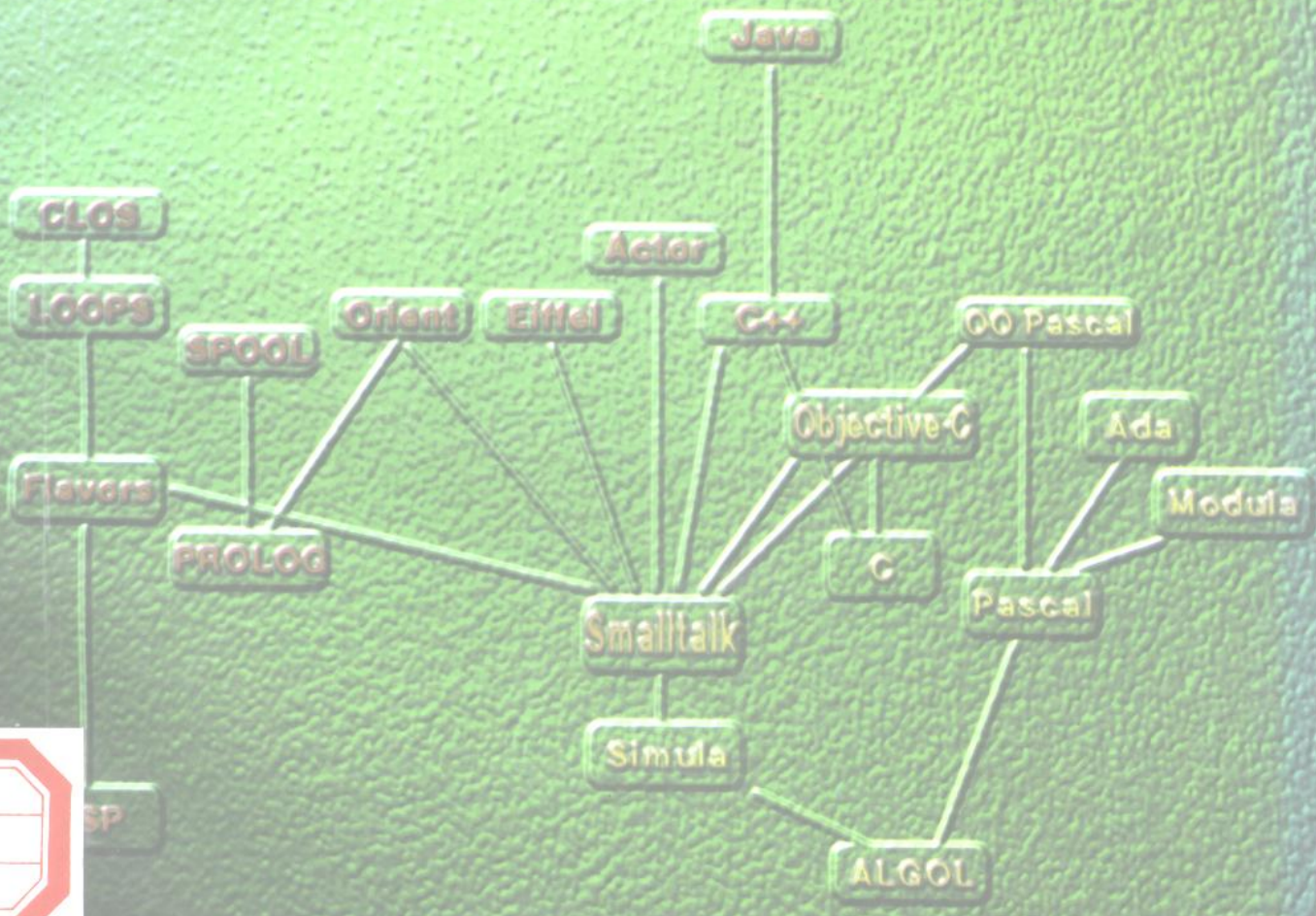
计算机程序设计语言原理

计算机程序设计
语言原理

高集荣　田　艳　编著



西北工业大学出版社

计算机程序设计语言原理

高集荣　田　艳　编著

西北工业大学出版社

1997年5月　西安

（陕）新登字 009 号

【内容简介】 自 50 年代出现 FORTRAN 语言以来，已有数百种计算机高级程序设计语言，最常用的也有几十种。掌握程序设计语言基本概念与基本原理，在短时间里了解计算机程序设计语言全貌及基本规律，学会几种适合自己专业应用的程序设计语言是非常必要的。

本书是参照国家教委推广的“93 计算机教程”中规定的程序设计语言教学大纲编写的。编者从理论与实践相结合的角度深入浅出地介绍高级程序设计语言的基本原理与结构。全书共分为两大部分，第一部分介绍传统程序设计语言共同性的概念与规律，包括程序设计语言综述、语言的定义、程序结构、数据类型、控制结构；第二部分介绍新型程序设计语言，包括函数型、逻辑型及面向对象型程序设计语言。

本书可作为高等院校计算机程序设计语言本、专科教材，也可供计算机爱好者自学或相关科学工作者与工程技术人员参考。

计算机程序设计语言原理

高集荣 田 艳 编著

责任编辑 王 璐

责任校对 齐随印

*

©1997 西北工业大学出版社出版发行

（710072 西安市友谊西路 127 号 电话 8493844）

全国新华书店经销

西北工业大学出版社印刷厂印装

ISBN 7-5612-0962-2/TP·131

*

开本：787×1092 毫米 1/16 印张：12.125 字数：309 千字

1997 年 5 月第 1 版 1997 年 5 月第 1 次印刷

印数：1—4 000 册 定价：16.00 元

前　言

程序设计语言是人与计算机进行通信的工具，它是软件开发最重要的工具。随着计算机科学技术的发展，计算机得到了越来越广泛的应用。计算机科学技术发展的每一步几乎都在程序设计语言中得到体现。可以说，计算机科学技术的发展与高级程序设计语言的发展有着密切的关系，计算机的很多问题就是程序设计语言的问题。因此，学习和研究程序设计语言是极其重要的。

当今程序设计语言已有数百种，最常用的也有几十种之多。目前，国内各大专院校普遍开设了各种高级程序设计语言课程，例如 BASIC，FORTRAN，COBOL，Pascal，C，Ada，LISP，PROLOG，C＋＋等语言。有些院校甚至开设了多门语言课程，但实际效果并不理想。各门语言课程的讲授重点一般都在如何使用语言编写程序上，而不在语言的特点与风格上。因此，许多学生尽管学了多种程序设计语言，却不能掌握好程序设计语言基本概念与基本原理，不能很好地学习、掌握新的语言。为了适应计算机程序设计语言教学改革的需要，我们编写了这本《计算机程序设计语言原理》。

在本书的编写中，参照了国家教委推广的“93 计算机教程”中规定的程序设计语言教学大纲，结合作者在多年教学、科研工作中，对各种计算机程序设计语言的应用所做的分析、归纳，试图从理论与实践相结合的角度深入浅出地向学生讲授高级程序设计语言的基本原理与结构，使学生能掌握高级程序设计语言的基本概念与规律。全书共分为两大部分：第一部分（第一章至第五章）介绍传统程序设计语言共同性的概念与规律（程序设计语言综述、语言的定义、程序结构、数据类型、控制结构等）；第二部分（第六章至第八章）介绍新型程序设计语言（函数型、逻辑型及面向对象型程序设计语言）。

本书可作为计算机程序设计语言教材，也可供相关科学与工程技术人员参考。要求读者掌握一两门高级程序设计语言（如 Pascal，BASIC，FORTRAN），并最好具备一定的数据结构方面的知识。学习高级程序设计语言必然要涉及多种程序设计语言，因此希望读者在阅读本书时最好参阅一些有关语言的标准和手册，这样能够较好地掌握本书的内容。

本书由高集荣、田艳编著。第一章至第五章以及附录由高集荣编写，第六章至第八章由田艳编写，胡飞参加了第五章部分内容的编写。西北工业大学蒋立源教授详细地审阅了全书，并提出了许多宝贵的意见。在全书的编写及出版过程中得到了西北工业大学计算机系胡正国教授、赵政文副教授的大力支持，在此表示衷心感谢。

由于编者水平有限，编写时间仓促，书中难免存在错误和不当之处，恳切希望读者提出宝贵意见。

编　者

1996 年 5 月于西安

目　录

第一章　绪　论 …… 1

§1.1　引言 …… 1
§1.2　程序设计语言的发展史 …… 1
1.2.1　第一代语言:机器语言 …… 1
1.2.2　第二代语言:汇编语言 …… 2
1.2.3　第三代语言:高级语言 …… 2
1.2.4　第四代语言:4GL …… 9
§1.3　高级程序设计语言的特点 …… 12
§1.4　程序设计语言的分类 …… 12
§1.5　程序设计语言的评价标准 …… 12
1.5.1　软件质量的评价标准 …… 13
1.5.2　程序设计语言的评价标准 …… 14
1.5.3　程序设计语言的选择标准 …… 14
§1.6　学习与研究程序设计语言原理的重要性 …… 15

习题一 …… 16

第二章　语言的定义 …… 17

§2.1　字符集 …… 17
§2.2　元语言 …… 18
§2.3　形式语言与文法 …… 19
§2.4　BNF 范式 …… 22
§2.5　语法图 …… 24
§2.6　语义与语用 …… 25

习题二 …… 26

第三章　程序结构 …… 28

§3.1　程序书写格式 …… 28
§3.2　表达式和语句 …… 29
3.2.1　算术表达式 …… 29
3.2.2　关系表达式 …… 30
3.2.3　逻辑表达式 …… 30
3.2.4　字符表达式 …… 31
3.2.5　语句 …… 31

§3.3 子程序、模块及程序 …… 32
3.3.1 子程序 …… 32
3.3.2 模块 …… 33
3.3.3 程序 …… 34
3.3.4 分块编译 …… 34
§3.4 常用语言的程序结构 …… 34
3.4.1 FORTRAN 语言的程序结构 …… 35
3.4.2 Pascal 语言的程序结构 …… 36
3.4.3 C 语言的程序结构 …… 37
§3.5 子程序 …… 38
3.5.1 子程序定义 …… 38
3.5.2 子程序调用 …… 45
3.5.3 参数传递 …… 46
§3.6 变量的作用域 …… 47
3.6.1 局部变量 …… 47
3.6.2 全局变量 …… 48
习题三 …… 48
第四章 数据类型 …… 50
§4.1 数据类型及其划分 …… 50
§4.2 基本数据类型 …… 51
4.2.1 常量与变量 …… 51
4.2.2 整数类型 …… 52
4.2.3 实数类型 …… 53
4.2.4 字符类型 …… 54
4.2.5 逻辑类型 …… 55
4.2.6 枚举类型 …… 55
§4.3 数组类型 …… 56
4.3.1 数组的定义 …… 56
4.3.2 数组的引用 …… 58
§4.4 字符串类型 …… 62
§4.5 集合类型 …… 65
§4.6 记录类型 …… 68
§4.7 文件类型 …… 71
§4.8 指针类型 …… 71
4.8.1 指针的定义和引用 …… 72
4.8.2 指针变量作函数参数 …… 73
4.8.3 指向数组的指针变量 …… 74
4.8.4 指针数组 …… 75

习题四 …… 76
第五章 控制结构 …… 77
§5.1 顺序结构 …… 77
§5.2 选择结构 …… 77
5.2.1 条件结构 …… 78
5.2.2 分情形结构 …… 82
§5.3 循环结构 …… 84
5.3.1 while 型循环语句 …… 85
5.3.2 repeat 型循环语句 …… 86
5.3.3 for 型循环语句 …… 87
5.3.4 循环结构中的其它控制语句 …… 89
§5.4 转移语句 …… 90
习题五 …… 93
第六章 函数型程序设计语言 …… 95
§6.1 引言 …… 95
§6.2 命令型语言的特征 …… 95
§6.3 函数型程序设计的基本概念 …… 97
6.3.1 函数的定义 …… 97
6.3.2 数学函数与语言函数的区别 …… 99
6.3.3 函数型程序设计语言 …… 99
6.3.4 典型的函数型程序设计语言 …… 100
§6.4 LISP 语言 …… 100
6.4.1 LISP 语言的基本语法 …… 100
6.4.2 LISP 语言的内部函数 …… 105
6.4.3 LISP 语言应用举例 …… 110
6.4.4 LISP 语言的特点 …… 111
§6.5 命令型语言与作用型语言的比较 …… 112
习题六 …… 113
第七章 逻辑程序设计语言 …… 114
§7.1 逻辑程序设计的基本概念 …… 114
§7.2 PROLOG 语言 …… 115
7.2.1 PROLOG 语言的基本语法 …… 115
7.2.2 PROLOG 语言应用举例 …… 126
7.2.3 PROLOG 语言的特点 …… 129
7.2.4 PROLOG 语言的适用范围 …… 131
7.2.5 PROLOG 语言程序设计的原则 …… 133

§7.3 LISP 语言和 PROLOG 语言的比较 …… 134
§7.4 逻辑程序设计语言展望 …… 135
习题七 …… 136
第八章 面向对象程序设计语言 …… 137
§8.1 引言 …… 137
§8.2 面向对象程序设计的基本概念 …… 137
8.2.1 面向对象的基本概念与特征 …… 137
8.2.2 面向对象程序设计语言的发展 …… 143
8.2.3 面向对象程序设计语言的优点 …… 145
§8.3 Smalltalk 语言 …… 146
8.3.1 引言 …… 146
8.3.2 基本特征 …… 147
8.3.3 消息和方法 …… 148
8.3.4 继承与多态性 …… 152
§8.4 C++语言 …… 155
8.4.1 引言 …… 155
8.4.2 类 …… 156
8.4.3 重载 …… 161
8.4.4 继承 …… 163
8.4.5 应用举例 …… 165
8.4.6 小结 …… 171
习题八 …… 171
附录 C 语言 BNF 及语法图 …… 173
1. C 语言 BNF …… 173
2. C 语言语法图 …… 177
参考文献 …… 185

第一章 绪 论

§1.1 引 言

语言是人们交流思想、传达信息的工具。中国人使用汉语，英国人、美国人等使用英语，日本人使用日语，等等，这类语言是在人类历史长期发展过程中逐步丰富形成的，我们称之为自然语言(Natural Language)。

除了自然语言外，还有许多专业性语言。例如五线谱是音乐语言，图纸是工程语言，等等。著名的语言学家韦伯斯特(Webster)给语言下的定义是："语言是为相当大的团体的人所懂得并使用的字以及组合这些字的方法的统一体"。这个定义揭示了语言的两个要素：词法和语法。

计算机是人们用来进行各种计算和操作的工具。人们要使用计算机，就必须把要解决的问题告诉计算机，计算机则按照人们的命令进行计算和操作。如何把解决问题的意图告诉计算机呢？也就是如何把解题的信息传达给计算机呢？这就得通过一种特定的"语言"来实现。人们将用这种语言所描述的一系列命令(指令)提交给计算机，告诉计算机所要处理的数据以及如何处理这些数据。提交给计算机的数据和命令(指令)的集合通常称为程序。

为了安全、有效地实现人与计算机之间的通信，人们设计出了各种词汇较少、语法简单、意义明确并适合于计算机的语言。这样的语言被称为计算机语言，又被称为程序设计语言(Programming Language)。

随着计算机科学技术的迅速发展，程序设计语言也不断地由低级向高级发展。程序设计语言简单地可分为低级语言和高级语言。

低级语言是与机器有关的语言，包括机器语言和汇编语言。高级语言是与机器无关的语言。

§1.2 程序设计语言的发展史

自从世界上第一台电子计算机 ENIAC (Electronic Numerical Intergrater And Calculator)于 1946 年问世以来，伴随着计算机硬件的不断更新换代，计算机程序设计语言也有了很大的发展，至今已有四代语言问世。

1.2.1 第一代语言：机器语言

计算机只能接收由"0"和"1" 组成的二进制信息。要计算机执行一定的操作，就要编写一系列规定的二进制代码。这种可由计算机识别并执行的代码叫做机器指令。这些机器指令的集合叫做机器语言。机器语言程序就是由一条条机器指令组成的程序。

用机器语言编写程序是一件非常困难、繁琐的工作。机器语言全是"0"和"1" 组成的符号，用其编写出的程序直观性差，难以阅读。要记住每一条指令的格式及其功能是十分困难的，所

以机器语言难学、难记，程序难以编写、检查、调试和维护。另外，不同的机器具有不同的指令系统，用一种机器语言编写的程序难以移植到另一种机器上去。因此，机器语言只适合计算机专业人员使用，而让用户掌握几乎是不可能的。这在很大程度上限制了计算机的推广使用。

1.2.2 第二代语言：汇编语言

在40年代后期和50年代初，为了减轻用机器语言编写程序的繁琐和辛苦，人们设计出了更接近于自然语言和数学语言的第二代程序设计语言——汇编语言。汇编语言是一种面向机器的低级程序设计语言。汇编语言是一种符号语言，它用符号来表示机器指令。例如，用"ADD"、"SUB"和"MOVE"分别代替机器语言中二进制代码的加法、减法和代码移动。汇编语言把指令、地址及数据都加以符号化，因此它比机器语言容易懂、容易记、容易修改。用汇编语言编写的程序必须经汇编程序加工成计算机能够接受的目标程序(即机器语言程序)，计算机才能执行。另外，作为一种面向机器的语言，汇编语言是为特定的计算机所设计的，因而不同的计算机所配的汇编语言也各不相同，它们不能够通用。但是，正是由于它依赖于机器，所以就与机器语言程序一样可以结合机器的特点编写出高质量的程序，即程序代码短，执行速度快。所以时至今日，汇编语言在程序设计中仍然起着重要的作用。编制那些使用频度高或要求处理时间短的程序，例如实时测控系统这类软件，仍用汇编语言来编写。

然而，由于汇编语言与具体的计算机结构有关，用户在编制程序之前仍需花相当多的精力去熟悉机器的结构。用汇编语言编写程序与用机器语言编写程序仍有相似之处，即繁琐、工作量大，而且编出的程序无通用性。

一般来说，作为一个高级程序设计人员，汇编语言是必须要掌握的一门语言。

1.2.3 第三代语言：高级语言

第一代语言(机器语言)和第二代语言(汇编语言)都是针对某些特定机器的语言，因此称之为面向机器的语言，通常也把它们称为低级语言。自然，人们希望有独立于机器的语言，希望这种语言能接近人们日常使用的自然语言，接近人们的习惯。为了区别于前两种低级语言，人们把这种语言称为高级语言。高级语言是一种类似于自然语言和数学公式的程序设计语言。它从根本上克服了低级语言的缺点，使程序设计语言适合于描述各种算法，而不依赖于具体的计算机，从而便于计算机应用的推广。于是50年代中期开始以FORTRAN语言为代表的各种计算机高级语言就应运而生了。

由于用高级语言编程比较接近人们熟悉的数学公式，便于学习、使用和交流，比用低级语言更容易编写、阅读、查错和修改；只需要选择正确的算法和适当的数据结构，就可以用这种语言编写解题程序，特别是为非计算机专业人员使用计算机创造了有利条件。同时，一个用高级语言编写的源程序在不同型号的计算机上都可以使用，这说明高级语言程序的通用性好。当然，有的源程序在使用时可能要做些小的修改，这取决于语言的编译程序以及计算机的外部设备。高级语言的语句功能很强，一条语句往往相当于许多条机器指令，这就使编程效率大为提高。

目前世界上已经有了1000多种高级语言，在计算机上实现的有数百种，其中最常用的有几十种，例如，BASIC、FORTRAN、COBOL、Pascal、C、APL、PL/I、Ada、LISP、FORTH、Modula-2、LOGO、PRG、PROLOG、Smalltalk、C++等。

1. 早期的高级语言

第一个高级程序设计语言诞生于50年代。当时的计算机非常昂贵，而且数量非常之少，如何有效地使用计算机是一个相当重要的问题。另一方面，计算机的执行效率也是人们追求的目标。为了有效地使用计算机，人们设计出了高级语言，用以满足用户的需求。用高级语言编写的程序需要经过翻译，计算机才能执行。虽然，程序翻译占去了一些计算机时间，在一定程度上影响了计算机的使用效率。但是实践证明，高级语言是有效地使用计算机与计算机执行效率之间的一个很好的折中手段。

这一历史时期最重要而又最具代表性的语言是FORTRAN、ALGOL 60及COBOL。

FORTRAN 是英文FORmula TRANslation的缩写，意思为"公式翻译"。FORTRAN语言是第一个高级程序设计语言。它于1954年首次被提出，1956年开始使用，以后研制出了许多不同的版本。但以FORTRAN Ⅰ(1958年)和FORTRAN Ⅲ(1962年)最为流行。FORTRAN语言一公布，便受到了极大的欢迎并很快流行起来。

ALGOL 是英文ALGOrithmic Language的缩写，意为"算法语言"。ALGOL语言是50年代后期开始研制的，1958年诞生了ALGOL 58。两年后，推出了与ALGOL 58风格完全不同的ALGOL 60。虽然FORTRAN语言的应用范围远比ALGOL语言大得多，但ALGOL语言对现代程序设计语言的发展所起的作用是FORTRAN语言所无法比拟的。ALGOL 60因其具有准确而完备的文本以及采用了形式化的语法描述体系——BNF范式(巴科斯范式)，而倍受计算机语言研究者和设计者的青睐。但由于ALGOL 60没有标准的输入/输出，同时它采用了按名调用的参数传递机制，从而使ALGOL 60得不到广泛的使用，得不到像IBM这样的大公司的支持，也是ALGOL 60不能很好推广的原因之一。尽管如此，ALGOL 60语言的许多设计思想直到现在都还影响着程序设计语言的研究与设计，例如，分程序概念、递归过程、按名调用、动态数组等。

COBOL 是英文COmmon Business Oriented Language的缩写，意为"面向商业的公用语言"。它于1959年5月开始设计，于同年12月正式推出了第一个版本。COBOL语言引入了文件和数据描述概念(进而产生了数据库管理系统)及类自然语言(英语)语法结构。COBOL语言被广泛地用于数据处理、事物处理、情报检索等。COBOL是世界上用得最多的程序设计语言。在美国、日本等国家，绝大多数与事务处理有关的程序都是用COBOL语言编写的。从大型机、中型机到小型机、微型机，大多数都配置了COBOL语言编译程序。COBOL语言之所以如此普及，其重要的一个原因是，使用COBOL语言编写程序与用英语描述很类似，程序编制人员，尤其是那些会英语的程序编制人员，感到很亲切，容易掌握。

在FORTRAN、ALGOL 60及COBOL中，FORTRAN与ALGOL 60被认为是科学计算(数值计算)语言，COBOL被看作是数据处理语言。除ALGOL 60外，FORTRAN与COBOL语言至今仍是使用最广泛的语言，这也证明了它们的有效性。当然，它们被如此长期、成功地使用，还有其它一些原因。例如，许多用户不愿意换用新的语言，以便能继续使用已有的程序。但更重要的是，这些语言本身也在不断地改进、完善和发展。例如FORTRAN语言就始终在不断地发展。它从最初设计的FORTRAN 0，发展到FORTRAN Ⅰ、FORTRAN Ⅱ、FORTRAN Ⅲ、FORTRAN Ⅳ以及FORTRAN 77，1995年又推出了最新版本FORTRAN 90。FORTRAN从诞生起一直是科学计算领域中最重要的程序设计语言。

2. 60年代语言的发展

60年代是程序设计语言发展的一个重要时期，在这期间，人们研制出了大约200多个高级程序设计语言。其中比较著名的有LISP、APL、SNOBOL、PL/I、SIMULA、BASIC，以及Pascal等。

LISP语言是50年代末60年代初，主要由麻省理工学院的John. McCarthy等人研制成功的。LISP是英文LISt Processing的缩写，意为"表处理"。LISP的应用范围主要是人工智能，这个研究领域中的问题促使John. McCarthy等人设计了这种概念上非常简单、适合于处理符号表达式的语言。他们给程序设计语言引入了许多新的概念。提出了一种数据和程序的统一表达式，叫做S-表达式(Symbolic expression)。此外，还引入了新的条件表达式、运算符的前缀形式以及基本控制结构的递归等概念。LISP语言定义是基于函数和函数作用的数学概念，它奠定了函数型(或作用型)这样一个新型语言类的基础。

APL语言是1961年研制成功的一个支持函数型程序设计的语言。APL是英文Automatically Programming Language的缩写，意为"自动编程语言"。APL是最早的，也是应用最广泛的数控语言(Numerical Control Language)。它能用来描述数控机床加工零件的几何形状和加工过程。它是一种功能强而表示紧凑的语言。它具有丰富的操作符，特别是对矩阵运算的操作符，使程序员从使用低级重复的、逐个矩阵元素的处理中解脱出来。

SNOBOL语言是于1966年首次被提出的。SNOBOL是英文StriNg Oriented symBOlic Language的缩写，意为"面向字符串的符号语言"。SNOBOL语言问世后不久，又有了改进版本，其中1968年推出的SNOBOL-4语言使用较广。它主要用于需要以复杂的方式处理大量字符串的领域。此外，SNOBOL-4也可用于谓词演算和微分方程等方面。SNOBOL语言的复杂字符串处理功能是其后任何语言都无法比拟的。

PL/I语言是60年代中期设计的。PL/I是英文Programming Language/one的缩写，它试图取代当时流行的FORTRAN与COBOL、LISP等语言，成为一个大型、全面、汇集了各种各样功能的、真正通用的、普遍采用的语言。PL/I语言从ALGOL 60、FORTRAN、COBOL与LISP中吸取了分程序结构、递归过程、数据描述、动态数据结构等概念。此外，它还引入异常处理及多任务处理的机制。由于PL/I有包罗万象的特性，所以它也被称为"公共汽车"语言。

SIMULA语言是60年代中期研制的。SIMULA是英文SIMULAtion的缩写，意为"模拟"。它是一个典型的离散系统模拟语言，主要用在模拟领域。它是ALGOL 60语言的扩充，它具有表加工等功能。SIMULA为程序设计语言引入了类(class)的概念，即把有关的子程序和数据组合成一个单元(unit)，用以增强说明的层次性。类也是现代面向对象程序设计语言的一个主要特征。

BASIC语言是60年代中期设计而至今仍被广泛使用的一种程序设计语言。BASIC是英文Beginner's All-purpose Symbolic Instruction Code的缩写，意为"初学者通用符号指令代码"。它具有一个类似于FORTRAN语言的简单代数语法、有限的控制结构和数据结构。BASIC语言以其小巧、灵活、浅显易懂、较强的会话功能等特点，赢得了广大的用户。BASIC语言本身并未引入任何语言上的新概念，但它是第一个支持高度交互及解释性程序设计语言。

BASIC是普及面最广、最简单易学的语言。此外，它也是发展最快的语言。首先BASIC具有众多的版本，从APPLE机的MS BASIC到PC机的PC BASIC、BASICA、GWBASIC，到80年代后期出现的True BASIC、Turbo BASIC和Quick BASIC，以及90年代出现的GFA-BASIC、Visual BASIC和获得1993年BYTE奖的Visual BASIC for DOS，版本数不胜数。其

次是质的变化。早期的BASIC语言是解释型的非结构化语言。1985年推出的新版本True BASIC给BASIC配置了编译器，使其成为"两栖语言"，即可解释执行，又可编译执行。另外，True BASIC已成为严格的结构化程序设计语言，彻底摒弃了GOTO语句，提出了库的概念，等等，使BASIC语言有了飞跃的发展。有不少专家预计，由于BASIC语言的以上特点，它将继续作为最受欢迎的程序设计语言而不断发展更新。

Pascal语言是这些语言中最成功的语言。它是瑞士苏黎世联邦工业大学的N. Wirth教授于60年代末设计完成的。N. Wirth教授研制Pascal语言的目的是设计出一种适用于教学及编写系统软件的简便的语言。N. Wirth命名这个语言是为了纪念17世纪法国科学家Blasise Pascal。Pascal语言最早实现于1971年，1973年对它进行了修改并重新实现。Pascal是以ALGOL 60为基础而设计的。它不仅继承了ALGOL 60的定义与描述方法、分程序等，而且描述更加精确、合理与简洁。Pascal语言对后来的语言设计产生了很大的影响。像Modula、Modula-2、Ada及并发Pascal等语言都是在Pascal的基础上发展起来的。

Pascal语言是一种结构化的程序设计语言(Structure Language)。由于它具有很强的过程结构，丰富的数据类型和数据结构，较强的语言表达能力以及良好的移植性，不仅被国内外许多高等院校采用为教学语言，而且被广泛地用于科学计算、数据处理，以及系统软件开发中。目前，Pascal语言已在从微型到大型的各类计算机上实现。它也是一种系统程序设计语言，既可以用它编写应用程序，也可以用它来编写像编译程序一类的系统软件。目前该语言已被广泛地应用于教学与各种领域的软件研制中。

3. 70年代语言的发展

到了70年代，由于软件工程的兴起，人们对计算机软件可靠性和可维护性的要求越来越高。而大型系统软件及应用软件的生产要耗费大量的人力、物力。因此，支持系统程序设计的语言也应运而生，从而推动了对程序设计语言的研究。60年代的程序设计语言是全方位发展的，既有命令性(imperative)语言(如ALGOL 60、PL/I、BASIC、Pascal等)，也有作用性(applicative)语言(如LISP等)；70年代则主要集中于命令性语言的研制及其相应理论的研究。70年代问世的主要程序设计语言有并发Pascal、C、Modula、Ada以及面向对象型语言Smalltalk等。

C语言是70年代初推出的一种系统程序设计语言。它是以与Pascal等语言相类似的思想为基础而设计出来的面向结构的程序设计语言。C语言既有高级语言的特点，又比较靠近硬件系统，具有像汇编语言那样可以直接访问硬件的功能，因此，它也被称为"高级汇编语言"。C语言可以处理计算机直接操作的大多数数据，适用于编写操作系统、编译程序及软件工具等。C语言程序的函数结构非常有利于把整个程序分割成若干相对独立的功能模块，并为程序模块间的相互调用及数据传递提供了便利。这一特点为把大型软件模块化，提供了强有力的支持。近年来使用C语言的趋势日益增强，当前几乎各种型号的微型、小型及大中型计算机上都配置有C语言编译系统。目前，微机上已配备了多种版本的C语言编译系统，例如，C86、Lattice C、Wils C、MCS C、Turbo C、Quick C及Visual C等语言。目前流行的是MCS C、Turbo C、Quick C及Visual C语言。C语言现已发展成为一种广泛使用的通用程序设计语言。C语言的特点是语言简洁、紧凑，使用方便灵活，具有丰富的运算符和数据结构，具有很强的结构化控制语句，生成的代码质量高，程序执行效率高，通用性好等。

Modula语言是一个模块化程序设计语言(Modula Language)，可作为系统程序设计语

言。Modula 语言是 Pascal 语言的创始人 N. Wirth 教授于 1979 年设计的。1980 年 N. Wirth 教授将其修改扩充成 Modula－2 语言。Modula－2 语言是 N. Wirth 教授为了克服 Pascal 语言的不足而开发的，它是 Pascal 语言的扩充和发展。Modula－2 语言与其它语言的最显著的区别是它的模块特性，即它的程序是由模块组成的。最外层的模块相当于其它语言的主程序，也可以在最外层模块之内定义其它模块。模块通常由一组有关的过程组成。模块特性减少了各程序不同部分之间的相互干扰。此外，Modula－2 语言可以把模块分别放在不同的文件中，并分别进行编译。这对于大型程序的开发及调试来说，无疑是非常重要和有益的。因为，只要一个个模块编译通过后，无论主程序重新编译多少次，它们都不必再编译了。而对其它语言来说，只要改变程序哪怕是很少一部分，整个程序就得全部重新编译。

除了具有模块特性和可以分别编译的优点外，所有在 Pascal 语言中具有的优点，Modula－2 语言也都具备。由于 80 年代才正式推出其商品化编译程序，所以 Modula－2 语言是一种较新的程序设计语言。虽然目前使用 Modula－2 语言的用户尚不普遍，但预计今后它将会逐渐流行起来。

Modula－2 语言作为 Pascal 语言的一种后续语言，使得熟悉 Pascal 语言的程序员只需花费几个小时便可学会 Modula－2 语言的使用。应该说，Modula－2 语言能鼓励好的结构化编程风格，它是通用编程，尤其是开发大型程序的一种非常理想的语言。

Ada 语言在某种意义上恐怕是上述语言中最引人注目的一个。它是迄今为止最完善的面向过程的高级程序设计语言。70 年代，世界上用于各方面的程序设计语言已有数百种之多，它们在开发过程中缺乏统一的标准，给不同语言之间的交流带来很大困难。基于这种情况，美国国防部为了适应陆、海、空三军对程序设计语言日益增长的要求，于 1975 年成立了一个高级语言工作组(HOLWG)，决定设计一种新的适合于嵌入式(embeded)计算机系统的高级标准语言。嵌入式计算机软件具有可靠性要求高、规模大、复杂性高、开发费用高、程序运行时间长等特点。

美国国防部之所以把这种语言取名为 Ada，是为了纪念英格兰诗人拜伦(Byron)勋爵的女儿 Ada，她曾整理和修正被称为现代计算机之父的查尔斯.巴贝奇(Charles Babage)的笔记和手稿，并为巴贝奇 1922 年设计的差分机(Difference Engin) 编制计算机程序，被誉为世界上第一位计算机程序员。

Ada 语言的设计目标是试图建立一个统一的标准语言。但就在 Ada 语言发表之后不久，也就是在美国国防部管辖范围内又出现了许多新的语言，如美国空军设计出了其通用的程序设计语言 JOVIAL。事实上，要统一成一种语言并不符合人们的“自由”心理。虽然 Ada 语言并没有完成语言的统一工作，但它综合了其它语言的长处，可用于大型实时军事系统，也适用于民用和工业应用。因此，Ada 语言被认为是传统的程序设计语言发展的顶峰。

Ada 语言是以 Pascal 语言为基础开发的，有些概念是 Pascal 语言的延伸，有些概念则摆脱了 Pascal 语言的局限性。但 Ada 语言并不是 Pascal 语言的简单扩充，许多新的、重要的概念是 Ada 语言所特有的。Ada 语言增加、扩充了数值计算、输入/输出、数据抽象、并发处理、实时处理、异常处理，以及与机器有关的设施。其中要强调的是，Ada 语言引入了一个非常有用的工具——程序包。程序包是一些资源的集合，它不仅提供了一个算法运算工具，而且还提供了与之相联系的数据、类型及其结构。Ada 语言强调程序的可靠性、效率及可维护性。它集中了 70 年代软件工程、程序设计语言学、程序设计方法学等学科的最新研究成果，能很好地支持程

序模块性、可移植性、可扩展性、抽象与信息隐藏等，并有助于高效、高质量地开发与维护程序。

近10年来，Ada语言得到了广泛的应用，取得了令人注目的成果。在1991年的海湾战争中，有好几个大型军用软件，如指挥通信系统、预警系统等都是用Ada语言编写的，实际使用效果令人满意，显示出了Ada语言的优越性。在1992年11月的TRI-Ada'92大会上，IBM公司介绍了用Ada语言编写的全美空中交通控制软件系统，该系统实际使用结果非常成功。美国国会已通过法令，规定从1991年6月1日起，所有的美国军用软件全部使用Ada语言编写。英、法、德等国军方也规定使用Ada语言。Ada语言在我国同样广泛地受到重视，也被确定为我军的标准程序设计语言。除了用于军方外，Ada语言也越来越多地用工业、商业、科学研究和大学教学等领域。目前，Ada语言在美国及欧洲得到了广泛的应用。

鉴于各国对Ada语言的需求不断高涨，美国宣布对目前流行的Ada 83进行修订，新的标准定为Ada 9X。1995年2月，ISO接受Ada 1995修订版并将其称为Ada 95。Ada 95包括了面向对象编程的特征，并支持实时系统。

4. 80年代语言发展趋势

传统的程序设计语言受到冯·诺依曼概念的制约，使其存在许多局限性。进入80年代后，摆脱冯·诺依曼概念的束缚已成为众多计算机程序设计语言研究者与设计者为之奋斗的目标。为此目标而研制的语言被称为新型程序设计语言(New Style Programming Languag)，也称为知识型程序设计语言。新型程序设计语言力求摆脱传统语言那种状态转换语义的模式，以适应现代计算机系统知识化、智能化的发展趋势。新型程序设计语言基本上可以分为三类，即函数型语言、逻辑型语言和面向对象型语言。

专家预言，下个世纪将是以知识信息处理为中心的信息化社会，计算机系统(硬件、软件、通信网络)除了继续在确定型数据处理领域发挥巨大作用以外，将为适应知识信息处理发生根本变革。

现行的各种程序设计语言，其知识信息处理的能力还比较低，而新型程序设计语言可以说是新的"软件危机"中的一线曙光。函数型语言、逻辑型语言和面向对象型语言代表了新一代程序设计语言的方向。

(1) 函数型语言　函数型语言以λ-演算和递归函数作为计算模型。其程序的基本构成单位是函数。函数型语言的程序设计就是构造函数，即从系统提供的基本函数出发，用复合和递归等方法构造出新的函数。

函数型语言也被称为作用性(applicative)语言。函数型语言程序具有"引用透明性"(referential transparency)。引用透明性是指函数的计算仅与其子函数的计算值有关，而与可能的计算顺序无关。

函数型语言适用于自上而下、逐步求精的程序设计方法。它具有数据结构化的能力和定义高阶函数的能力。

函数型语言的代表性语言有LISP、FP、FFP和APL等。

FP语言是由图灵奖获得者John Backus提出来的一种纯函数型语言。FP语言的基本概念建立在原子(atom)的基础上，原子用于建立对象与表达式。而FFP语言是在FP语言的基础上提出来的另一种函数型语言。FFP语言还引入了一些新的概念，例如作用(application)、元合成(metacomposition)、存(store)、取(fetch)等。FFP语言与LISP、APL语言有许多一致的地方，因此受到了人们的普遍重视。

(2) 逻辑型语言　逻辑型语言以一阶谓词逻辑作为计算模型，它是一种描述型语言。

PROLOG(PROgramming in LOGic)语言是由法国马赛大学于 1972 年开发出的一种逻辑型语言。它以一阶谓词逻辑及关系理论为基础，使用户以接近自然语言的逻辑描述形式编写程序。PROLOG 具有下列几个特征：

① 以面向人类而不是以面向机器的语言(谓词)为基础；

② 具有现有程序设计语言所不具备的高级功能(如知识库、模式匹配、回溯等)；

③ 程序使用比较方便。

目前，PROLOG 语言已被广泛地应用于关系数据库、抽象问题处理、数理逻辑、公式处理、自然语言理解、定理证明、专家系统、编制棋牌类程序，以及人工智能的许多领域。日本政府于 80 年代初宣布以 PROLOG 作为第五代计算机系统的核心语言后，更加引起了人们对该语言的重视。

除了基本顺序的 PROLOG 语言外，80 年代以来又出现了一些并行的 PROLOG 语言。例如，1980 年 K. L. Clark 发表了 IC - PROLOG。1981 年他又推出了 Relational Language。1983 年 E. Shapiro 提出了 Concurrent PROLOG。1984 年 L. M. Pereira 在 Concurrent PROLOG 的基础上提出了 Delra - PROLOG。同年，K. L. Clark 也提出了 PARLOG 语言。

PROLOG 目前已有多种版本，例如 DEC PROLOG、PDP - 11 PROLOG、SYSTEM - 10 PROLOG、UNIX PROLOG、Micro PROLOG、Turbo PROLOG，以及 PDC PROLOG 等都是目前比较成熟的 PROLOG 系统。在这些系统中，Turbo PROLOG 除了它的核心部分外，还具有中英文兼用、多窗口、图形、声音、DOS 交互调用以及动态数据库等先进功能，所以它在我国应用比较广泛。

(3) 面向对象型语言　面向对象(Object Oriented)型语言的程序设计思想起源于 Simula 67 语言。该语言的基本思想是提供图像处理功能以建立友好的人机界面，使未经专业训练的人能方便地使用计算机。它对问题领域实行自然分割，按照人们一般思维方式建立问题的模型及事物间的联系。

Smalltalk 是 70 年代提出的一个面向对象型语言。经过多年连续不断的研究、改进之后，在 1980 年推出了商品化的面向对象的程序设计语言 Smalltalk - 80。进入 80 年代，面向对象型语言的研究发展非常迅速，大部分的研究都是以 Smalltalk 为基础。研究出的语言可以分为两类：

① 传统语言结合 Smalltalk 的面向对象思想模仿 Smalltalk。例如：

美国 Apple 公司于 1980 年开发出了 Object Pascal 语言。

PPI 公司于 1980 年开发出了 Objective - C 语言。

Kriya System 公司于 1984 年开发出了 Neon 语言，它是一个面向对象的 FORTH 语言。

Coral Software 公司于 1986 年推出了 Object LOGO 语言。

② 以传统语言为基础扩充面向对象思想。例如：

在 AT&T 公司贝尔实验室工作的 Bjarne Stroustrup，从 1980 年至 1983 年，在 C 语言的基础上，增加和扩充了面向对象的思想，开发出了 C with classes(带类的 C)。当时还没有操作符重载、引用(reference)和虚函数(virtual function)等许多的细节。

AT&T 公司贝尔实验室于 1985 年推出了 C++语言，它以 C 语言为基础扩充了类的概念，并具有继承性。1988 年又推出了 C++语言的新版本，使类具有多重继承性。C++语言虽

然问世只有几年的时间，但其影响已很大。目前新的 UNIX 操作系统几乎都提供了 C++语言，IBM PC 及其兼容机上也已配有多种 C++语言，例如，Microsoft C++、Turbo C++、Borland C++、Microsoft Visual C++、Zortech C++、Symantec C++、MetaWare High C++，以及 Watcom C++语言等。另外，AT&T 公司贝尔实验室的 UNIX 操作系统的新版本及其它系统软件也在用 C++语言编写。

Xerox 公司开发的 Loops 语言，是在 Intel LISP 语言基础上扩充了面向对象的概念，于 1983 年推出的美国面向对象的 LISP 西海岸版本。

符号处理公司于 1985 年开发出了 Flavos 语言，它具有多重继承性的类。它也被称为美国面向对象的 LISP 东海岸版本。

Xerox 公司于 1985 年开发出了 Common Loops 语言，它是面向对象的 LISP 版本。

IBM 公司日本分部于 1985 年推出了 SPOOL 语言，它是以面向对象的思想扩充的 PROLOG 语言。

Keio 大学于 1984 年开发出了 Orient 84K，它是基于 PROLOG 语言和 Smalltalk 语言，并且能够并行执行的语言。

联邦德国 Kaiserslauter 大学于 1986 年设计出了 Mod Pascal 语言，它是以面向对象的思想扩充的 Pascal 语言。

DEC 公司于 1986 年开发出了 Trellis/Owl 语言，它具有多重继承性、强类型、静态联编等功能，其基本语言为 Pascal。

面向对象的方法和技术是目前软件研究和应用开发中最活跃的一个领域，被认为是 90 年代的核心技术之一。专家们预计，面向对象的程序设计思想将会主导今后程序设计语言的发展。

1.2.4 第四代语言：4GL

人们从使用机器语言到使用汇编语言，到高级语言（例如，FORTRAN、COBOL、Pascal、C 语言等）来指挥计算机工作。使用这类语言指挥计算机工作，必须严格按照所用语言的语法编制计算机程序，程序的长短与要求计算机解决的问题的复杂度密切相关。为了使用计算机完成复杂的工作。需要编制出非常庞大的计算机程序，例如，美国 70 年代的反导弹系统 Safe-Guard 包含的程序长达 200 万条指令；70 年代末“穿梭号”宇宙飞船的控制软件有 4000 万行机器指令代码。由于传统的程序设计语言的语法繁琐、复杂，编写程序的技巧难于掌握，因此，只有受过专门训练的计算机专业人员才能熟练、正确地使用传统的程序设计语言编写计算机程序。特别是为了能编写规模庞大的程序，不仅需要受过专门的程序设计训练，有使用所选用的程序设计语言的丰富经验，而且更重要的是应该受过严格的软件工程训练，有从事大型软件开发的经验。

因此，人类迫切需要一种能够更容易地向计算机发布命令指挥计算机工作的通讯工具，也就是说，需要新一代的计算机程序设计语言（即第四代语言），以便克服人类应用计算机的过程中所遇到的一系列问题。第四代语言的基本目标是提高计算机应用系统的质量和开发生产率。

那么，什么是第四代计算机程序设计语言 4GL（4 Generation Language）？目前还没有一个公认的严格定义。一种意见认为，3GL 是过程化的语言，其目的在于高效地实现各种算法；