程序设计语言的层次体系

程序设计语言分为低级语言和高级语言两类。"级"是指程序员与计算机对话的复杂程度。例如，就查询语言来说，程序员(经常是用户管理人员)只需要涉及到计算机"做什么"，而不必涉及到计算机"怎样做"。在高级语言中，程序员必须提供详细说明"做什么"和"怎样做"的指令，而软件工具将帮助简化程序员的工作。低级语言要求程序员详细说明由计算机执行的每一级操作。在使用高级程序设计语言时，程序员不必记住数据在主存中的存储单元。这是由系统自动做的。而在使用低级语言时，程序员必须在程序中提供相应的逻辑，以便监视数据在主存中的物理位置。

低级语言

虽然人们用 COBOL、PASCAL 和 FORTRAN 来谈论程序设计，但是只有一种语言能够在给定的计算机上执行。那就是所谓的机器语言。所有其他语言都要被编译(翻译成机器语言)，并且最终用机器语言来执行。

机器语言是很繁琐的，因为指令(又称操作码或 OP 码)和运算数据的单元是用二进制(一串 0 和 1)来表示的。符号语言(有时称作汇编语言或汇编级的语言)具有一个本质上与机器语言一一对应的指令系统。汇编语言的优点是它用助记符号而不是用 0 和 1 来表示指令。例如，汇编语言中直接用"A"来表示"加"指令，而不用一串 0 和 1 来表示。

在 1970 年之前，机器级和汇编级语言常常用于应用程序的开发，特别用于系统软件的开发。当时信息服务人员认为目标程序利用计算机的效率高。自那时以来使用高级语言无论在发挥人或是计算机的效率上都超过了机器语言。由于这一原因，多数现行的程序设计都是用高级语言来完成的。

高级语言

1.面向过程的语言

面向过程的语言(POL)是极其灵活的，可以用来刻划几乎所有科学的和商业的过程。程序顺序地编写指令，而系统则根据用户的规定进行处理。例如，工资系统就是使用 POL 来编定的。除非由程序逻辑控制去做其他事情，否则程序中的每一条指令都是按顺序执行的。在一个工资单生成系统中，执行程序指令的特定顺序取决于被处理的职工的工资计算方法(是按月发工资，还是按工时计算工资)。对每个职工都要重复该程序流程的顺序。有三种主要的 POL：科学计算的、商用的以及多用途的。下面我们分别讨论每一种 POL。

(1)科学计算的语言。科学计算的语言是代数或公式化的语言。这种语言是为了满足典型的科学计算处理要求(矩阵操作、精度计算以及其它方面)而设计的。第一个而且仍然是最为流行的科学计算语言是 FORTRAN(公式翻译程序的缩写)。尽管在没有商业语言时(在计算机不支持任何其它语言的情况下)FORTRAN 已作为一种商用语言来使用了，但是 FORTRAN 的商业处理能力是有限的。

APL 语言很快地抓住了一些用户(主要是工程师)。APL 是唯一使用带有专用符号的交互式键盘来编写程序的语言。

(2)商用语言。目前很流行的第一个商用语言是 COBOL(面向商业的公用语言的缩写)。它是一种功能很强而又极为冗长的语言。发展 COBOL 语言的前提是：该语言的语句应该近似于英语。有些程序员发现语言使用起来很麻烦。然而，COBOL 语言尚处于方兴未艾的时期，而且被广泛地接受。今天，人们正在继续对它进行改进。在若干年内，COBOL 仍将是一种流行的商用语言，但是使用该语言的相对百分比将会下降。COBOL 最适合于具有循环处理周期的环境(例如打印工资支票)以及数据操纵量相当大的环境。

美国国家标准研究所(ANSI)已经对 COBOL 和其他语言建立了标准。建立这些标准的目的是使得在一台计算机上编写的程序可以移植到另一台(即另一个厂商制造的)机器上去。不幸的是，ANSI 标准很少被遵循；因此，COBOL 程序只是部分可移植的。

报表程序生成程序(RPG)可能是仅次于 COBOL 的最为流行的商业语言了。最初设计的 RPG 是在 IBM 公司数据录入级的计算机上使用的。它用于一些成批处理环境的小型商业工作。RPG 与其它 POL 不同之处在于程序员还必须通过选择所要求的程序设计特色来说明某些处理要

求(例如何时打印小计记录的选择等等)。

BASIC 设计者的最意图体现在该语言的名字上——初学者通用的符号指令码(Baginers All-Purpose Symbolic Instructional Code)。最初它被当作讲授程序设计的一个工具，但是在这个"初学者"语言的能力被充分扩充后，它变得如此流行，以致目前 BASIC 是小型计算机系统上用于应用程序设计的一种上要语言。在大系统上也使用扩展 BASIC，但不是用于生产系统。由于 BASIC 通常也用来进行少量的科学计算，因此，有人可能把它划归为多用途 POL 一类。

(3)多用途语言。多用途语言对于商业和科学计算是同等有效的。最为明显的多用途语言是程序计算语言 1/(即 PL/T)。PL/1 是在 1956 年由 IBM 公司普制出来的，但是，与任何一种语言一样，它需要几年的时间来排错。许多公司(主要是 IBM 公司的用户)，已经采用 PL/1 作为它们所使用的唯一的 POL。在刚刚引入 PL/1 时，由于它克服了现有程序设计语言的许多缺点而受到欢迎。然而，PL/1 仍然没有得到(原先期望那样)普遍地接受。其原因并非它的质量和能力，而是由于许多公司已花了巨大投资用 COBOL 和 FORTAN 编制了大量应用软件。同时也由于使用这两种语言的势头还在增长。

以最快速度发展起来的语言是 PASCAL，它是以 17 世纪的数学家 Blaise Pascal 来命名的。PASCAL 被看成是一种最新的 POL。尽管只有 1%-2%的商业生产程序是用 PAS-CAL 写的，但是它的能力、灵活性以及自我说明结构是不可忽略的。致使这个语言被广泛接受(而且接受面还在继续增大)的原因或许是因为绝大多数的学院和大学的计算机科学教程主张将 PASCAL 作为未来的 POL。随着毕业生将这种主张带进商业界，目前它已经引起了各界的兴趣。ADA 是新近引入的一种语言，它是美国国防部开发的一种多用途语言。尽管只有少数人知道和理解 ADA 的用法，然而对于它能否被广泛接受(不仅在军界，对其他部门也一样)。这一点，它的设计者们是很乐观的。

2.面向问题的语言

面向问题的语言，是专门为了满足某种特定应用或解决特定问题的一组语句。面向问题有的语言不要求像面向过程的语言的那种详细说明。例如，有几种面向问题的语言就是专为统计分析而设计的。这种语言的用户将注意力更多地集中在输入和输出上，而不是在数学上。数学是嵌入在语言中的。

已经为几十种应用设计了各种面向问题的语言，这些语言正在被用户使用。这些应用是：离散和连续模拟(例如，GPSS、SIMSCRIPT、GASP-IV)，程控机器刀具(例如，APT)，辅助工程师进行建筑和桥梁上的受力点分析(例如，GOGO)，辅助系统分析(例如，SAS)以及辅助办公人员进行字处理(例如，SCRIBE)。

面向问题的语言具有做统计、字处理和任何语言所满足的应用的灵活性。然而，一种面向问题的语言总是局限于某种应用。

3.查询语言

与其它高级语言相比，用户和用户管理人员更喜欢使用查询语言。然而，根据不同的情况，某个用户可能会发现某种特定的面向问题的语言对他更有用。查询的语言是朝着用户环境发展的典范。一个用户管理人员经过几个小时的培训和实践就能有效地使用一种查询语言。然后，他用很短的时间就能从一个信息系统中抽取信息或产生出一张报表。而通常这点时间只够用来向一个系统分析员或程序员说明技术要求。用户只需说明做什么，而查询语言软件将自动规定怎样做。

查询语言利用高级的、类似英语的命令来检索和编排满足管理查询和制表要求的数据。用查询语言可以交互式地完成一次询问(直接与计算机通信)。执行程序产生的输出直接在终端上显示出来，或者产生一份硬拷贝。查询语言具有以下特色：类似于英语的命令，对数据的有限的数学运算操作，对报表的自动编排、排序以及按关键字挑选记录等。

4.应用程序生成程序

尽管应用程序生成程序的概念至今没有严格的定义，但是目前已经使用的那些应用程序生成程序的目标以及各种开发步骤的目标是相同的。即它是不需要过程级指令就能够说明开发一个信息系统的所有程序设计任务的一种语言。某些应用程序生成程序(通过与程序员进行交互式对话)已经接近这一目标。当用应用程序生成程序还在早期的开发阶段。现有的应用程序生成程序并不具备面向过程语言的那种灵活性，而且不能用来开发完善的信息系统。然而当用于预期的用途时，它将能成倍地提高程序员的生产效率。当它们成熟时，在信息系

统的开发中应用程序生成程序将起着越来越重要的作用。

未来的语言

程序设计语言的进一步发展是自然语言。要采用的那种自然语言对程序员只需要很少的(甚至不需要)程序设计训练。程序员将直接写或口述程序功能说明书，而与程序设计的结构和语法(产生程序指令的规则)无关。

目前研究人员正在致力于开发自然语言。在开始时自然语言将带有某些语法限制。虽然很难用几句话来概括未来的程序设计语言，但是可以预见未来的语言将是一种可以不受限制地在个人与计算机之间会话的语言。