第1章

引　论

任何课程的学习，都应从最基本的知识出发，由简单到复杂，循序渐进地进行。学习高级语言程序设计也不例外。本章主要介绍 Pascal 语言及其程序设计的一些背景知识，语言的字符集、符号、约定与解释，语言的特点，Turbo Pascal 语言源程序的结构及其特点。

▷▷ 1.1　计算模型、高级语言与程序设计

从绪论中已知，无论是用计算机系统完成一项计算，还是理解一项计算，当问题本身比较复杂时，应该借助计算模型与计算机系统，计算方法、算法、计算机语言、程序来弄清整个计算，因为计算有多个层次和多个侧面。人们讨论计算，离不开计算方法。用计算机求解某个或某一类问题，有了计算方法，还需要考虑在什么样的计算机系统上来进行计算。而且，即使选定了计算机系统，也还需要考虑使用什么样的语言来写程序，最后完成计算任务。由于人们对问题的认识不同，从不同的角度看问题，在分析解决问题的过程中先后发展了多种理论和形形色色的解决问题的方法。于是，从解决问题方法的角度观察和认识问题，又可以把各种问题划分为不同的种类。这样，不同问题种类在实际进行计算时，有可能需要不同的计算模型、算法、计算机系统、计算机语言、程序设计技术和软件开发方法。例如，一些面向过程的高性能科学计算问题，如果计算涉及的数据量和数值很大，精度要求很高，就需要考虑使用超级计算机系统，用 Fortran 语言来编写程序。又如，针对一个人工智能问题，如果这个问题能够经过抽象用一阶逻辑系统中的公式来表达，那么，这个问题的计算方法很可能是一种推理方法，求解程序可能需要一种逻辑推理系统，用逻辑程序设计语言来写程序。

围绕高级程序设计语言，这里所说的计算模型还是狭义的，主要是指能够识别语言的计算模型。现实的计算机系统是依靠 CPU，根据程序来控制和执行它的指令完成计算任务的。指令集是计算机系统与外界进行交流的语言。这种语言属于低级语言，与人们日常使用的自然语言之间存在一定的差别。为了实现人们事先设计好的计算，软件工作者在计算机裸机系统和高级语言程序之间建立了一组软件系统(系统软件)，使得人们可以容易地使用计算机系统。很多人会由此自然联想：是否只要不断地发展人类的各种语言或程序设计语言，进一步发展系统软件技术，就可以不断地、无限地扩展计算机系统的功能呢？

答案是否定的！因为这涉及到计算机的本质属性和能力。什么是可以计算的，什么是不可以计算的？这个问题在 20 世纪 30 年代自“丘奇-图灵”论题提出后就已经基本解决。今天，绝

大多数计算机科学家都相信，图灵机是一种充分简单，功能十分强大的机器。所谓可计算等价于图灵机可计算。换言之，无论你发展什么语言，要想让这种语言能够为今天的通用计算机系统自动处理，那么，这种语言的能力已经被图灵机所限定。实际上，20 世纪 50 年代和 60 年代在计算机科学理论的研究工作中已经指出：不同复杂程度的语言文法分别与某种计算模型在处理(或识别)语言的能力上相等价。计算机的指令系统具有递归性预示着程序设计语言本身具有递归表达能力的属性。有了这些认知之后，就应该能够认识到语言本身也是一种计算模型，它在计算机裸机系统与人的思维、描述方式之间架起了一座沟通的桥梁，只是在描述、刻画、表达计算时所处层面和方便程度不同而已。这就不难帮助读者理解语言是计算模型的外在表现形式。程序设计语言的数据机制刻画了计算模型(抽象计算机)的抽象存储系统，程序设计语言的控制机制决定了语言的类属，而且可以折射出计算机系统的体系结构。所以说，利用程序设计语言编写程序实质上是对这个抽象计算机的扩充。

用高级语言进行程序设计，离不开程序设计语言。程序设计与程序设计语言是一枚硬币的两面，两者之间存在着密切的联系，相融而不可偏废。程序设计语言是用算法和程序的方法描述与表达计算的工具。人们遇到的各种计算问题的类型是五花八门的，各种问题计算的思想方法和算法也不同，为此，科学家和工程师先后发展了诸多程序设计思想、原理、方法和技术。进一步，对各种程序设计思想、原理、方法、技术在实践中进行科学的总结，通过新型程序设计语言更清晰、简洁、有效地表达出来，融入语言的设计之中，能够更好地促进程序设计的发展，解决大量实际问题。于是，在学科的发展历程中，先后产生了许多不同风格的高级程序设计语言，程序设计语言的科学地位也不断得以巩固。

在程序设计语言大发展的同时，由于计算机应用的广泛开展，也促进了程序设计方法和技术的进步。与早期的小规模程序设计不同，随着程序的规模不断增大，程序的复杂程度也在提高。如何开发和设计大型程序和软件系统，使之不仅满足计算要求，而且有利于今后的维护、扩展、移植和交流，就需要有一整套的程序设计方法和技术，同时还要有办法保证程序的正确性。程序设计方法和程序设计技术属于程序设计方法学的范畴，读者可以参考相关的文献。这里简单介绍有关保证计算正确性的问题。

有经验的程序员都知道，一个程序，如果充分简单，不仅编制程序比较容易，调试程序和保证其正确也是容易做到的。但是，如果一个程序比较复杂或非常庞大，要保证程序的正确性就很困难。一般地说，人们不能穷尽所有的可能通过测试程序来保证程序的正确性。著名学者 E. W. Dijkstra 曾经指出：“测试只能发现程序有错，但不能证明程序正确。”要保证程序正确，还必须回到程序的正确性证明这条路上来。

针对一个问题的计算，如果计算方法正确，算法正确，程序准确地刻画了算法，那么，这个程序一定是正确的。计算方法的正确性由数学理论来保证，算法的正确性由算法设计与分析中涉及的算法理论来保证，而问题常常是出在程序设计方面。一个程序，是否准确地刻画了算法？如何来保证？不难想象，一个有一定复杂程度的程序，要一眼看出其中可能存在的问题不是一件容易的事情。显然，要弄清楚一个程序是否正确，关键是要弄清楚程序中每一条语句的语义和相互之间的联系。而要表达清楚程序的语义，需要有更严格的语义描述方法。由于高级语言的每一条语句与机器指令并不是完全一一对应的，一条语句往往要用一组指令或一段程序来表示，因此，要表达清楚这条语句的含义，可以用一种能够更严格、更准确、其语义本身是清楚的(数学)语言(也称元语言或元方法)来描述和表达这段程序。于是，解决问题，又需要回到更基本的计算模型层面。历史上，先后发展了多种元语言或元方法来描述语句的语义，其中，有著名的抽象机

方法(操作语义)、逻辑方法(公理化语义)、函数方法(指称语义)、逻辑与代数方法(代数语义)。由于这些描述方法和工具位于整个学科比较基础的层面，对高级程序设计语言和程序语义的研究着眼于计算正确性问题的解决，又是对计算问题本质属性的研究，所以，它们也都可以划归为计算模型的研究范畴。

▷▷ 1.2 程序设计语言 Pascal 简介

迄今，人类已经发明了上千种程序设计语言，但只有很小一部分得到了广泛应用。其中，Pascal 语言就是广泛应用的一种。下面介绍这种语言的发展历史、特点、组成及其程序结构。

1.2.1 Pascal 语言的发展及其启示

第一个高级程序设计语言 Fortran 诞生于 1954 年。五年之后，1959 年出现了 Algol 语言，在做了一些修正后，当年年底定名为 Algol 60。与 Fortran 语言相比，Algol 60 语言最初并没有得到许多计算机制造商的使用和支持，但由于 Algol 60 语言的系统化结构及文法的形式化描述深受学术界的青睐与好评。于是，Algol 60 语言被学术界作为科学研究的对象，并着手对它进行扩充、完善和实现，使之应用于更为广泛的领域。为此，IFIP(International Federation for Information Processing，国际信息处理联合会)成立了一个由 24 人组成的工作组来研究 Algol 60 语言，并开始设计 Algol 60 的下一代语言。

随着工作的开展，针对如何设计 Algol 60 的下一代语言，工作组内出现了两种不同的意见，从而形成了两个不同的派别，即激进派和实用派。激进派主张打破 Algol 60 的框架，重新设计一个全新的语言，能够为各类人员使用，从而使新语言能够在程序设计语言发展史上树立一个新的里程碑。而实用派则主张保留 Algol 60 的主体和语言的顺序结构，在新的语言中扩充程序员易理解的结构特征。基于这种思想，除了基本数据类型外，还提出了双精度实数类型、复数类型、枚举类型、以及记录类型等，并将 Algol 60 参数的按名调用扩展为参数的按址调用，循环语句改为更为严格的有效形式。

1965 年，作为实用派的主要人物之一，瑞士的 Niklaus Wirth(以下简称 N. Wirth，沃思)负责提交了反映实用派一方的报告。然而，N. Wirth 的报告并没有引起人们的认真对待，很多人反而欣赏荷兰人 A. R. Wijingaaren(A. R.维京格尔滕)的报告。在 1966 年秋天的华沙会议上，会议决定采用 A. R. Wijingaaren 的报告作为 Algol 60 的下一代语言(被称为 Algol X，也就是后来对程序设计语言发展有着重要意义的 Algol 68)的基础，只有 E. W. Dijkstra 等少数几个学者发表了一个声明，支持 N. Wirth 的结构化程序设计语言 Algol W(W 是 N. Wirth 姓氏的首字母)。遗憾的是，Algol X 由于过分追求功能齐全，适应各行各业的需要，语言设计得十分庞杂，引入了许多新的语言成分，语法表达和语义描述非常复杂，结果导致编译系统迟迟无法推出，读者理解和推广也很困难。因为时间上的延误，该语言的编译系统直到 1970 年代初才得以完成，并交付用户使用。而此时，另一种语言 Algol W 早已推向社会，被程序员广泛接受。与此同时，N. Wirth 仍然坚持自己的初衷，继续完善他的报告，并将英国计算机科学家、后来的图灵奖获得者 G. R. Hoare 教授提出的动态数据结构和指示字约束的概念吸收进去，发展了指针类型，并在美国国家科学基金会的支持下，于 1966 年在斯坦福大学的 IBM 360 机器上实现。N. Wirth 提出的语言称

为 Algol W。重要的是 Algol W 在 Algol 60 的基础上增加了表示双精度浮点数和复数等新的数据结构，以及位串和指针链接的动态数据结构，受到了用户的广泛好评。但是，由于 Algol W 缺乏表示复杂数据的类型，使得它没有成为系统程序设计的工具。于是，N. Wirth 教授决心摆脱以前的束缚，设计一种更具有普遍意义的高级语言。

1968 年，N. Wirth 在苏黎世联邦高等工业大学任副教授时发现，该校很多研究人员在进行数值计算时采用 Algol 60 语言编写程序。那时，学校新引进了 CDC 6600 计算机系统，而该机器上运行的 Algol 60 编译系统存在很多错误，使得他们不得不放弃 Algol 60 语言，并且在进行程序设计尤其是系统程序设计的教学中，只好选用 Fortran 语言和汇编语言作为编程工具。当时，计算机科学界正在热议“软件危机”，结构化程序设计风格开始倍受推崇，而他设计的 Algol W 语言在融入了结构化程序设计的思想和成分之后，能够有效缓解“软件危机”带来的矛盾和问题，受到程序员的欢迎。为此，很多大学相继采用该语言进行程序设计教学，使得 Algol W 语言很快流行开来。后来，他将该语言定名为 Pascal。1970 年，N. Wirth 教授发表了具有结构化程序设计风格的 Pascal 语言的定义，同年由 V. Ammann、E. Marmier 和 R. Schild 完成了其编译程序的设计和实现。1971 年，Pascal 语言开始出现在大学高级语言程序设计的课程中。作为语言设计大师，N. Wirth 后来还设计了一个著名的面向模块化程序设计的高级语言 Modula-2，该语言对整个学科的发展影响深远。因为对程序设计语言和程序设计的贡献，N. Wirth 在 1984 年获得图灵奖。

在 Pascal 语言的定义公开以后不久，出现了很多有关 Pascal 语言及其编译程序的文献，其作者大多并非 CDC 6600 计算机系统的用户。这使得 N. Wirth 认识到，必须设计一种合适的新体系结构，采用 V. Ammann 设计的编译程序给这种理想的体系结构产生代码。1973 年，这种体系结构被人们称为 P-machine，代码为 P-Code，编译程序为 P-compiler，而 P-kit 是指 P-Code 的编译程序和 Pascal 源程序的翻译程序。这种 P 系统的出现使得 Pascal 得以运行在其他机器上。但是，由于它的效率不高，使得它在当时只能适用于教学。

1975 年，加利福尼亚大学 K. Bowles 小组(UCSD)接受了 P-Kit 的思想，将 Pascal 语言的编译程序移植到微型计算机上。当时微型计算机也只是刚刚流行，K. Bowles 小组在移植 Pascal 编译程序的同时，建立了包括程序编辑、文件系统和调试工具的一整套系统。1978 年开始，这种 UCSD 的 Pascal 系统就被众多的用户所接受，从此，Pascal 语言开始走进工业界。1980 年，Three Rivers，HP，Apollo，Tektronit 四个主要的工作站生产商都采用 Pascal 作为其系统配置的程序设计语言。

P 系统不仅使 Pascal 语言得以推广，而且向人们展示了一种功能强，可移植性好并且可靠性强的编译程序。很多程序员开始学习 P 系统，并且将这个编译程序移植到其他机器上，如 ICL1990、IBM360、PDP-11、PDP-10 等。到了 1973 年，Pascal 语言已广为人知。

1974 年，Minnesota(明尼苏达)大学计算中心 Pascal 兴趣小组公开出版了有关 Pascal 的刊物，宣传 Pascal 语言的思想和成果。1977 年至 1979 年，英国、美国、ISO(International Organization for Standardization，国际标准化组织)工作组分别召开会议拟订 Pascal 标准。1983 年，美国制定了 ANSI X 3.97 标准，ISO 不久也制定 ISO 7185 标准。与此同时，很多公司也在发展自己的 Pascal 语言。1986 年，完整的 Pascal 语言标准公布于众。

Pascal 语言对于程序设计语言的设计和发展有很大的影响，并激发了众多新语言的产生。1975 年，P. B. Hansen 设计了并发 Pascal 语言，引入了并发进程和同步原语的概念。基于同样的目的，J. Welsh 和 J. Elder 发明了 Pascal Plus 语言，这种语言特别强调了基于并发进程激发分立

事件。B. W. Lampson[①]等人又设计了一种大型的语言 Mesa，其目的是要提供现代大型软件工程所需要的所有功能，为此引入了模块化的新概念，它的编译程序采用分块编译的原则。

抽象数据类型(Abstract Data Type)的思想和这种思想的结合也应用于以后的 Modula-2 语言的设计中。与 Mesa 语言不同的是，Modula-2 语言简洁、经济而紧凑。Pascal 语言的另一个分支是 Euclid 语言，它是基于形式化定义的面向数学家的程序设计语言。而 Object Pascal 语言是 Pascal 语言与面向对象程序设计概念相结合的产物，可称之为面向对象的 Pascal 语言。

在 Pascal 语言问世以来的几十年间，主要有 5 个版本的 Pascal，它们分别是 Unextended Pascal、Extended Pascal、Object-Oriented Extensions to Pascal、Turbo Pascal 和 Delphi Object Pascal。其中，Unextended Pascal、Extended Pascal 和 Object-Oriented Extensions to Pascal 是由 Pascal 标准委员会所创立和维护的，Unextended Pascal 类似于瑞士 N. Wirth 教授和 K. Jensen 于 1974 年联名发表的 Pascal 用户手册和报告，而 Extended Pascal 则是在其基础上进行了扩展，加入了许多新的特性，它们都属于正式的 Pascal 标准。Object-Oriented Extensions to Pascal 是由 Pascal 标准委员会发表的一份技术报告，在 Extended Pascal 的基础上增加了一些用以支持面向对象程序设计的特性，但它属于非正式的标准。美国 Borland 公司(宝蓝公司)对 Pascal 语言的推广和发展做出了很大贡献。该公司成功开发了 Turbo Pascal 和 Delphi Object Pascal(简称为 Delphi)两个版本。Turbo Pascal 是基于 CP/M、DOS 和 Windows 3.x 等不同操作系统的 Pascal 语言的系列扩展版本，而 Delphi Object Pascal 是用于 Windows 的 Delphi 和 Linux 的 Kylix(本意是古希腊的一种带耳朵的酒杯)的面向对象程序设计语言，它们都不是正式的 Pascal 标准。

Turbo Pascal 1.0 是由 Borland 公司的创始人 Philippe Kahn(菲利普·卡恩)和 Anders Hejlsberg(安德斯·海尔斯伯格)两人于 1983 年合作开发的，它是 Turbo Pascal 的第一个编译器。1989 年，Borland 公司把面向对象程序设计的思想引入到 Pascal 语言中，推出了 Turbo Pascal 5.5，标志着 Object Pascal 的开端。Turbo Pascal 的最后一个版本是 Turbo Pascal 7.0，它于 1992 年 10 月发布，包含一个增强的 DOS 下的 IDE(集成开发环境)和编译器，可以创建 DOS 和 Windows 程序，后来被 Borland 公司于 1995 年 2 月发布的 Delphi 1.0 代替。

Delphi(这里借用了古希腊神话中一个智慧女神的名字)继承了 Pascal 语言简单易学的特点。它拥有一个可视化的集成开发环境(IDE)，采用面向对象的编程语言 Object Pascal 和基于构件的开发结构框架，加上高速的编译器，强大的数据库支持，使之成为第四代编程语言。“真正的程序员用 VC，聪明的程序员用 Delphi”，这句话是对 Delphi 最经典、最实在的描述。可以说 Delphi 同时兼备了 Visual C++功能强大和 Visual Basic 简单易学的特点。Delphi 发展至今，已经从 Delphi1(16 位编译器)、Delphi2(32 位编译器)到现在的 Delphi XE6，不断添加和改进各种特性，功能越来越强大，是目前市场上最流行的可视化编程的主流环境之一。

另外，由于用户不满足当时 Turbo Pascal 这一 16 位机器的编译器，于是人们开发了 32 位计算机上的 Pascal 编译器——Free Pascal。Free Pascal 简称 FPC(原名为 FPK Pascal)，是一个 32/64 位计算机上的 Pascal 及 Object Pascal 编译器，属于开源软件。它兼容了 Turbo Pascal 和 Delphi 等系统，并支持多种处理器(例如，Intel x86，AMD 64/x86_64，Power PC，Power PC 64，Sparc，ARM)和多种操作系统(例如，Linux，FreeBSD，NetBSD，Win32，Win64，OS/2，BeOS，SunOS，QNX 等)。Free Pascal 1.0 于 2000 年 7 月正式发布，2014 年 10 月的最新版本是 Free

① B. W. Lampson，巴特勒·兰普森，1942 年生于美国华盛顿。1964 年获得哈佛大学物理学士学位，1967 年获得 UC Berkeley 电子工程与计算机科学博士学位。因在分布式计算环境、个人计算环境的研发和实现技术，其中包括工作站、网络、操作系统、程序设计语言、计算机显示、计算机安全和计算机文档排版处理等方面取得的成就而获得 1992 年图灵奖。

Pascal 2.6.4。如今，Free Pascal 已经成为了一个跨平台的编译器，真正实现了自己提出的“写一次代码，在各处编译”的理想。

创立于 1970 年代初期的 Pascal 语言，是第一个体现结构化程序设计思想的高级语言。设计者 N. Wirth 教授的本意是寻求一种有益于教学，且易于高效实现的程序设计语言。由于它语句简明，数据类型丰富，程序结构严谨，因而其效果已远远超出了设计者的初衷，成为程序设计语言发展史上的一个里程碑。时至今日，Pascal 语言已成为世界上最通用的程序语言之一，为各种通用计算机系统所必备。实际上，用 Pascal 语言书写程序，有助于培养良好的程序设计风格，因此，它被公认为计算机科学与技术专业人才培养的最佳入门语言和教学语言。

Pascal 语言的发展，带给读者如下几点启示：

(1) 科学研究始于科学问题。要善于从学科的发展和实际工作的需求中发现科学问题；

(2) 科学史上的任何发明创造都有其客观背景和演变过程，Pascal 语言的产生和发展也是这样；

(3) 计算机科学的方法与技术的发展、软硬件系统性能和程序设计环境的发展推动了程序设计语言的发展；

(4) 没有积累就没有创新，没有继承就没有发展。

1.2.2　Turbo Pascal 的特点

与之前的高级程序设计语言相比，Turbo Pascal 7.0 的主要特点是：

(1) 是集命令-过程型、面向对象型为一体的多范型程序设计语言；

(2) 编译器占用内存少，编译速度快；

(3) 操作环境极佳，在同一屏幕内可进行编辑、编译、运行、连接和调试等操作。这样的操作环境被称为集成开发环境(Integrated Development Environment)，简称为 IDE；

(4) 支持多种新的数据类型，具有高效的数值运算能力，具有与底层软件和硬件打交道的能力，支持 Windows 程序设计；

(5) 提供了几百个内部预定义的标准子程序，具有强大的图形图像处理功能；

(6) 引入了单元的概念，支持模块化程序设计；

正是它的以上诸多优点，使得 Turbo Pascal 7.0 已经成为大型软件系统开发的主要工具之一。

1.2.3　Turbo Pascal 的符号、约定

1．Turbo Pascal 的字符集合

Turbo Pascal 7.0 的字符集合(Character Set)由字母、数字和其他一些字符构成。它是 ASCII 码(American Standard Code for Information Interchange, 美国信息交换标准代码)的子集。具体如下：

(1) 字母(27 个)

26 个英文字母和_(下划线)。除了在字符串中外，Pascal 语言字母的大小写被视为相同的。

(2) 数字(10 个)

0　1　2　3　4　5　6　7　8　9

(3) 其他(23 个)

+ - * / = < > .,′: ; ^ ( )[ ] { } ⊔(表示空格)# @ $

2．Turbo Pascal 的符号

字符集合中的若干字符可以构成符号(Token)，又被称为单词(Word，即通常语言中的“字”)。在程序设计语言中，有一些单词已经被预先定义成为具有特定意义的符号，这些单词不允许再被程序员重新定义而另作它用。它们称之为保留字(Reserved Word)或关键字(Key Word)。一般情况下，对保留字和关键字不再加以区分。

在程序设计语言中，关键字和保留字是两个不同的概念。从字面含义上理解，保留字是语言中已经被预先定义过的单词(字)，它可以没有被应用于当前语言版本的语法中，而是为了考虑到今后语言的可扩展性被预先定义。关键字也是语言中预先定义为具有特定意义的单词(字)，但它必须是当前语言版本语法的组成部分。例如，当前版本的 Javascript 有一些未来保留字，如 abstract、double、goto 等，它们并不是关键字。之所以把 goto 定义成保留字，是因为考虑到未来要使 Javascript 增加直接跳转功能。由于当前版本的 Javascript 不支持 goto 的直接跳转功能，而 goto 已被定义成保留字，使得在当前的程序中不允许将 goto 另作它用，这样就做到了使当前版本的程序代码能够向后兼容。

(1) 保留字(特定符号字，51 个)

Turbo Pascal 7.0 共有 51 个保留字，除了标准 Pascal 的 35 个外，还扩充了 16 个。标准 Pascal 的 35 个保留字是：

and array begin case const div do downto else end file for function goto if in label mod nil not of or packed procedure program record repeat set then to type until var while with

扩充的 16 个保留字是：

asm constructor destructor exports implementation inherited inline interface library object shr string shl uses unit xor

(2) 定界符(非字特定符号，26 个)

Turbo Pascal 还有一些不是表示成字的特定符号，这些符号用来确定语法单位，称它们是定界符(分隔符(Separator)，26 个)：

+ - * / < <= > >= = <> ^ . .. : , ; := ′ # @ $ (和) [和] (.和 .) {和} (*和*)

(3) 标识符

标识符(Identifier)的概念是由英国著名计算机科学家、图灵奖获得者威尔克斯(M. V. Wilkes)提出的。其功能是在程序中用来标识(表示)符号常量、变量、类型、过程、函数、程序、文件的名字。所谓标识符，就是程序员根据自己的需要在程序中自己定义(造)的字(单词)。

Turbo Pascal 规定，标识符是由英文字母或下划线开头，后面是英文字母、数字、下划线的任意组合，且有效长度不超过 63 个字符(大小写相同)的字符序列。Turbo Pascal 中除了字符串(String)之外，不区分大小写字母，所以，标识符 DATA 与 data 是同一标识符。

Turbo Pascal 中的标识符被分为标准标识符和用户自定义标识符两种。

① 标准标识符

标准标识符(Standard Identifier)是 Turbo Pascal 预先定义好的标识符，它们有特定的意义，用来作为常量名、类型名、过程名、函数名、文件名等，因此又称为预定义的标识符。Turbo

Pascal 中常用的标准标识符有：

标准常量名：false、true、maxint、maxlongint；

标准类型名：shortint、integer、longint、byte、word、real、single、double、extended、comp、char、boolean、text；

标准过程名：new、dispose、mark、release、assign、reset、rewrite、seek、truncate、read、readln、write、writeln、close、randomize、delete、insert、str、val；

标准函数名：abs、sqr、sqrt、exp、round、sin、cos、arctan、trunc、succ、pred、chr、ord、ln、odd、eof、eoln、filesize、truncate、filepos、random、copy、concat、length、pos、ptr、sizeof；

标准文件名：input、output。

② 用户自定义的标识符

用户可以根据自己的需要，遵循下面语法定义标识符。这种标识符称为用户自定义的标识符。Turbo Pascal 中标识符的语法图如图 1-1 所示。

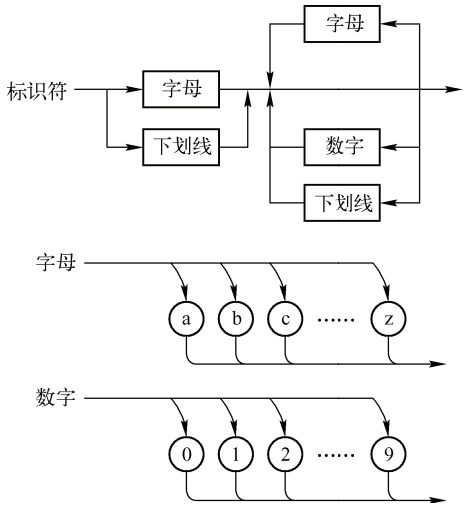


图 1-1　标识符的语法图

用户自定义标识符时，应注意以下几点：

a）在 Turbo Pascal 的程序中，用户自定义的标识符有两种出现的情况。一种情况是定义性出现，另一种情况是引用性出现。一般情况下，定义性出现必须先于引用性出现，这就是所谓的“先定义，后引用”的原则。如果不定义就引用标识符，编译程序会报告程序错误。

b）Turbo Pascal 中，在一个标识符中不能出现语法上不允许的字符(包括定界符和空格)。例如，a&b、a⊔b、my+name 等作为标识符均是错误的。

c）由于保留字(含关键字)在程序中具有特定的语法含义，因此禁止使用保留字作为用户自定义的标识符。例如，将 begin、for 等用作表示程序实体的标识符是错误的。

d）从语法上讲，允许使用标准标识符作为用户自定义标识符。但是，为了保证程序的可读性，避免引起错误，建议不要使用标准标识符作为用户自定义的标识符，许多编译程序也禁止用户将标准标识符另作它用。

e）标识符的命名风格是程序设计风格(Programming Style)的一个组成部分。程序设计风格

中的“风格”是指程序员在创作中喜欢和习惯使用的表达自己作品题材的方式，并形成统一的风尚。通俗地说，程序设计风格是指，为了使写出的程序容易被人们阅读、理解和使用而建立的一整套约定、准则、方法和规定，等等。程序的可读性即可理解性。衡量程序的可读性的原则是理解程序代码时需要关心和记忆的知识越少越好。为了增强程序的可读性，用户自定义标识符时应该遵循以下四个原则：

第一，按义取名的原则，即所定义标识符的名字能够完全而又准确地描述它所代表的问题领域的实体对象，或者说用户命名的标识符最好是看到名字就能知道它所表示的含义。例如，用 Area 表示某程序中用到的面积变量，用 Time_of_Day 表示一天所含有的时间变量。

不过，在程序中使用标识符所代表的问题领域实体对象描述作为标识符名字时，可能带来标识符名字过长的问题。例如，Student_number_of_xiamen_university。标识符名字过长不但使程序的输入变得困难，而且也给程序布局带来不便。为此，**建议标识符的命名最好是在描述其所代表的问题领域实体对象的前提下,尽可能使其长度少于 20 个字符。**为了将过长的标识符的长度缩短到少于 20 个字符，常使用缩写技术。下面是常用的缩写技术：

a）使用标准的缩写，例如字典缩写表中的标准缩写；

b）使用每个单词的头一个或几个字母，一般来说，单词的首部比尾部重要；

c）去掉无用后缀；

d）保留单词每个音节中最容易引起注意的发音，一般辅音比元音重要；

e）保留标识符中具有典型意义的单词。

第二，由名字能够区分标识符种类的原则。Turbo Pascal 的标识符有符号常量、变量、类型、过程、函数、程序和文件等七种名字。符号常量、变量、类型、程序和文件的名字应该是名词或名词词组；函数和过程名字应该是动词或动词词组，最好是动宾词组；变量名字最好能够体现其所属的数据类型等等。

第三，体现其作用域的原则。标识符的名字最好能够体现其作用域，特别地，要突出其是否为全局变量。当然，应该在程序中尽可能地少使用全局变量。一旦在程序中使用了全局变量，那么最好能把它们显式地表示出来。例如，每个全局变量均以 global 作为其前缀，等等。

第四，规范化的书写原则。标识符规范化的书写技术是用下划线或大写字母将名字的各单词区分开来。例如，student_name 或 StudentName 显然比 studentname 可读性好得多。

最后，下面列出一些应避免的标识符名字：

a）应避免容易产生误会的名字或缩写。如果程序同时用 Compact_disk 和 Current_date 两个变量，那么最好不要把任意一个缩写为 CD；

b）应避免含义相同或相近的名字，如最好不要同时使用 input 和 in_val；

c）应避免含义不同但拼写相似的名字，如 c1ient_records 和 c1ient_reports，不应该用 client_recs 和 client_reps 来表示；

d）应避免使用发音相同或相近的名字，如 know 和 now；

e）应避免在名字中使用数字，如 file1，file2 等，尽量使用其他方法区别这种变量，而不是仅仅在后面加上不同的数字；

f）应避免同时使用含有难以辨认的字符，如数字 1、字母 l 和字母 I，数字 0 和字母 o 或 O，数字 5 和字母 S 或 s，数字 6 和字母 G 等；

g）应避免使用汉语拼音作为标识符名字，更不要使用汉语拼音缩写，因为它们破坏程序的协调性。程序设计语言多源自英文，插入一个汉语拼音会显得很不自然，将来程序也难以对外交流。

1.2.4　Pascal 源程序的结构

Turbo Pascal 的源程序(Source Program)结构是有严格规定的，而且也是固定的。为了说明 Pascal 语言源程序的结构，先看一个简单的例子。

例 1-1　下列给出了自动生成杨辉三角形的一个程序。

```
{  程序名称：YanghuiTriangle1
   文 件 名：YanghuiTriangle1.pas
   作　　者：赵占芳
   创建日期：2012-01-20
   程序功能：生成并输出杨辉三角形。
}
Program YanghuiTriangle1(input,output);
Const
  n=10; {The value of n is the line numbers of yanghui's triangle.}
Type
  yht=array [1..n,1..n] of integer;
Var
  row: 1..n ;
  column: integer;
  indent: integer;
  yh: yht;

Procedure generate_yh_Triangle; {生成杨辉三角形}
begin
  for row:= 2 to n do
    begin
      yh[row,1]:= 1;
      yh[row,row]:= 1;
      for column:= 2 to row-1 do
        yh[row,column]:= yh[row-1,column-1]+yh[row-1,column];
    end
end;

Procedure print_yh_Triangle; {打印杨辉三角形}
begin
  indent:=30;  {每一行开始打印的空格数}
  for row:=1 to n do
    begin
      write(' ':indent);
      for column:=1 to row do
        write(yh[row,column]:6);
      writeln;
      indent:=indent-3;
    end
end;

Begin  {主程序}
```

```
    yh[1,1]:= 1;
    generate_yh_Triangle;
    print_yh_Triangle;
    readln
End.
```

从上面的程序可以看出，Turbo Pascal 源程序的结构分为两部分：**程序首部**和**程序体**。

1．程序首部

程序首部(Program Heading)是指程序的开头部分，它是由保留字 Program、程序名和程序参数表(Program Parameter List)三部分构成，并以分号结束。程序名是用户自定义的标识符，用于标记程序。用户命名时要严格遵守标识符的有关规则。程序参数表标明程序与外部环境之间的关系。在现代计算机系统中，这个外部环境通常指操作系统(Operating System)。最常用的参数是 input 和 output，它们是两个标准文件(Standard File)，分别表示程序的输入数据是从标准文件——键盘——输入，计算的结果将输出到标准文件——显示器——荧屏上。Turbo Pascal 规定：当程序参数表中的两个参数为 input 和 output 时，可以省略程序参数表。例 1-1 的程序中，程序首部可以简写为：

```
Program YanghuiTriangle1;
```

另外，Turbo Pascal 7.0 允许在程序首部和程序体之间加 uses 字句，说明被程序直接和间接使用的单元。

2．程序体

程序体(Program Body)是由程序说明部分和程序执行部分构成。

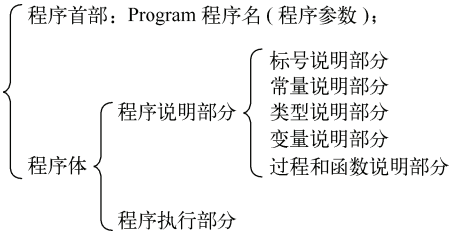
(1) 程序说明部分。

程序说明部分(Program Declaration Part)是对程序的计算过程中用到的所有标识符的说明。Turbo Pascal 规定：用户凡是在程序中使用的标号、常量、类型、变量及过程和函数，除了语言本身预先定义的标准量之外，都必须在程序的说明部分说明(定义)之后才能使用。其程序说明部分中，在不违背“先说明(定义)，后引用(使用)”的原则下，标号说明、常量说明、类型说明、变量说明、过程说明和函数说明的顺序是任意的。例 1-1 的程序中，说明部分分别为：常量说明、类型说明、变量说明及过程说明。关于说明语句的详细介绍见第二章和第四章。

(2) 程序执行部分。

程序执行部分(Program Executable Part)是指紧跟在程序说明部分之后用 begin 和 end 括起来的部分。end 之后以句点结束，它是整个程序的结束标志。该部分是由一系列语句构成，语句之间用分号隔开；每一行可以写一条语句，也可以写多条语句。程序执行部分描述了程序的计算过程，它是 Turbo Pascal 语言源程序的核心部分，也是程序中不可缺少的部分。

综上所述，Turbo Pascal 源程序的结构如下：



下面的图 1-2 给出了 Turbo Pascal 源程序的语法图。

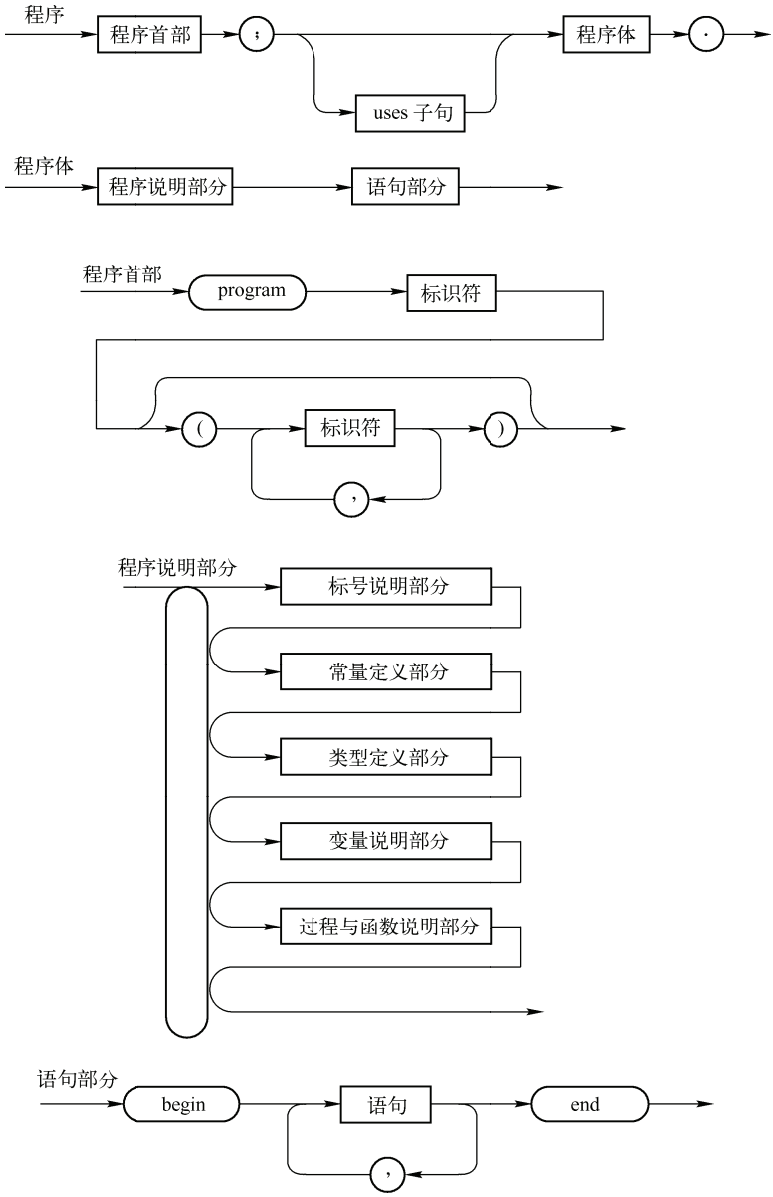


图 1-2　Turbo Pascal 源程序的语法图

3．附注

(1) Turbo Pascal 允许省略程序首部。建议读者书写程序时不省略程序首部为宜。

(2) 为了增强程序的可读性(Readability)，便于调试(Debugging)程序，程序的书写形式一般采用缩进(Indentation)格式(或称为锯齿格式)，它是良好的程序设计风格(Programming Style)的组成部分。建议读者书写程序时一定采用这种格式。本节例题的程序书写时就采用了这种格式。

在程序设计书写时，要注意 Turbo Pascal 不区分英文字母的大小写，但有的编译系统规定了区分大小写。程序中的空格是为了便于语法处理和利于读者的阅读理解而引入的，本身并不是语

言的一部分。因此，读者在书写程序时应充分利用这一点，尽可能做到书写的程序美观。

程序的书写形式实质上是程序布局风格的问题。好的程序布局风格应使程序的布局体现程序的逻辑结构，做到不仅内容正确，而且形式也美观。关于程序的布局风格有许多细节问题需要读者在学习和实践过程中慢慢体会和总结。作为初学者，首先要建立一个正确的观念；其次，养成良好的程序设计风格十分重要！

(3) 注释

注释(Comment)是程序员为了增强所编写的程序的可读性而在程序的任何两个符号之间添加的一段注解。编译程序对程序的注释不进行任何处理。

Turbo Pascal 的注释是用“{”和“}”这两个配对花括号或者“(*”和“*)”这两个配对的复合括号括起来的任何长度的字符序列。用花括号括起来的注释的语法图如图 1-3 所示。

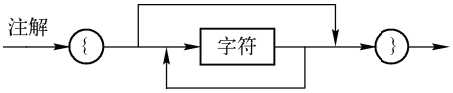


图 1-3 注释的语法图

在添加注释时应该注意以下几点：

- 注释中的字符不是指 Turbo Pascal 字符集中的字符，而是指字符类型的数据中的字符；
- 为了避免混乱，规定注释内不得包含花括号“{”和“}”；
- 注释可以出现在程序的任何两个符号之间，而不是任何两个字符之间；
- 注释行的多少，以一般人能够读懂为原则。

由于注释对程序的阅读、修改、调试和交流起着十分重要的作用，因此建议读者养成适当添加注释的习惯，它也是良好的程序设计风格的重要组成部分。好的注释风格可用注释来解释程序意图和一些重要的标识符或参数，使程序不仅表达怎样做，而且还表达为什么这样做，并记住这些标识符和参数的作用。本节例 1-1 的程序中就适当地加了一些注释。根据程序开发人员的经验，一般对于稍大一点的程序，程序中被注释的语句数量应该**不低于 50%**。能够规范地书写程序的注释，并具有这样一种能力是职业程序员的一种基本素养。印度软件公司的程序员在这方面有着良好的职业素养。最著名的例子是印度软件职业技术学院的程序设计与软件工程的教学。从那里毕业的程序员在程序设计风格和软件开发规范方面具有高度的一致性，对印度软件业的发展贡献良多。针对相同的算法，他们的程序员不仅在程序设计描述方面高度一致，甚至连程序注释也一样，令人赞叹，受到学术界和产业界的普遍赞赏，成为印度软件业能够迅速崛起、发展的重要基础。

(4) Turbo Pascal 的程序体中，程序说明部分的语义是静态的(Static Semantics)，而程序执行部分的语义是动态的(Dynamic Semantics)。也即程序说明部分的语义处理是编译时进行的，而程序执行部分的语义处理是程序运行时进行的。从哲学上讲，动与静是一对矛盾，在程序中两者相互依存，缺一不可，体现了对立统一的思想和规律。

本 章 小 结

由美国著名语言学家 Noam Chomsky(乔姆斯基)创立的转换生成语言学理论可知，语言是某个特定字母集合(表)上的句子构成的集合，进一步说它是由有穷的字(母)、数字、符号按照一定的语法规则组成的语义上正确的句子构成的集合。作为学习 Turbo Pascal 及其程序设计基础的开始，本章介绍了这种语言的字符集、符号、约定与解释，说明了它的特点，并通过一个简单的

Turbo Pascal 源程序的示例，说明了它的结构及其特点。

在学习高级语言程序设计的过程中，就语言部分，读者一定要与学习自然语言的过程加以对比，把高级语言中的基本概念与自然语言中的基本概念加以对比，从中体会和感悟，这样有助于读者对程序设计语言的理解。就程序设计部分而言，读者应经常与日常生活中的思想方法、工作流程等方式加以对比，把程序设计中的基本思想、概念、方法与日常生活中行为方式、处事风格加以对比，这样有助于对程序设计的理解。

习　　题

1．下面的字符序列哪些是 Turbo Pascal 的标识符？

（1）exp　（2）h(x)　（3）ex_1　（4）x[1]　（5）_x*y　（6）20021225x

（7）integer-1　（8）$liu　（9）true　（10）very good

2．举例说明在 Turbo Pascal 中什么是符号，什么是字符，它们之间有何关系？

3．Turbo Pascal 语言的标准标识符有哪些？保留字有哪些？它们的差别是什么？

4．请你将 Turbo Pascal 语言中的词法、语法、语义、语用、符号、程序与汉语言中的相应的概念作对比，你有什么体会？