面向 21 世纪课程教材

程序设计语言原理

钱树人　编著

高 等 教 育 出 版 社

内 容 提 要

本书是教育部"高等教育面向21世纪教学内容和课程体系改革计划"的研究成果,是面向21世纪课程教材和教育部高等学校计算机科学与技术"九五"规划教材。本书从横向的角度对较常见的程序设计语言中的基本概念、基本设施、基本成分、各种风范、设计要求及相关的理论进行多侧面多角度地讨论和分析,使读者掌握程序设计语言的共性和各类语言之间的差异。全书共有十一章:引言、程序设计语言的刻划、词法成分、数据成分及其加工、控制成分、抽象和程序结构、并发、预定义成分和语境、逻辑式程序设计语言、函数式程序设计语言和对象式程序设计语言等内容。本书的结构清晰、层次分明、语言叙述流畅,适合作为高等院校计算机有关专业的教材,也可供从事计算机软件开发工作的技术人员参阅使用。

前　言

程序设计语言伴随着计算机的产生而产生、发展而发展，可谓历史悠久，种类繁多。常见的程序设计语言也不下于二三十种，而且仍在继续推陈出新，不断发展。绝大多数的程序设计语言教材着重于介绍某种程序设计语言的内容和使用方法，即着重于程序设计，如 Pascal 程序设计、Java 程序设计、C++ 程序设计、C 程序设计、Fortran 程序设计和 Ada 程序设计等，而且随着语言版本的更新升级，又有相应教材出现。这类教材对使用者来说是重要的，故且称为纵向教材。

本教材与上述的纵向教材有很大的不同，故且称为程序设计语言的横向教材。其特点是对较常见的程序设计语言中的基本概念、基本设施、基本成分、各种风范、设计要求及相关的某些理论进行综合地、多侧面多角度地较深入细致地讨论和分析，其目的并不在于立即使读者采用指定的程序设计语言进行编程，而在于更高层次的学习和理解，使读者掌握程序设计语言中的共性和各类语言之间的差异。这对读者提高程序设计语言及软件的素养，选择合适的程序设计语言，以及对研究、设计和实现程序设计语言是很有好处的。这门课程主要是为计算机专业高年级学生开设的，当然也可用于有关的计算机工作者。

程序设计语言原理是计算机专业的一门专业课。学生在上这门课之前至少已系统学习和实践过一门(如 Pascal、Fortran、C、C++、Ada)程序设计语言，已具有一定的相关概念、知识和编程能力。通过本课程的学习可使学生对程序设计语言的各个方面有相当广度和一定深度的理解。该课程主要是横向综合，它处于实践和理论的接合点。

与层出不穷、迅速发展的用于各种用途的程序设计语言相比较，本课程内容相对稳定，但达到科学性、先进性、实用性、综合性的融合和统一是很困难的。其困难处在于：需要适应读者群的希望和要求；需要在量大面广、发展迅速的程序设计语言中提取关键概念和设施内容；解决好实用性和理论探讨之间的关系；解决好课程和语言发展中的衔接；等等。

程序设计课程在计算机专业中一般开设的有 Pascal 程序设计、C++ 程序设计、Java 程序设计等，因此存在着课程之间的配合和衔接。

本课程中涉及的内容主要有下列几类：

(1) 程序设计语言中所包含的主要机制和设施；

(2) 有关各种机制和设施的概念、含义、描述以及分析；

(3) 有关程序设计语言的发展和评测；

(4) 程序设计语言的风范、刻划方法，以及与分析执行等实现阶段有关的某

些关联问题;

(5) 有关程序设计语言的若干形式描述方法和形式化技术;

(6) 除了对命令式语言作详细分析阐述外,还对具有特色的作用式语言中的函数式语言和逻辑式语言作比较详细的介绍,并简要分析介绍了对象式语言。

全书共十一章:第一章引言;第二章程序设计语言的刻划;第三章词法成分;第四章数据成分及其加工;第五章控制成分;第六章抽象和程序结构;第七章并发;第八章预定义成分和语境;第九章逻辑式程序设计语言;第十章函数式程序设计语言;第十一章对象式程序设计语言。

从内容看,第一、二章是独立的部分,主要讨论程序设计语言的概况和描述基础。第三章到第八章以命令式语言为主要对象进行多层次、多角度的功能分析和讨论。第九章和第十章介绍作用式语言中最具影响和典型特色的两类风范迥然的语言。第十一章扼要介绍分析对象式程序设计语言。本课程的内容具有积木式特点,可以根据学时和同学接受能力加以适当的组合。本课程的总时数在 60 学时左右,即一学期的课程。

本书是在全国计算机有关教材领导小组的指导和组织下,以及在南京大学计算机科学与技术系的领导、前辈和同事们的帮助支持和鼓励下完成的,在此表示衷心的感谢。特别是对支持和鼓励完成本书的费翔林教授致以谢意。本书初稿已试用过多次,在作者长期讲授‘程序设计语言”及其相关课程的基础上听取有关意见后修改成本教材。由于各种原因,一定存在许多缺陷和不当之处,敬请赐教。

对在本教材中有些章节引用了附录中列出的著作中的某些内容谨在此向各位作者致以深深的感谢和敬意。

全书的初稿由复旦大学计算机系夏宽理教授审阅,并提出了宝贵的意见,在此致以感谢和敬意。

编 者

2001 年 2 月

于南京大学

目　　录

第 1 章　引　　言

1.1　程序设计语言的作用

1.1.1　程序设计语言在语言层次中的特征

人类之间进行信息交流,最主要的方法是语言和言语交流。语言通常指书面语,而言语则是指口头语。相对而言,语言比言语的结构和表达规范,对语境语姿的依赖远比言语的依赖要少。人和计算机之间的信息交流则是自计算机诞生之日起即存在,其复杂程度、方式方法、表达形式日渐多样化,性能也愈来愈高。程序设计语言是人机交互的一种表达方法,是使用计算机的一种工具。它比人类之间的自然语言简单而比机器语言复杂,同时目前许多软件的描述和使用均离不开它。对它的深刻全面理解有助于计算机领域中的许多问题的研讨和解决,它是建立计算机精确处理模型和人类智能处理模型之间的某种形式的桥梁(或称界面)。当然,目前计算机能处理的语言仍主要是字符串式的语言,尚不能很好地使用和处理其他语言(如,图形语言、形体语言、声像语言等等)。

语言层次表示如下:

(1) 自然语言
(2) 受控自然语言
(3) 类自然语言
(4) 描述语言
(5) 程序设计语言
(6) 形式语言
(7) 机器语言

自然语言、受控自然语言、类自然语言可以归属于自然语言类,而描述语言、程序设计语言、形式语言、机器语言则可以归属于人工语言类。从人机角度看,自然语言最容易被人接受,而机器语言则最容易被计算机接受。上述语言层次表明了逐渐逼近人机交互的自然化和形式化的层次特征。

自然语言是人类日常使用的语言,如中文、英文、日文等等。它的特征有:词汇量大、结构复杂、变化多等等。从理论上看存在着许多的“实际无限性”和“开

放性”。如词汇集在不断的扩大,有新词、死词存在;并不存在一个语法规则集合包含所有的语言结构;而且语言结构也在不断地变化。

受控自然语言是对自然语言加以某种约束。例如,所考虑的自然语言仅限于某个领域,所使用的词汇是某个有限集,所使用的句子的语法结构是某个有限集,所使用的句子的语义理解的深度广度有一定的限度等等。这种受控的自然语言不但是目前许多软件的处理对象,而且还可作为描述工具中的某些成分。

类自然语言则比受控自然语言有更多的限制。所以 COBOL 语言中的语法规则表示即属于此类。其语法规则允许用户写出的符合 COBOL 语法但与自然语言语法的句子有相当大差距的源程序。这种“English-like”正是类自然语言的例子。

描述语言的含义极广。软件工程中的各个阶段均有相应的软件支持,这种支持软件均有相应的描述语言与之对应。数据库、网络等等均有相应的软件,这些软件也有不同程序不同格式的描述语言与之对应。这种描述语言有的和一般意义上讲的程序设计语言很接近,有的有较大的不同。总的说来,和程序设计语言有密切关系。

程序设计语言是指用于描述源程序的语言。它有有限的语法规则集,有限的词汇集,有严格的语义解释。其种类很多。

形式语言是研究自然语言和人工语言(如程序设计语言)的数学工具,它只研究语言的组成规则而并不研究语言的含义。

机器语言是指“纯”的裸机语言和汇编语言,它和计算机的指令系统密切相关。

上述分层方法只是笼统的不是绝对的。各层彼此之间存在着联系。可见程序设计语言处于语言层次中的中间,对其理解和研究有其特殊重要性。其内容、表示方法和概念具有承上启下的作用。

1.1.2 程序设计语言在软件描述中的作用

软件工程是计算机软件的重要学科,它有理论、方法、技术、系统和目标分析、梯队组织、环境工具等多方面,它研究和指导着整个软件生成的全过程。针对软件生命周期各个阶段的描述,有不同的模型存在。不管是什么样的模型,有多少个阶段,阶段之间又是如何过渡的,均会存在着相应的支持系统。CASE 系统即是其例,CASE 系统在不同的阶段有相应的软件予以支持。例如,需求阶段的 PSL/ PSA,粗设计阶段的 DFD 和 SC,细设计阶段的 PDL;编程阶段的各种程序设计语言;测试和维护阶段的某些软件(如测试数据的自动生成软件等)的描述语言。

应用软件包含了许多领域。例如,数据库描述系统,从建模到操纵均有相应的命令使用,这种命令的形式和作用,使用和语义往往和程序设计语言有密切的联系(例如 RPG 程序即是一例),有的描述语言亦可视为一种程序设计语言,如第四代语言。

1.1.3 程序设计语言的基本功能和分类

一、基本功能

程序设计语言出现早、种类多、标准化早且完整,至今有的语言仍在不断地出现新版,而且不断地出现新的程序设计语言。

在大百科全书的词条“程序设计语言”中说“程序设计语言是用于书写计算机程序的语言。语言的基础是一组记号和一组规则。根据规则由记号构成的记号串的总体就是语言。在程序设计语言中,这些记号串就是程序。”程序设计语言包括多个方面,如语法、语义、语用、语境等。语法表示程序的结构或形式,即表示构成语言的各个记号之间的组合规律,但不涉及这些记号的特定含义,也不涉及使用者。语义表示程序的含义,表示各个记号的特定含义,也不涉及使用者。语用表示程序与使用者的关系(包括数据错误、运行错误)。语境则表示程序理解、执行和实现的环境。这四个方面是相辅相成的,其中,语法、语义最为重要。

语言的种类千差万别,但仍可概括出若干特征。一般说来,语言中应有五类成分:数据成分,运算加工成分,控制成分,传输成分和程序结构成分。程序结构成分用于将前面四类成分组成源程序,故有时并不将此作为基本成分看待。但实际上,它与源程序的组成形式和各种成分的语义理解以及设计方法学有很大的关系,作为一种成分理解是适当的。数据成分用于描述程序中所涉及的数据,用于刻划其类型,可取接度(可见度),分配特征等。运算加工成分用于描述程序中的运算和加工操作。有的运算是程序设计语言中预定义的(例如,算术运算符+、—、×、÷……),有的加工操作可以通过语言中提供的设施来自行定义(例如,函数说明和函数命名符)。运算加工成分中应当包括值的初置和改变操作。控制成分主要用于描述执行流程,主要有串行和并行两种方式。串行流程最为基本,并行流程最近研究逐渐多起来。异常(包括实时)则是较为特殊的控制成分。传输成分在程序设计语言中往往处于特殊的地位,形式多样,刻划深度亦有不同,语法描述和语义解释亦有不同。

二、程序设计语言的分类

如何分类?这依赖于分类的准则和需要。早期的机器语言、汇编语言和符

号汇编语言均曾被称为程序设计语言,故后继出现的 ALGOL、Fortran、COBOL 等程序设计语言则称为高级程序设计语言,以示区别。而现在则很少出现“高级”字眼,这可以认为这种分类标准已失去实际需要。

程序设计语言品种很多,风格迥异。按“中国大百科全书”上的划分方法可列出如下几种:。

按编程用户角度划分,可分为过程式语言和非过程式语言。过程式语言亦称为强制式语言或命令式语言。过程式语言的主要特征是用户(指程序员)指明一列可顺序执行的运算,以表示相应的计算过程。换言之,过程式语言需要指明其执行流程。非过程式语言的执行流程由系统决定而不是由用户来指明。

按应用范围划分,可分成通用语言和专用语言。应用范围比较广且不局限于某个应用范围的语言通常称为通用语言(如 Fortran、COBOL、C、C ++、Ada 等)。仅用于很窄的领域的语言,这种语言称为专用语言(如 APT 等)。通用专用区别有时并不非常严格,例如早期的 COBOL 语言主要用于数据处理,早期的 Fortran 语言主要用于数值计算,而 Pascal 语言则主要用于软件系统的编程和软件教学,一般这三种语言均称为通用语言。随着发展,各语言的版本不断地更新(兼容发展),机制设施增多,功能扩大,因此其应用领域也愈来愈广。

按使用方式划分,有交互式语言和非交互式语言之分。具有反映人机交互作用成分的语言称为交互式语言,如 BASIC 语言即是一例。当然交互功能是要由实现系统支撑的。如果没有反映人机交互作用成分的语言则称为非交互语言。非交互式语言占程序设计语言的多数。

按语言的成分或设施划分,可以分成顺序语言、并行语言、实时语言等等。只含有顺序成分的称为顺序语言。如果含有并行成分则称为并行语言。如有实时成分则称为实时语言。由于语言中常常有多种特征的设施,故会出现如实时并发语言等说法。

按语言的组成方法来看,可以分成汇集式语言和可扩充语言。语言中的成分应有尽有,足以满足各类用户的要求,这种语言称为汇集式语言。例如,PL/ 1 就是集 COBOL、Fortran、ALGOL 的所有设施于一体,从而引进了这一概念。所谓可扩充语言是指语言由一个基本成分集和若干个可扩充设施集组成。从当前的情况来看,语言的发展趋势是两者结合,即既有功能相当齐全的成分又有可扩充设施。可扩充设施包括预定义成分的可扩充设施和用户自定义设施,前者往往由实现环境提供设施来完成,后者由语言提供设施由实现系统来完成。当前语言大多兼有汇集式和可扩充式特征。

程序设计语言按“代”的划分一般分为:

第一代语言:机器语言

第二代语言:汇编语言

第三代语言:高级语言

第四代语言:超(甚)高级语言

第五代语言:多范式语言

第一、二、三代语言已经得到认同,但第四代、第五代语言并未得到一致看法。有的认为第四代语言主要指与数据库系统有密切联系的编程环境(平台),第五代语言则指作用式语言。目前来看,基于冯氏体系的过程式(命令式)语言及其相应的平台有强盛的生命力,在可见的一段时间内在语言市场上仍会占有绝对优势的份额。

1.2 程序设计语言的发展动力和发展趋势

1.2.1 程序设计语言的发展动力

从程序设计语言出现到现在,种类数量增加迅速,功能不断扩大,版本也在更新,出现了兼容式和汇集式的发展趋势。那么这种经久不衰的发展动力何在呢?归结起来,包括如下因素:应用领域的扩大和深入;用户需求不断增加;新计算机体系(包括软件硬件等)的飞速发展;技术、方法、理论的新发展。从厂商角度看,商业竞争是其重要原因,乃至是主要原因。没有厂商的支持,任何一种程序设计语言都难以健康地发展和生存。没有用户的支持和使用,也难以推广和取得良效。换言之,用户和厂商(市场)是主要的发展动力。

计算机已经成为技术发展乃至社会发展的重要因素之一,计算机在许多领域中已经成为不可缺少的工具。因此对这些领域中出现的问题需要进行抽象,构成具有适当逼近程度的模型,研究其处理的算法,用适当的设施和机制进行刻划。因此有的将推出新的语言,有的对原有的程序设计语言进行分析,判定功能是否满足条件,性能是否符合要求。大多数语言采用功能的“重组”和“扩展”,采用兼容式的发展道路,推出新版本。例如,在许多老的程序设计语言中并未提供适当的设施来刻划实时要求,而这在应用领域中却是很有用的。为了提高性能,出现了并行成分。为了处理异常,引入异常处理设施。以及引进线程和对象引用概念等等。

用户需求在市场经济中更是至关紧要的,这也是商业竞争中的重要依据。目前许多程序设计语言的功能是相互渗透的,也就是厂商尽可能地吸收其他语言中有吸引力的功能放入自己的程序设计语言系统中。换言之,用户需要的喜欢的设施和功能,厂商往往采用兼容式的方法吸纳之。

新的计算机体系近来发展得很快。特别是分布式、网络、多媒体和人机交互等各方面的技术和设施发展尤为迅速。需要引进新的机制用于隐式或显式地提供给用户,以提高功效扩大功能。

技术、方法、理论的新发展往往对程序设计语言有相当大的推动作用。例如,结构化设计思想就是一个例子,它的出现不但促使新的程序设计语言的出现,而且对已有的程序设计语言的更新也有推动作用。语言中兼容式地引入新的成分和语句括号即是例子。与程序设计语言有关的用户有许多类,各类人员对程序设计语言有着不同的需求,这种商业因素、市场驱动的动力是不可忽视的。最终用户要求适用、可靠、高效、低耗;开发人员要求方便、面向对象、结构化、易移植、易改动、易集成;管理销售人员则希望易推广,具有兼容性和开放性;理论工作者则希望可验证,易形式化等等。总之,程序设计语言的许多功能就是从不同侧面不断发展,相互推动,综合推进的。换言之,不同的人员有不同的侧重,商品和产品是各种要求的综合。流行的受到广泛注意的程序设计语言通常是综合实力的体现。

1.2.2 程序设计语言的发展趋势

程序设计语言从出现到现在已有半个世纪的历史,种类很多,了解其发展趋势特征有利于对它的掌握和分析。这种发展有许多方面,例如,包括技术发展、使用发展、市场发展、理论发展等,同样也包括发展途径、发展方式、发展措施等等。

程序设计语言发展很快,一种方式是推出新语言,另一种更常见的方式是版本的升级(包括 ISO、国家标准、部门标准、企业标准等版本)。在发展中长期保持有生命力的语言为数并不很多,只有得到广泛应用、用户认可、厂商支持的才最有生命力。

程序设计语言的发展主要采用兼容方式。所谓兼容发展,是指语言版本中的功能愈来愈多。某程序设计语言一旦被广泛应用,那么就有许多软件基于它而存在,这就决定用户不希望由于该程序设计语言的升级而使原来的程序不能使用。这种用户愿望成为市场需求,从而决定了程序设计语言采用兼容式的发展方式。

程序设计语言功能扩充的主要方式有两种。一为引进新的功能。这时通常采用与原语言风范相一致的表示,对语法语义语用进行全方位扩充。这种新功能可能是本质扩充。二为对语言中的已有成分进行扩充。这种扩充可能涉及语法语义语用,也可能仅涉及到语义语用,其大多是功能的部分扩充。

发展中对已有成分的修改和删除持极为谨慎的态度,原则上尽量保留。对

于公认的不值得使用的成分常采用“过时”处理而不采用“删除”处理。所谓“过时”处理是指语言文本中虽然保留这些成分且在实现系统也仍然保留这些成分的实现,但在文本中却提醒用户不要使用或尽量少使用这些成分。例如,GOTO语句就是典型的例子。在版本的升级中往往会对原版本中的问题(包括语义不明确的地方)加以澄清,对若干语义进行增加、删除、修正,这种改动一般不涉及语法,从而使必须改动的地方保持在最小限度内,或者说,使升级版本引起的不一致性减低到最小。

程序设计语言中的设施、表示、描述方法往往被其他软件描述语言作为蓝本而借鉴。程序设计语言的适度复杂性和它的地位使它成为开展理论研究和算法研究的主要对象之一。

从程序设计语言总的发展过程来看,是沿着面向过程→面向数据→面向对象→面向目标→面向认知前进的。根本性的发展是面向应用,面向用户,面向市场的。

1.2.3 与程序设计语言相关的若干领域

程序设计语言的研究和其他领域和学科常常有密切的关系。

语言学中的许多概念被引进到程序设计语言中。语言学的语法、语义、语用和语境的概念早已出现且已有相当深入的研究,其中的许多结果和概念已被引入,有的将可能被引入而促进程序设计语言的发展。程序设计语言中,语法概念远比语义、语用、语境概念研究深入。但是一个正确的源程序仅从语法角度认识是远远不够的,必须从全方位来认识,这就离不开相应的语义、语用、语境。

程序设计语言和自然语言之间有着巨大的差别。采用计算机来处理自然语言是十分自然的。计算机处理是基于工程观点的,计算机所能处理的自然语言必然是自然语言的一个子集(故且称为计算自然语言)SNL。那么,SNL的范围多大?描述能力如何?特征是什么?如何使计算机能够处理?处理的目标是什么?处理的方法是什么?诸如此类的问题正是自然语言处理、计算语言学、机器翻译等领域中需要研究的。其概念和方法,技术和成果有不少和程序设计语言有关。

程序设计语言和形式化研究也有着密切的关系。目前形式化研究的对象主要是程序设计语言,或其子集或其模型。

程序设计语言和许多基础理论学科还有密切的关系,也与认知机理等学科已开始有联系,当然与计算机应用和技术发展更有联系。

1.3 程序设计语言的回顾

首先通过图 1.1～图 1.3 可以看到一些程序设计语言的发展过程。这些图表既不完整全面也不能保证其完全正确。因为程序设计语言家族实在太庞大，情况也非常复杂，需要精确地分析统计是非常困难的。

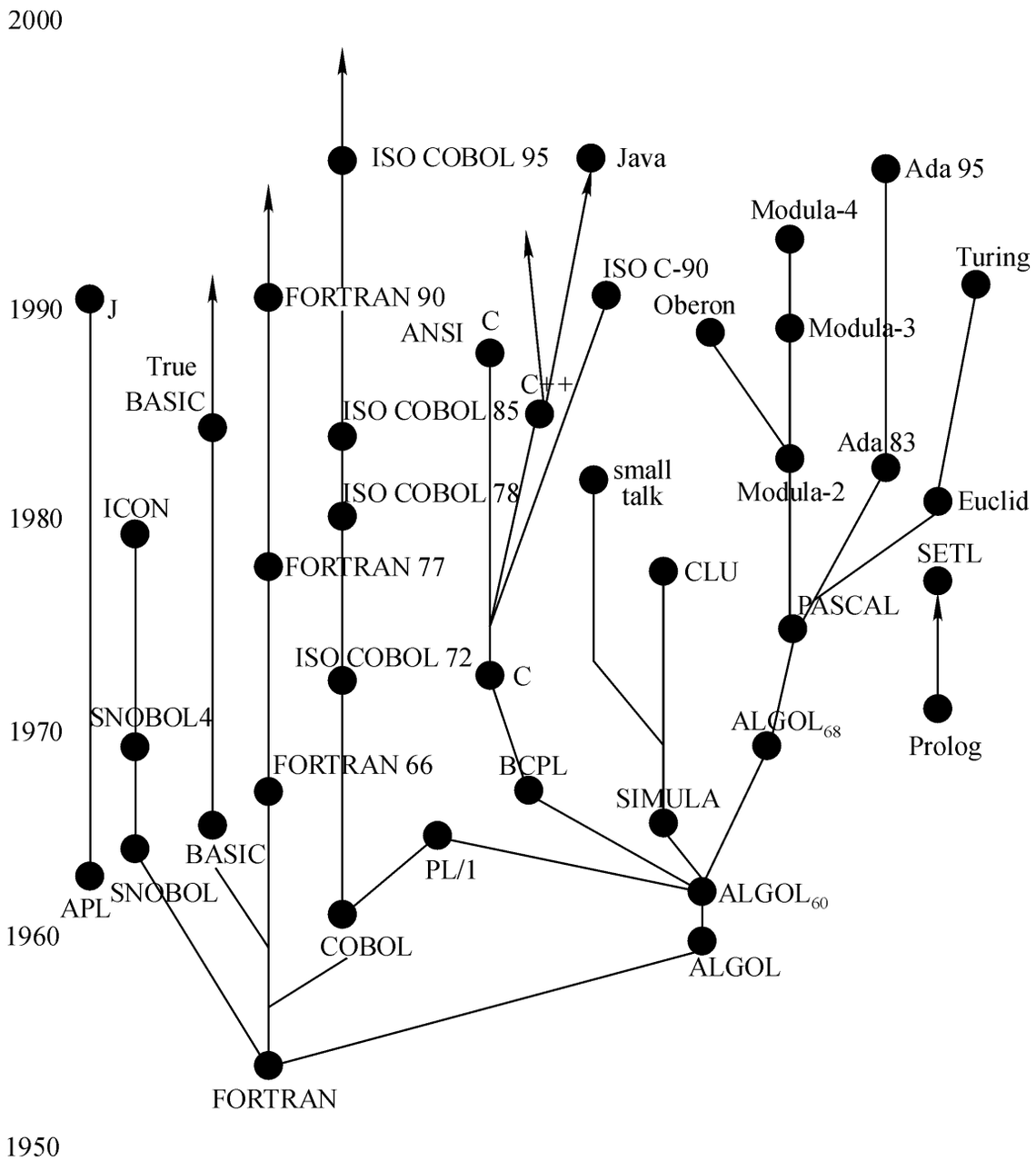

图 1.1

其中所述及的若干语言名称在本节后面加以叙述。这里列出的只是其中的一小部分，但从中已可以看到程序设计语言队伍的庞大和复杂。

从 20 世纪 50 年代中期开始出现比机器语言和汇编语言更接近于人的所谓