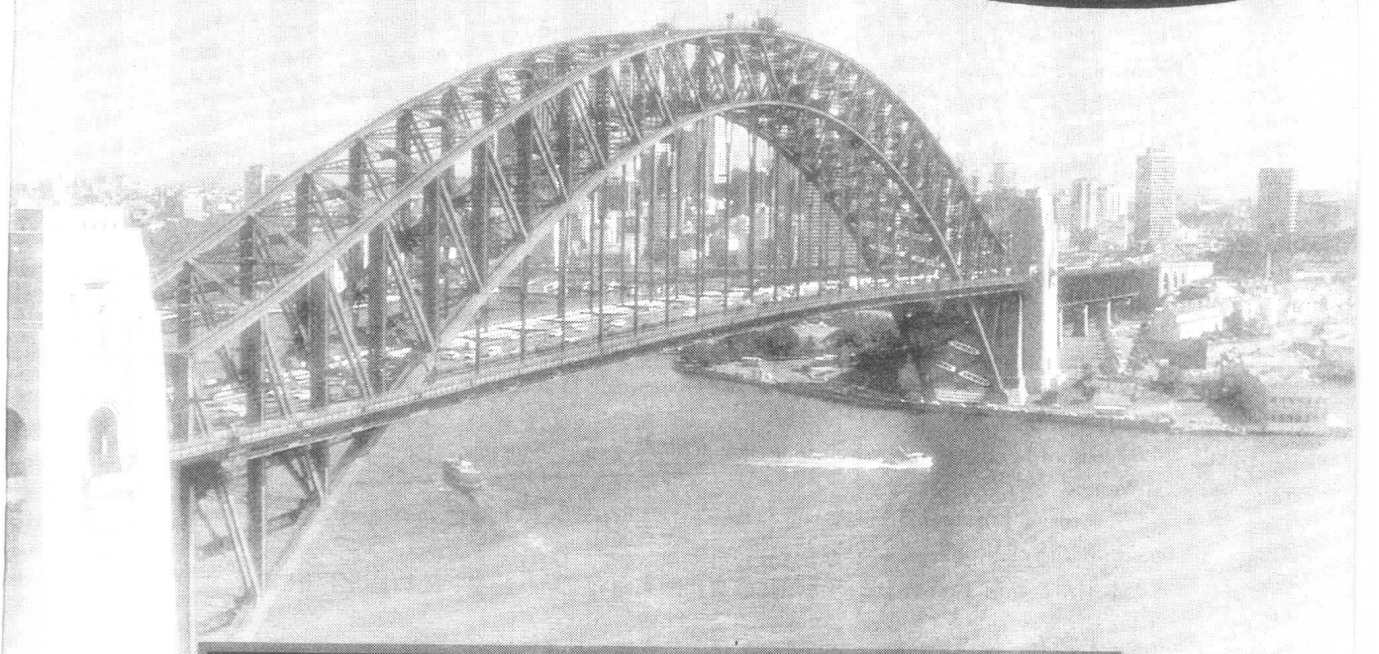
(Third Edition)

（美） Timothy A. Budd 著　　黄明军　李桂杰　译

机械工业出版社
China Machine Press

软 件 工 程 技 术 丛 书

对象技术系列



面向对象编程导论

（原书第3版）

An Introduction to Object-Oriented Programming

(Third Edition)

（美）Timothy A. Budd 著　　黄明军　李桂杰　译

机械工业出版社
China Machine Press

本书通过对象、方法、继承（包括多重继承）和多态等概念，以独立于编程语言的方式介绍了面向对象编程的原理。书中所列举的实例涉及多种编程语言，其中包括 Java、C++、C#、Delphi、Python、CLOS、Eiffel、Objective-C、Smalltalk 等。通过研究这些编程语言，读者可以更好地理解隐藏在各种语言语法之后的基本原理。此外，作者还从面向对象的角度对这些语言进行了分析比较。

本书内容全面，特别适合作为计算机专业本科高年级和研究生一年级的教材，同时也可以供那些从传统的面向过程编程转向面向对象编程、想要了解面向对象基本概念的初学者使用。

本书版权登记号：图字：01-2003-1997

图书在版编目（CIP）数据

面向对象编程导论（原书第3版）/（美）巴德（Budd, T.A.）著；黄明军等译. -北京：机械工业出版社，2003.9

（软件工程技术丛书　对象技术系列）

书名原文：An Introduction to Object-Oriented Programming (Third Edition)

ISBN 7-111-12666-1

Ⅰ.面…　Ⅱ.①巴…②黄…　Ⅲ.面向对象语言-程序设计　Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 068432 号

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：迟振春

北京昌平奔腾印刷厂印刷·新华书店北京发行所发行

2003 年 9 月第 1 版第 1 次印刷

787mm × 1092mm 1/16 · 30.25 印张

印数：0 001-5000 册

定价：45.00 元

译者序

本书介绍了面向对象编程和面向对象设计的基本概念，如对象、方法、封装、继承、多态等。

本书最重要的特点就是没有针对某种特定的编程语言来讨论面向对象技术，而是介绍了对所有语言都通用的面向对象思想。书中的实例涉及C++、Java、Delphi Pascal、Python、C#等各种面向对象编程语言。通过这些具体实例，读者可以更好地理解隐藏于各种语言语法之后的、也是更加重要的面向对象原理。此外，作者还从面向对象的角度对这些语言进行了分析比较。

本书主要包括以下几部分内容：

- 面向对象概念和面向对象设计。
- 类、方法和消息。
- 继承和软件复用。
- 继承的细节。
- 多态。
- 多态的应用。
- 对象互连。
- 高级主题。

本书首先围绕着面向对象所涉及的各种技术进行了全面的讨论。然后，介绍了设计模式、反射和内省、网络编程、面向对象语言的实现等高级技术，虽然篇幅短小，但却是本书的一个亮点。作者通过简要的文字将这些高级晦涩的技术讲解得简单易懂，可以使读者迅速地掌握其骨架。

本书既可以作为面向对象编程的教材，也可以供那些从传统的面向过程编程转向面向对象编程、想要了解面向对象基本概念的初学者使用。本书的读者应该至少具有一种传统语言（如C、Pascal等语言）的编程经验，当然，如果读者具有面向对象编程语言的经验，将会更容易理解本书的内容。

本书已经根据英文原版的最新勘误表（最近更新日期为2003年5月18日）进行了修正。

本书由黄明军、李桂杰共同翻译而成，由于水平有限，错误在所难免，恳请广大读者批评指正。

译　者

2003年5月

前　言

当我于1983年在Smalltalk上开始编写第一本书的时候，我清楚地记得，必须尽快地完成这本书，不要错过面向对象编程最热的时机。谁会想到20年之后，面向对象编程似乎仍然很热，它经历了一个多么长的、神奇的旅程啊！

在这20年中，人们对面向对象编程进行了全面的研究，它已经成为编程领域的范例。在这个过程中，它几乎改变了计算机科学的每个方面。并且我发现，与本书的第1版相比，我编写本书第3版的目的依然没变。我仍然希望学生们甚至读者们，能够理解基于一般原则的面向对象编程思想，而不是针对特定语言的细节。

在编程这一领域，各种语言来去匆匆，令人眼花缭乱。在本书的第1版中，我讨论了Objective-C语言和Object Pascal的Apple版本，当时这两种语言的使用都很广泛。虽然这两种语言现在仍然存在，但是都已经不再占统治地位了（然而，从语言的角度来看，Objective-C语言具有很多有趣且独特的特性，因此在第3版中，我还是对其进行了讨论）。在出版本书第1版和第3版之间，许多语言都已经消失了（例如Actor语言和Turing语言），同时又产生了一些新的语言（例如Java、Eiffel和Self语言）。许多原来就存在的语言也增加了对象扩展（例如Common Lisp语言和Object Perl语言），也有一些语言突然出现在人们的视野之内，又很快地消失了（例如Sather语言和Dylan语言）。还有Beta语言，这种语言通过简单的语法揭示了非常有趣的思想。很难对各种语言未来的前景做出预言。刚刚出现的语言，例如Ruby，是否会继续强大呢？会不会走Dylan语言的老路呢？C#语言又如何呢？很难想像微软公司所发明的语言不会成功，但是，奇怪的事情总是会发生的（我个人认为，由于C#语言为原来的Visual Basic程序员提供了一种更好的语言，因此会得以延续，但是Java程序员或者C++程序员几乎不会迁移到这种新语言上来。时间会告诉我们，我的预言是否要比他人的预言更加准确）。

在这一版中，我们扩大了用于举例的语言的数目，但是也减少了很多针对特定语言的长篇叙述。关于各种技术的描述通常都是通过表格或者更短小的形式给出的。本书的前两个版本，并没有作为任何语言的参考手册，想要使用我所举例的语言来编写程序的学生或读者，最好使用与特定语言相关的参考书籍。

但是，在第3版中，我还是尽量保持前两版的整体结构。本书结构包含如下各个主题。

Ⅰ.简介和设计。第1章介绍了面向对象编程的基本概念。第2章讨论了计算机科学家所使用的各种处理复杂问题的工具，以及面向对象技术是如何适应这种框架的。第3章通过介绍责任引入了设计原则。前面这三章内容是本书的基础，非常重要。为了增强对概念的理解，我强烈地建议读者能够以多个小组（至少是一个小组）的方式通过讨论CRC卡（在第3章中介绍）是如何解决问题来理解面向对象编程的概念的。通过小组程序操纵CRC卡，是我所遇到

的研究和理解行为、责任和封装等概念的最好方法。

近十年来，面向对象设计所涉及的领域越来越大。对于很多读者来说，第3章的内容可能会显得太少或者太多——对于那些已经具有大量面向对象建模语言和设计经验的读者来说可能是太多了，对于那些从未听说过这些概念的读者来说可能又太少了。但是，我会尽量平衡。虽然有很多可以替代责任驱动设计的面向对象设计技术，但由于这种设计方法是我认为的对于初学者最简单的设计技术，因此还是对其进行了讨论。

Ⅱ. 类、方法和消息。第4章和第5章介绍了我们的实例所使用的编程语言（Smalltalk、C++、Java、Objective-C、Object Pascal、Delphi Pascal和其他几种编程语言）的语法，并建立了类和方法，发送了消息。第4章主要关注的是编译时特征（类和方法），第5章讨论了动态行为（创建对象和发送消息）。第6章和第7章通过一系列的案例进一步讨论了这些思想，这些使用面向对象技术开发的案例程序说明了面向对象技术的各种特征。

Ⅲ. 继承和软件复用。虽然第1章就引入了继承，但直到第8章才主要讨论这个概念。继承和多态替换都是作为软件复用的主要技术进行讨论的。第9章中的案例研究是使用新引入的语言C#来编写的。这两个案例都说明了继承的应用和标准API（应用程序编程接口）的使用。

Ⅳ. 继承的细节。第10章至第13章深入地研究了继承和替换的细节。一种编程语言如果引入继承的特性，几乎会影响这种语言的所有方面，并且这种影响对于初学者来说很难察觉。第10章讨论了子类（subclass）和子类型（subtype）之间的细微差异。第11章讨论了各种不同的语言是如何支持静态和动态特征的。第12章说明了语言引入继承和多态替换特性之后所产生的一些奇怪的现象。第13章讨论了关于多重继承的常见的误解。

Ⅴ. 多态。面向对象编程的许多强大功能都来自于对各种形式的多态的应用。第14章介绍了面向对象语言用于实现多态的基本机制。之后的四章非常详细地研究了多态的基本形式。

Ⅵ. 多态的应用。第19章研究了多态最常见的应用之一，即用于公共数据结构抽象的类的开发。第20章也是一个案例研究，讨论了C++语言新增的STL技术。第21章介绍了框架（framework）的思想，这是软件复用常用的一种非常成功的手段，它也是基于多态技术来实现的。第22章描述了一种著名的框架，即Java抽象窗口化工具包（AWT）。

Ⅶ. 对象交互。从第23章开始，我们提升了抽象的层次，开始考虑类的更一般的关系，而不仅仅是父子关系。第23章讨论了两个或更多个类（或者对象）相互作用的方式。其中的很多相互作用都通过一种称为设计模式（design pattern）的形式来描述和定义。第24章讨论了设计模式的概念，并介绍了一些最常见的设计模式。

Ⅷ. 高级主题。最后三章讨论了一些对于这样一本介绍性书籍来说很高级的主题。其中包括：反射和内省技术（第25章）、网络编程（第26章）和面向对象语言的实现技术（第27章）。

我在俄勒冈州立大学教学这一门课需要十个星期的课时，上面描述的每个主题都大约需要一周的时间。这一课程的主要学生是高年级的大学生和一年级的研究生。通过这门课的学习，学生可以使用他所选择的面向对象语言来完成多个大小适中的项目，在一学期结束的时候，每个学生都可以完成多个项目的设计和编码工作。

介绍任何一个纷繁复杂的主题都会遇到如何对这些内容进行排序的问题，本书也不例外。

通常，我的方法是尽早地介绍这种技术思想，然后在后面的章节中再对其深入研究，这样做的结果是在读者第一次遇到这些概念时无法对其完全领会。虽然我比较倾向于这种排序方式，但是我也发觉，其他人可能更想使用另外的排序方式。尤其是某些教师认为关于软件工程的内容非常重要，并将其放到前面，与关于设计的章节（第3章）一起来介绍，而我却将其推迟到第23章。类似地，虽然多重继承也是一种继承形式，应该归类到继承部分，但是由于多重继承很难与多态一起讨论，因此当读者学习完多态部分后再来学习多重继承可能效果会更好。因此，教师在使用本书时可以自由地选择各个主题，并根据具体情况安排讲解顺序。

对读者的要求

我曾经声明过，本书的内容只适合那些已经了解某种传统编程语言（例如Pascal或C语言）的读者。在我的课程中，这些内容对于高年级的大学生（三年级或四年级）和一年级的研究生都取得了成功。对于某些内容（尤其是本书的后四分之一），如果读者有更深入的知识将会更好，但并不是必须的。例如，如果一个学生学习过关于软件工程的课程，将会发现第23章中的某些内容与其相关。如果学习过编译器结构的课程，将会发现第27章更容易理解。如果需要，这两章在讲解时都可以简化。

许多章节都用星号（*）进行了标识，这表示该章节为可选内容。这些章节可能会很有趣，但并不是本书想主要讨论的。它们通常是与特定的面向对象语言相关的，而不是所有面向对象编程都普遍具有的一些内容。可以根据教师和学生的兴趣，以及授课进度的安排，权衡是包含还是省略这些内容。

获取源代码

关于本书所列举的案例的源代码，可以通过匿名ftp从ftp.cs.orst.edu的/pub/budd/oopintro目录下获取。该目录下还包含其他内容，例如勘误表、各章的学习问题以及我在授课时所使用的关于该书籍的幻灯片。这些信息也可以通过我的个人主页http://www.cs.orst.edu/~budd来获取。如果还有其他要求，可以通过电子邮件budd@cs.orst.edu或者通信地址Professor Timothy A. Budd, Department of Computer Science, Oregon State University, Corvallis, Oregon 97331与我取得联系。

致谢

这里，我要感谢那些在1989年秋季使用本教材第1版、听取我在俄勒冈州立大学所开设的CS589课程的65个学生。他们通常都是在每次讲课的前一两天才收到相关章节的教学材料。他们的耐心令人钦佩，他们所提出的很多具体的建议、错误、批评、评论都非常有价值。这里尤其要感谢Thomas Amoth、Kim Drongesen、Frank Griswold、Rajeev Pandey和Phil Ruder等人所提出的详细的建议。

第9章中讨论的纸牌游戏是受到了Kim Drongesen所完成的项目的启发。第7章中讨论的台球游戏是基于Guenter Mamier和Dietrich Wettschereck的项目实现的。但是，我对这两个案例的所有代码都进行了重新编写。实际上，为了实现演示的目的，这两个案例的代码都被裁剪得非常少，根本无法与那些学生原来所完成的项目的规模相比。

对于一个作者来说，能够获得别人对其作品的独立的评价，是非常幸运的一件事。在这方面，位于比勒陀利亚[⊖]的南非大学的计算机科学与信息系统系的Arina Brintz、Louise Leenen、Tommie Meyer、Helene Rosenblatt和Anel Viljoen等人为本书第1版准备的学习指导材料提供了很多有价值的观点。

对于本书的第1版和第2版，无数的读者指出了错误或遗漏之处，并提出了改进的建议，为我提供了大量的帮助。在这里，非常感谢他们，并非常抱歉，无法将他们的姓名一一列举。

我非常感谢为本书第3版草稿提供建议的几个读者。他们是Ali Behforooz（陶森大学）、Hang Lau（康考迪亚大学，加拿大）、Blayne Mayfield（俄克拉何马州立大学）、Robert Morse（艾文斯维尔大学）、Roberto Ordóñez（安佐斯大学）、Shon Vick（马里兰大学，巴尔的摩县分校）和Conrad Weisert（Information Disciplines, Inc.）。根据他们的建议，我做了大量的修正，因此本书如果还存在错误都是因为我的原因，与他们没有任何关系。

Addison-Wesley出版社的Susan Hartman-Sullivan是本书第3版的编辑，他和他的助手Elinor Actipis都非常有能力，工作高效，而且很有耐心。Diane Freed协调完成了本书的最后一份拷贝。Paul Anagnostopoulos和Windfall软件公司的Jacqui Scarlott负责本书的排版和印刷。到目前为止，我已经与Paul和Jacqui合作出版了多本书籍，结果都令人非常满意，我对他们的感激之情无法言表。

⊖ 南非首都。——译者注

目　录

第1章 面向对象思想

我们现在所说的面向对象编程思想虽然早在20世纪60年代就产生了，但是直到20世纪80年代，面向对象语言才真正引起计算领域的普遍关注。这与下面这两个事件是密不可分的：一是 *Byte* 杂志的出版（1981年8月），*Byte* 介绍了 Smalltalk 编程语言；二是关于面向对象编程语言和应用的第一次国际会议于1986年在美国俄勒冈州的波特兰市举行。

现在，虽然已经过去20年了，但在本书的第1版（1991）中所描述的情形还依然存在：

> 面向对象编程（OOP）近年来已经广为流行。软件制造商们都迫不及待地发布自己产品的面向对象版本；出版社和期刊社出版了数不清的关于这方面的书籍和专著；学生们开始在他们的简历上列出“面向对象编程经验”。从这些狂热的行为来看，对面向对象编程的热情程度甚至比早期革命性的观点（例如“结构化编程”、“专家系统”）还要高。

本书在前两章里将通过研究和解释面向对象编程的基本原理来阐明以下两个观点：

- OOP是一种革命性的思想，总体说来，不同于以往的各种程序设计思想。
- OOP是编程技术前进的一个阶段，它是过去编程思想发展的自然产物。

1.1 为什么 OOP 这么流行

为什么面向对象编程能够成为近20年来主要的编程方法呢？其中有几点重要原因：面向对象编程的适用范围非常广，从最细微的问题到最复杂的项目，都可以通过它来解决；面向对象编程为技术人员提供了一种行之有效的解决问题的抽象方法；同时大多数主流面向对象编程语言都提供了大量的且越来越多的类库来支持各个领域应用程序的开发。

在众多针对“软件危机”的解决方案中，面向对象编程是最近提出的一种方法。概括地讲，软件危机是指我们通过计算机来解决复杂任务的难度几乎总是要超出我们的实际能力。

面向对象技术确实有利于构建复杂的软件体系，然而重要的是，不能认为OOP是万能的。计算机编程仍然是人们所必须担负的一项非常困难的任务。优秀的程序员需要天赋、创新、勤奋、逻辑思维、构建和提取抽象的能力以及经验——即使有最好的编程工具。

我认为，像C++和 Delphi（Smalltalk、Beta不同）这些语言十分流行的一个原因是：管理者和程序员们都希望，仅仅通过在职务头衔名称前加上几个字符，就可以把一个C程序员或者 Pascal 程序员轻易地转变成一个C++程序员或者 Delphi 程序员。但遗憾的是，从理想到现实还要走很长的路。面向对象编程是一种新的思考问题的方法，它着重于面向对象对于计算的含义，以及如何构建信息才能把我们的意图与其他人和机器进行顺利的交流。要想成为优秀的面向对象技术人才，需要对传统软件开发方法进行彻底的重新认识。

1.2 语言和思维

Benjamin Lee Whorf在他的 *Language, Thought & Reality*（语言、思维和现实）一书中讨论了语言学家 Edward Sapir 的观点：

> 人类不是孤立地生活在客观世界中，也不是孤立地生活在通常所认为的那种由社会活动构成的世界中，而是生活在处于特定语言环境下的社会里，语言已经成为人们与社会交流的工具。有的人认为不需要使用语言就可以完全适应社会，或者语言只是一种偶然的解决思想交流和反馈等特定问题的方法，但这些想法只是一种幻想。问题的实质是："现实世界"在很大程度上是无意识地建立在人们的语言习惯上的……我们看到的、听到的和得到的其他方面的经验就像我们真实做过一样，这是由于我们所处社会的语言习惯所导致的必然结果。

这个例子强调了这样一个事实：我们所使用的语言直接影响我们观察世界的方式。这不仅适用于自然语言，比如20世纪早期美国语言学家 Edward Sapir 和 Benjamin Lee Whorf 所研究的语言，而且也适用于人造语言，比如那些在计算机编程中使用的语言。

1.2.1 爱斯基摩人和雪

这是一个几乎在世界范围内广泛引用的关于语言影响思维现象的例子，尽管它可能是错误的。在爱斯基摩人（因纽特人）的语言里，用不同的单词来表达雪的不同形式——湿的、绒毛似的、厚的、冰冷的等等。这并不奇怪。任何一个有着共同兴趣的社会都会自然地产生一些特殊的词汇来表达他们所关注的概念（面对同样的问题，但气象学家使用特定的词汇来描述雪）。

重要的是，不应该过分夸大我们从这个简单的观察中所得到的结论。不是爱斯基摩人的眼睛在某些方面与我们有着显著的差异，也不是爱斯基摩人能够看到我们所不能觉察的东西。通过一定时间的训练，我们也能够区别不同类型的雪。但是我们所说的语言（英语）并不能迫使我们这样做，并且对我们来说，这样做也是不自然的。因此，不同的语言（比如 Inuktitut 语言）能够引导人们（而不是要求人们）去用不同的方式观察世界。

充分有效地利用面对象原则要求人们以一种新的方式来观察世界。但是，简单地应用一种面向对象语言（比如 Java 语言或者C++语言）本身并不能使我们成为一名真正的面向对象程序员。虽然使用面向对象语言会简化面向对象解决方案的开发，但是，具有讽刺意味的是，"FORTRAN 程序可以用任何语言来编写"（此处意味着使用面向对象语言也可以开发面向过程的程序，使用面向对象语言并不代表掌握面向对象思想。——译者注）这样一个不争的事实也同样存在。

1.2.2 关于计算机语言的一个例子

在人造的计算机语言中，我们所提到的语言与思维之间的关系比其在自然语言中体现得还要显著。也就是说，一个程序员选择哪种程序语言来考虑待解决的问题，会影响或改变算法的

设计。

下面是一个证明计算机语言和问题解决方案之间关系的例子。七年前，一名进行基因研究的学生承担了一项分析 DNA 序列的任务。这个问题可以简化成一个相对简单的形式。DNA 可以用一个由 N 个整数值组成的矢量来表示，其中，N 非常大（大约好几万）。任务是判断 DNA 序列中是否包含重复的长度为 M 的序列片断，其中，M 为固定的小数值常量（比如 5 或者 10）。

程序员通过思考写出下面这个简单而直接的 FORTRAN 程序：

```
    DO 10 I = 1, N-M
    DO 10 J = 1, N-M
    FOUND = .TRUE.
    DO 20 K = 1, M
20  IF X[I+K-1] .NE. X[J+K-1] THEN FOUND = .FALSE.
    IF FOUND THEN ...
10  CONTINUE
```

当试运行程序时，令人非常失望的是，程序要花数小时才能完成。于是他和另外一名同学讨论这个问题，恰好这名同学精通 APL 编程语言。她建议用 APL 编写程序来解决这个问题。前面的这位同学表示怀疑。毕竟，FORTRAN 被公认为是一种非常“高效”的编程语言。它是编译执行的，而 APL 仅仅是解释执行的。尽管持有一定的怀疑态度，但他还是采纳了这个建议，结果发现用 APL 写出的算法只需几分钟而不是几小时就可以解决这个问题。

用 APL 编程所做的事情就是需要重新考虑这个问题。把数据重新安排在一个 N 行 M 列的矩阵中，而不是前面的由 N 个元素组成的矢量。

$$\begin{array}{cccccc}
x_1 & x_2 & \cdots & & x_m \\
x_2 & x_3 & \cdots & & x_{m+1} \\
\vdots & \vdots & \cdots & \vdots & \vdots \\
x_{n-m} & & \cdots & & x_{n-1} \\
x_{n-(m-1)} & & \cdots & x_{n-1} & x_n
\end{array}$$

把矩阵按行排序（即把一行看作一个单元，在排序过程中移动整行）。如果有任何序列片断被重复，那么，在排序后的矩阵中相邻的两行就会有相同的数值。

```
  .  .  .
T G G A C C
T G G A C C
  .  .  .
```

检查如何满足这项条件很容易。APL 程序之所以快与 APL 语言本身无关，APL 程序之所以比 FORTRAN 程序快，是由于算法的不同；简单地讲，FORTRAN 程序使用了一个 O（$M \times N^2$）的算法，而 APL 程序使用的排序方案需要大约 O（$M \times N\log N$）次操作。