程序设计语言基础

〔美〕 Ellis Horowitz 著

裘宗燕 译

北京大学出版社

程序设计语言基础

[美] Ellis Horowitz 著

裘宗燕 译

北京大学出版社

内 容 简 介

本书系统讨论了程序设计语言的各种重要机制，介绍了变量、表达式、语句、类型划分、作用域、过程、数据类型、异常处置和并行性等基本概念及其在不同的程序设计语言中的实现；从控制结构、数据结构的角度分析了一批典型语言的特征及其优缺点。书中采用按语言特征组织材料的方式，内容丰富，编排合理。读者可以从中得到对各种程序设计语言的深入理解。

本书在IEEE提出的教学计划中很受推崇，被国外许多大学所采用，对大学计算机系师生及学习、使用和开发各种程序设计语言的读者是一本很好的教材和参考书。

程序设计语言基础
[美]Ellis Horowitz 著
蔡宗燕 译
责任编辑：李怀玺
*
北京大学出版社出版
（北京大学校内）
北京大学印刷厂印刷
新华书店北京发行所发行 各地新华书店经售
*
850×1168毫米 32开本 14.25印张 360千字
1990年2月第一版 1990年2月第一次印刷
印数：0001—4,000册
ISBN 7-301-00760-4/TP·008
定价：5.95元

前　言

"……我一直在做程序设计语言方面的工作，因为依我看，如果一个人不能理解程序设计语言，他就不能真正地理解计算机。理解程序设计语言并不是仅仅意味着会使用它们，许多人能使用它们但是并不真正理解。"

——Christopher Strachey

程序设计语言的发展是计算机科学这个新学科中最富于智慧的成就之一。至今为止，沒有任何我所知道的其他领域象它那样激动人心和神妙莫测。在论述这个充满变化的领域时我是非常谨慎小心的，虽然如此，作为一个教授，我仍然认为应该用一种全新的方法来处理这一个论题。

传统的关于程序设计语言的书籍好象是一些简化的语言手册，而本书则是从完全不同的观点出发的。我认为，要把握和理解今天的程序设计语言，一个可能最好的方法是把注意力集中于一批基本概念。正是这些概念形成了本书的纲，它们包括：变量、表达式、语句、类型划分、作用域、过程、数据类型、异常处置和并行性等等。通过了解这些概念是什么以及它们是怎样在不同的程序设计语言中实现的，我们可以得到对程序设计语言的深入理解。这样的理解远不是用某几种语言写几个程序所能得到的。进一步讲，关于这些概念的知识也为我们提供了认识未来的语言设计的框架。

这种研究程序设计语言的方法类似于语言学家研究自然语言。语言学家 Victorin Fromken 和 Robert Rodman（在"语言学导论"中）相应的说法是："语言学家企图找出一个语言中的规律以及属于所有语言的规律"。在他们的书里，他们试图标识

出一般的语法、语义和语音。语言学家认为他们自己的工作就是识别这些东西，而计算机科学家应该期望的更多。关于程序设计语言性质的研究可以促进我们对现存程序语言的深入理解，帮助我们在将来设计出更好的语言，本书所使用的讨论方法就是考虑到了这个目标的。

当 Jean Sammet 的书 "Programming Language: History and Fundamentals" ——它的著名的封面上是程序设计语言的 Babel 塔——出版的时候，FORTRAN 和 COBOL 分别占据着科学计算和商务计算的领域，FORTRAN 支配着教育领域。ALGOL60，PL/1，BASIC 以及其他一些语言也有人使用，但在用高级语言写的程序中这些语言的程序占得比重很小。今天的情况是很不同了，Pascal 和继它而来的语言在教育和微机应用方面占据了主要的地位；PL/1 的使用有了实质性的增长；UNIX 操作系统的成功造就了一批程序设计者，他们有自己的 (A，B 和) C；APL 继续吸引着一批信徒，他们用它去解决各种类型的问题；其他的许多语言也在继续为人们使用。这种多样性还在进一步发展着，新的语言正不断地出现在地平线上，而且它们都有着变成非常重要的实在的可能性，这里我要特别提一下美国国防部的新语言——Ada。如果说69年画出语言的 Babel 塔表示了我们想阻止语言数目快速增长的愿望，那么今天这个塔是更大得多了，这也使得我们在研究程序设计语言这个领域时必须集中注意力于那些基本的概念和论题。

用比较的方法研究程序设计语言是可能的吗？到今天，命令式语言已经有了25年以上的历史，对于它的许多概念我们已经有了很好的理解。对于另一些概念，虽然今天仍然在争论，但争论的问题已经是什么时候决定把它们纳入到一个语言中去了。所以，用一种分析的方法处理这些语言概念是可能的。还有一些概念，如异常处理和并行性，是较新的因而更少为人所理解。对于这些方面的处理就只能是给出问题，指出现存的支持这些概念的

语言特征的力量和弱点，不能提供比较完整的答案。

虽然这些年的工作大部分是在命令式语言方面，我们也不能小看函数式语言。作为一种非常不同的程序设计模型，它们也是必须研究的。而且 LISP 的长期而稳定的成功也说明了这类语言并不是只有理论价值。这些问题也应该在本书中得到反映。

本书的编写是遵循 ACM 课程委员会课程设计 CS8 “程序设计语言的组织”大纲的，关于这方面的细节参见 “Communications of the ACM” 1979年3月号。本书适用于已经有了一年程序设计经验的计算机科学专业的本科生，有 Pascal 的知识是最好的，PL/1,ALGOL60,ALGOL68 或其他类似的语言也可以代替 Pascal。这本书也曾成功地用在研究生一年级的课程中，目的是帮助取得对现有语言的重要特征的全面认识。

我曾基于本书给两类不同的对象上过课。第一类是由二、三年级本科生组成，他们只有 Pascal 和数据结构的知识，与此课同时进行的有一个实验课，指导老师先教 PL/1 然后教 LISP 并进行程序设计练习。在讲课中我直接使用书中的内容，例子主要选自 PL/1 和 Pascal，最近，我已经用 Ada 取代了 PL/1。第二类听众是由计算机系的研究生组成，没有实验课，但每个学生要负责某个程序设计语言的一个方面。例如一个学生给出 Ada 的概述，另一个讨论 Ada 中的包机制，还有一个讨论 Ada 的作业机制。还有人专门搞 Pascal,APL,ALGOL68,Euclid 等等，他们负责讲自己的语言的突出特征。通过与这些学生的密切合作，我感到能够对这些语言有一个统一的高水平的表述了。

当讨论一个语言时，查阅它的文本是非常基本的事，这使我收集了一批为这个课而用的语言文本。进一步，我发现加进一些著名研究者的文章是有益的。这个工作最后产生了一个文选并与本书一起出版了，它的名字叫作 “Programming Languages: A Grand Tour” （“程序设计语言：伟大的历程”）。选集中包括某些经典文献，如 “ALGOL60 报告”，某些引起了广泛争论的

文章，如 John Backus 的“Can Programming be Liberated from the von Neumann Style？”（“程序设计能从冯·诺依曼风格中解放出来吗？”）。还有 Pascal, Concurrent-Pascal, Ada, MODULA, C, ALGOL68, CLU, Euclid, FORTRAN77, LISP 和 APL 的概述。它还包含 Ada, ALGOL60, ALGOL-W, C, 以及 MODULA 的完整的语言文本。我希望读者和教师能觉得这是一本对本教科书有用的选集。

这里我还想就 Ada 语言讲几句话。在写这本书的那段时间里，我看着 Ada 从一个需求文件成长为一个标准化的程序设计语言，在研究这个语言的同时，我逐步地把它结合进这个课程。现在我发现我的许多例子来自 Ada，这主要是因为 Ada 是这样的很少几个语言之一，它们在某种意义下包含了所有我希望讨论的语言特征。所以，Ada 就能成为不同章节中例子的一个公共的线索。对 Ada 的这种使用促使我把它的完整文本放进了选集里。我认为 Ada 无疑地会在程序设计语言的市场上占据一个重要的份额，但是本书中这样地使用它表示的不是对它的赞同，而是对它的领地和它的逐渐增长的重要性的认识。

许多人对于本书的改进有着贡献，我想在这里对他们的帮助表示感谢。最主要的帮助来自听我的这个课程的南加州大学和以色列的 Technion 的学生们。John Guttag 和 Barbara Liskov 在 MIT 教授他们自己的程序设计语言课对我也是一个鼓舞。Richard Gillmann, Paul Hilfinger, David Jefferson 和 Richard Schwartz 仔细地审阅了本书，对我提出了许多具有指导性的意见，对此我深表感谢。学生 Alfons Kemper, Georg Raeder, Simon Wegner 以及 Ronald Williamson 也提出了一些很有帮助的意见。Brian Reid 和 Ray Bates 在使用正文编排系统 Scribe 上提供了帮助。我还应该感谢 USC 的信息科学研究所允许我使用他们的系统。

对于程序设计语言的研究是迷人的但绝不是容易的，我希望

这本书能帮助人们认清观察和思考问题的正确方向。

许多改进的建议已经体现在这个新的版本里，这应该感谢David Watt,Ira Steven和许多无名的评论者。

Ellis Horowitz
南加州大学
1983

目　录

第一章　程序设计语言的演化

“然后上帝说：‘看，他们是一群人，都使用同一种语言；而这只是他们要做的事情的开始。’”

《创世纪》11:6

一个程序设计语言是一套系统化的记法，我们用它描述计算过程。这里所讲的计算过程只是指一些步骤，它们可以由机器执行去解决某个问题。为了把一个问题的解决办法告诉计算机，我们就需要知道计算机可以理解和执行的一组命令。看一看今天计算机能做的各种各样的工作，人们会自然地发现计算机的内部机制是过分的原始，当一个计算机从装配线上下来时，它通常只能做算术和逻辑操作，输入和输出，还有某些“控制”功能。这些组成了计算机的**机器语言**。由于这样的语言与人们的思考方法和描述问题解答的方式相距太远，于是人们就构造出了所谓的**高级程序设计语言**。这些语言用的不是机器语言那样原始的记法，因此，需要有一个程序把它们的意思解释给计算机，这个程序一般不是计算机线路的一部分，而是作为**系统软件**的一个组成部分与计算机一起提供的。本书的目的就是研究为了迎合人和机器的需要，这些程序语言应当怎样设计。

程序设计语言的概念是相当新的。使人感到比较奇怪的是在历史上几乎找不到有关描述计算过程的方法发展的证据。在过去很多世纪里，数学家为描述静态的，函数式的关系开发了高度复杂的记法，二项式定理及著名的傅里叶变换都是很好的例子。但是就至今所了解的，直到很近的年代人类也没有设计出描述计算过程的形式的记法。当需要的时候，人们通常采用自然语言

和非形式化的风格。

现代数字计算机的发展大大激发了人们对计算过程的描述方法的开发。因此，程序设计语言也常被认为是为计算机描述计算过程的记法。因为人们必须构造和修改他们的程序，读其它人写的程序，所以程序设计语言必须是对人和机器都是容易理解的。

如果能找到一个单独的程序语言来描述我想讨论的概念那当然很好，但这不是可行的，因为没有一个语言以合适的形式包含了所有那些基本的东西。所以，在这本书中，只要值得讨论，我将不加限制地提到不同语言的特征。有两个语言是反复使用的，这就是 Pascal 和 Ada。当前，Pascal 已成为计算机科学教育方面的通用语言。它是只用了一个小的特征集合非常成功地构造出的强有力的和高效的程序设计工具。第二个我经常作为例子的语言是Ada，某些人会认为这是欠考虑的，因为在我写书的时候，还没有一个一般可用的 Ada 编译器。但无论如何，Ada的文档是可以用了[Ada79ab,Ada80]，而且 Ada 的入门书也正迅速地出版着。读这本书并不需要知道 Ada，事实上这本书可以部分地作为 Ada 语言的入门书。因为 Ada 实质上包含了现代程序语言的所有基本特征，在80年代用它作例子是合适的。

在第二章我们要讨论一个引起争论的话题：怎样评价一个程序语言的质量。但首先让我们考查一下程序语言的发展的简单历史，这会有助于我们更好地理解在程序设计语言这个领域中已经取得的进展以及我们所面临的情况。

1.1 早期的历史

已知最早的算法写在考古学家发掘出来的粘土板上，这些粘土板的年代大约是公元前1500—3000年，也就是大约3000—5000年以前。它们是在美索布达米亚（伊拉克）接近古代城市巴比伦

的地方发现的，那地方距离现代城市巴格达不远。巴比伦人使用六十进制系统，我们现在关于时、分、秒的记法就是从他们那儿来的。他们还使用了一种浮点记法，所以三个60进制数 8 ,50,36 可以表示

$$8\times60^2+50\times60+36=31,836$$

或

$$8\times60+50+36/60=530.6$$

或一般地

$$8\times60^k+50\times60^{k-1}+36\times60^{k-2}=31386\times60^{k-2}$$

以60为基数，乘和除很容易做，而估计以60为因子的数量级也不困难。

为了做数学用表，巴比伦人需要解代数方程，他们的做法是给出了一个求值的“算法”。虽然算法是一般的，但算法步骤中散布着对实际数目的计算。在最后他们写了一个短语，这短语可以不太严格地翻译为“这是一个过程”，这或许是第一次出现的程序设计语言的记号。

他们的算法中没有象“如果 x 小于 0 则分枝”这类的条件测试，因为巴比伦人不知道负数，甚至不知道零。要表达多于一种的可能性，他们就必须多次地写同一个算法。包含了重复的算法展开成若干步，直到解决了给定的问题为止。Donald Knuth 在关于古代巴比伦人算法的文章中[Knuth72]给出了关于这些的说明。这里是他的文章中的一个例子（稍微有点修改）：

> “投资了一个色克尔（注：货币名）之后，经过多少年利息将与本金相同？
>
> 你要照如下方法做：把8年的利息算出来，（本金和利息）放在一起超过了2个色克尔，这总数超出本金加7年利息乘上什么数可以给出总数减2？ 8,50,36(月)，从8年中减去8,50,36月就得到了所要求的年和月的数目。
>
> 这是一个过程。”

问题是求出要多长时间可以使投资加倍，用的是圣经中的货币单位。例子中假定利率为10%，因为

$$1.95 = 1.1^7 < 2 < 1.1^8 = 2.14.$$

答案是在7与8之间的某个地方。巴比伦人认识到在任一年中资产的增长是线性的，所以他们用内插法，以下面的公式求出比8年少的月数：

$$12(1.1^8 - 2)/(1.1^8 - 1.1^7) = 8.84.$$

这就是60进制的8,50,36，即算法给出的数字。巴比伦人描述的一般计算过程包括形成 $1+r$ 的幂，这里 r 是利率，直到找到第一个 n 使 $(1+r)^n > 2$。然后计算：

$$\text{diff} = 12((1+r)^n - 2)/((1+r)^n - (1+r)^{n-1})$$

最后的答案是“几年减去 diff 个月”。

其他古代民族也开发了高深的数学概念和记法，但却没有人在巴比伦的基础上发展能用于描述计算过程的方法。另一个值得在本节里提出的人是欧几里德，他生活在大约公元前300年。在他的《Elements》第7本命题1.2中，欧几里德陈述了一个计算两个整数的最大公约数的算法。他的方法给出的时间比前面讲的巴比伦人的东西晚一千五百年，但是却被许多人认为是最早的算法。为描述这个算法，欧几里德用了他自己的语言。因为那时人们还不知道零，也不认为数1是合法的除数，欧几里德只好重复地写他的方法来处理这些特殊情况。参照欧几里德原来提出的方式用英语写出的算法可以在[Knuth 69]中找到。虽然算法中有迭代，作为描述计算过程的记法，它的形式与巴比伦人的东西比较进步是很小的。更重要的，它的使用只是孤立的事件而不是一个连续发展过程之中的一步。

1.2 早期现代史

我们要考查的下一批研究者是19世纪到20世纪初的一些对计