最新 FORTRAN 程序设计实用教程

天津科学技术出版社

最新FORTRAN程序设计实用教程

高福成　姜玉泉　梁静毅　编著

天津科学技术出版社

津新登字(90)003号

责任编辑:刘万年

最新 FORTRAN 程序设计实用教程

高福成　姜玉泉　梁静毅　编著

*

天津科学技术出版社出版

天津市张自忠路189号　邮编300020

河北省唐山市印刷厂印刷

新华书店天津发行所发行

*

开本787×1092毫米　1/16　印张21　字数510 000

1993年2月第1版

1993年2月第1次印刷

印数:1—5 000

ISBN　7-5308-1219-X/TP·38　定价:11.00元

内 容 提 要

本书以应用为目的，以算法为主线，以程序设计方法学为依据，系统地介绍了FORTRAN77程序设计技术。全书共六章，分别介绍了程序设计的基本概念、基本思维原则，结构化和模块化程序设计方法与技巧，以及以文件为基础的输入、输出程序设计。FORTRAN77语言知识的介绍则根据程序设计的需要，穿插其中。书中配以大量的程序实例及习题，以数值算法为主，兼顾非数值算法；以通用程序为主，普及与提高兼顾，可以满足不同读者的需要。

本书注重实用，内容深入浅出，循序渐进，方便自学，可作为高等学校教学用书，也可供有关工程技术人员和研究生参考。

前　言

FORTRAN 语言是最早问世的计算机高级语言。近40年来，FORTRAN 语言本身不断充实、完善，以它“擅长”数值计算而广泛流行、长盛不衰。迄今为止，FORTRAN 语言在科技计算中仍占统治地位，被广大科技工作者视为不可缺少的工具。

本书是在作者从事 FORTRAN 语言教学和应用实践的基础上编写的，在选材和内容的编排上注意了以下几点：

1. 任何计算机语言都是进行程序设计的工具。如何将语言和程序设计有机地衔接起来，许多专家、学者进行了有益的探索。我们知道，程序设计的任务是将实际问题的要求转化为程序，计算机语言则是人为规定的计算机指令，用某种程序语言描述程序设计的思维活动就生成程序。因此，程序设计并不一定要和某一种语言发生必然的联系。任何程序语言，哪怕是通用语言，都存在各自的规定和限制，人们为了淡化语言对程序设计的制约，总是先根据问题的要求进行程序设计，然后再选择能满足要求的语言。就是说，程序设计统帅和支配着程序语言。程序语言的产生和发展，淘汰和更新，都是由于程序设计的需要引起的。程序设计发展到今天，已经由技艺推进到科学，有了自己的一套基本原理和方法，其中比较成熟的有自顶向下，逐步求精的程序设计方法，结构化、模块化程序设计方法等。可以这样认为，语言是一种技能，程序设计是一门科学，我们应该把程序设计作为本课程的出发点和归宿，语言的教学应该服从于程序设计的需要。因此，本书以程序设计贯串始终，根据程序设计的需要安排语言知识的介绍，把程序设计作为科学来讲授，把语言作为技能来培养，让读者在大量的程序设计实践中自然而然地熟悉和掌握语言。

2. 进行程序设计，需要很强的逻辑思维能力，是一种极富创造性的智力劳动。对初学者而言，往往最使他们望而生畏的，也正是这一点。事

实上，电脑作为人脑的模拟，在许多方面与人的思维习惯是一致的。如能迅速地把人们的思维习惯与程序设计的思维方法吻合起来，无疑会提高他们学习的兴趣和信心。因此，本书通过大量简单程序设计的实例，详尽阐述了程序设计的基本思维原则，即枚举原则、归纳原则和抽象原则，帮助初学者很快掌握一批最基本、最常用的程序单元，使之能轻而易举地编写它们，得心应手地使用它们。

3. 程序设计是用计算机解决实际问题的关键环节，而不是全部。提高解决实际问题的能力，最好的途径是从实际问题入手，研究它们，解决它们。对科技方面的读者来说，最关心和最感兴趣的是科技计算；况且，FORTRAN 语言与数值计算有着天然的联系。因此，进行科技计算常用算法的程序设计，既发挥了 FORTRAN 语言的特长，又满足了读者的急需。本书把算法和程序设计方法结合起来，对每个算法，将算法原理、算法框图和对应程序联成一体，使读者能加深理解算法，提高根据算法编制程序的能力；同时，将各种算法分门别类，以使读者能分类掌握各种算法，便于了解其优劣和使用场合，增长根据实际问题选择算法的才干。

4. 一个程序的实用性往往表现在它的通用性上。本书提供的程序绝大多数是通用程序，都在 UNIX 系统和 DOS 系统上运行过，因此颇具实用价值，读者可以直接使用。

基于以上考虑，本书内容共分三个层次：①作为程序设计基本功训练的基本思维方法及基本程序单元的设计；②以科技计算常用算法为主线的实用程序、以文件为基础的输入/输出程序的结构化程序设计方法与技巧；③以大型程序为目标的模块化程序设计技术。全书内容按先简单程序后复杂程序、先单一程序模块后多个程序模块的顺序，所有算法也按先简后繁的顺序编排，数值算法和非数值算法并蓄，普及与提高兼顾，读者可根据需要进行取舍而不影响完整性。

全书共六章及两个附录，前五章由高福成编写，第六章和附录由梁静毅和姜玉泉编写。全书由高福成统编和定稿。

本书承天津大学李宗耀副教授、天津商学院李延年副教授审校。在

编写过程中，得到天津商学院院长李炳威同志和天津商学院计算机应用教研室全体同志的支持和帮助，在此表示衷心的感谢。

由于作者水平有限，且成书仓促，错误及疏漏之处，敬请批评指正。

编著者

1992年3月

目　录

第一章 程序设计基本概念

程序设计是计算机科学的一个核心问题。计算机作为一种现代化的高速计算工具，随着科学技术的不断进步，它的工作性能、技术水平正在日益提高，其应用领域也在不断扩大。计算机的应用大致可分为科学计算、数据处理、实时控制、人工智能几大类。不管是在哪一个应用领域，用计算机解决一个实际问题，都必须经过下面几个步骤：首先从实际问题抽象出一个数学模型，然后设计一个解此数学模型的算法，最后编出程序、进行调试、测试直至运行而获得最终解答。

由于计算机的一个重要特点就是运算的高速度，这是人工手算所无法比拟的。为了充分发挥计算机的利用效率，在机器解题的过程中，应当尽量减少人工的干预，实现操作自动化。为此，上机解题之前，必须先将所制订的解题方案"告诉"计算机，让计算机按照人们所规定的计算顺序去自动地执行。这样，就必须用计算机所能接受的"语言"来描述解题步骤，这项工作就叫做程序设计。离开程序设计，就谈不上计算机应用。也可以说，计算机要动作，就必须有程序。程序设计是一项艰苦的脑力劳动。目前，程序设计主要还是靠人完成。人们在程序设计方面已做了大量的研究，取得了许多重要的成果，但是，依然面临许多困难。所谓的"软件危机"就是这些困难的集中表现。程序设计是人类面临的一个智力挑战，程序设计在质和量两方面的复杂性和困难性，使得程序设计工作将永远有创造性。

本章将简要介绍与程序设计有关的几个基本概念，使读者对程序设计有一个完整的认识。

第一节 数学模型

数学模型是用数学方法描述要解决的实际问题。把实际问题抽象为一个数学问题的过程称为建立数学模型。寻求数学模型的实质是分析问题，从中提取操作的对象以及这些操作对象之间存在的联系，然后用数学语言加以描述。

一、数学模型可以分为两大类

一类是数值模型。在科学及工程计算领域中，一个物理现象或化学现象的变化规律，往往可以用一组代数方程或微分方程来描述；在企业管理领域中，大量使用线性规划、整数规划、动态规划以及非线性规划等运筹学模型；在过程控制领域中，则常常要把观察到的一系列离散的数值通过插值或拟合的方法形成某种函数。上述这些数学模型都是一些数学方程或函数，其中涉及到的数据一般都是整数、实数或复数，对数据的操作主要是计算，这就是数值模型。

另一类是非数值模型。在事务处理、情报检索等信息处理领域中，大多数问题无法用数学方程加以描述，而是采用诸如图、树、表等非数值型的数学模型；在人工智能领域中，则大量使用"事实"和"规则"这样的描述方法。这类数学模型涉及到的数据除了数值之外，还有大量的符号，对数据的操作主要不是计算，而是数据的采集、存贮、传输、变换及检索等，这就是非数值模型。

二、FORTRAN 语言主要是针对数值模型的

不同的专业，或者同一专业不同的实际问题，其数学模型可能极不相同。例如，在电学领

域，经常要遇到常微分方程初值问题；在机械及力学领域，通常是偏微分方程问题。即使在同一电学领域，微波方面的问题，就是偏微分方程问题了。因此，数学模型与相关专业密切联系在一起，数学模型的好坏，直接影响到求解的质量；数学模型的准确性也是反映建模者理论水平高低的具体表现。同时，数学模型的准确性往往与其复杂性紧密相关，越准确往往越复杂；而模型越复杂，求解就越困难，花费的计算时间就越多。因此，需要在模型的复杂性及计算时间上进行权衡、折衷，抓住主要因素，摒弃次要因素。另外，对要解决的问题进行一番系统而周密的探索，收集足够的资料，建立一个可以检测、评估的模型。理想的模型，通常是一个解析模型，或者是一个附有解析意义的标准模型。

为了提高建立模型的水平，必须充分研究和掌握本专业的问题特性和规律，加强基础理论的学习，充分开展预先研究，并且还要掌握足够的数学方法。

第二节　算　　法

数学模型建立以后，就要根据特定的数学模型拟定计算方案和规划计算步骤。不过，在人工手算时，解题步骤不一定要写成文字的形式，可以留在人们的头脑里酝酿。在解题过程中，人们常常随时处理各种“意外”情况，充实或者修改原定的方案。由于计算机只能机械地执行人们的指示和命令，不会主动地进行思维，不可能发挥创造性，因此，交给计算机执行的解题方案中的每个细节都必须准确地加以定义，并且全部解题过程应当完整地描述出来。这种解题方案的准确而完整的描述，就是我们通常所说的算法，或者称为“计算机方法”。

一、算法也可以分为两大类

一类是数值算法，其目的是求数值解。例如，数值代数主要研究线性和非线性方程组的数值解法、特征值的计算方法等，它以矩阵为主要工具；数值逼近主要研究如何用一个简单的函数近似地代替某个复杂的函数，或者把一系列的离散值“联成为”连续变量的函数，其主要手段是插值和拟合；而在求解常微分方程和偏微分方程时，则需要将连续变量的问题“离散化”为一系列离散点来进行计算。这些方法称为“计算方法”或“数值分析”，属于数值算法。数值算法的研究比较深入，算法也比较成熟，常常有现成的算法可以选用。但是，由于计算方法还未被许多人所掌握，尤其未被广大的工程技术人员所掌握，因而成为我国目前计算机在工程技术领域的应用尚不普遍的一个原因。

另一类是非数值算法。它们多用于信息处理领域及人工智能领域，例如，情报检索、多叉路口的交通灯管理、计算机下棋等。由于非数值运算的种类繁多，要求各异，很难统一或规范化；到目前为止，只有一些比较典型的非数值算法，例如排序算法、数据搜索算法等进行了比较深入的研究。

二、FORTRAN 程序主要使用数值算法

FORTRAN 语言主要用于科学和工程计算以及运筹学问题，因此，FORTRAN 程序主要使用数值算法。

下面，简述几个与算法有关的问题。

1. 算法的描述

描述一个算法可以采用多种方式，主要有：

(1)框图或流程图；

(2)类似于数学语言的自然语言；

(3)计算机语言。

前两种描述方法主要用于程序设计者之间的信息交流，其优点是清晰了然，易于理解，易

于修改，经常用在算法构思阶段；第三种描述方法主要用于程序设计者和计算机之间的信息交流，用计算机语言描述的算法也就是计算机程序。

2. 算法的优劣

计算机的特点是运算速度高，存贮的信息量大，并能自动地完成极其繁复的计算过程。

计算机的功能虽然很强，但如果算法选择不当，计算机的利用效率就得不到充分的发挥。例如，用克莱姆(Cramer)法则求解 n 阶线性方程组，要计算 n+1 个 n 阶行列式的值，总共需要 n!(n−1)(n+1)次乘法。按这个计算量，求解一个 20 阶左右、规模不算太大的方程组，即使用每秒千万次的计算机，也得要连续工作千万年才能完成。当然，这是完全没有实际意义的。因此，计算量的大小是衡量算法优劣的一项重要标准，它标志着计算效率的高低。

设计算法所要考虑的另一个因素是算法逻辑结构的复杂性问题。虽然计算机能够自动地执行极为复杂的计算方案，但计算方案的每个细节都是需要人来制订的。从计算人员的角度来讲，总是希望算法的逻辑结构简化，容易验证，便于编制程序。

算法的精度是挑选和设计算法的重要依据。

在进行数值计算时，误差是不可避免的。误差的产生主要来自三个方面：

(1)在将实际问题归结为数学模型时，通常总要加上许多限制，总要忽略一些次要的因素，这样建立的“理想化”的模型，虽然具有“精确”而“完美”的外表，其实只是客观现象的一个近似而粗糙的描述。这种数学描述上的近似必然会产生误差。

(2)计算当中遇到的数据可能位数很多，甚至会是无限不循环小数或无穷小数，然而受机器字长的限制，用机器代码表示的数据必然舍入成一定的有限的位数，这就会引起舍入误差。少量运算的舍入误差一般是微不足道的，但在计算机上完成了千百万次运算之后，舍入误差的积累就可能很惊人了。

(3)许多数学运算，如微分、积分及无穷级数求和等，是通过极限过程来定义的。然而计算机只能完成有限次的算术运算和逻辑运算。因此，需要将解题方案加工成算术运算与逻辑运算的有限序列，这就使一种无穷过程产生截断而变成有限过程，由此产生的误差通常称作截断误差。

由上可见，在数值计算的过程中会出现各种不可避免的误差。有时，误差会严重“泛滥”，而完全“淹没”了所需要的真值。因此，进行任何一项计算，必须首先在满足精度要求的前提下，选择和设计好的算法。例如，两个相近的同符号数之差作除数，就可能造成相当大的误差，这在算法上应该尽量避免。

算法的数值稳定性对一些坏条件(即刚性)问题的影响特别重要，即使对一些好条件问题，采用不同的算法，由于数值稳定性不同，其结果的精度也可能大不相同。算法的数值稳定性表明了计算结果受初始数据的误差或计算过程中产生的舍入误差影响的程度。尤其对刚性问题，一定要选择适合刚性问题的算法；否则，计算结果不具可靠性。例如，对小的 ε 值，计算

$$\sin(x+\varepsilon)-\sin x$$

若直接计算，精度就会很差；若利用三角恒等式将上式化为

$$\sin(x+\varepsilon)-\sin x=2\cos\left(x+\frac{\varepsilon}{2}\right)\sin\frac{\varepsilon}{2}$$

来计算，结果就比较精确。

又如，对大的 x 值，计算

$$\sqrt{x+1}-\sqrt{x}$$

由于当 x 很大时，$\sqrt{x+1}$ 与 $\sqrt{x}$ 很接近，直接计算它们的差，结果的精度无法保证。若将上式经恒等变形，可得

$$(\sqrt{x+1}-\sqrt{x})\frac{\sqrt{x+1}+\sqrt{x}}{\sqrt{x+1}+\sqrt{x}}=\frac{1}{\sqrt{x+1}+\sqrt{x}}$$

按这个公式计算，可以得到比较好的结果。

第三节　数 据 结 构

算法实际上是对数学模型中的数据进行操作。不同的数学模型中出现的数据在类型上和数量上都各不相同，数据与数据之间有的是互相独立的，有的则存在某种联系形式和相互关系。例如，计算机管理图书目录问题。当你想借阅一本参考书，但不知道书库中是不是有这种书的时候；或者，当你想找某一方面的参考书，而不知道图书馆有哪些书的时候，你都需要到图书馆去查阅目录卡片。在图书馆有各种名目的卡片，有按书名编排的，有按作者编排的，还有按分类编排的，等等。若利用计算机帮助查阅书目，首先需要将这些目录卡片存入计算机。如何存放？既要考虑查询时间短，又要考虑各种查询方法的需要，还要考虑节省存贮空间。一个最简单的办法，就是建立一张表，每本书的信息只用一张卡片表示，在表中占一行，如表 1－1 所示。此时，计算机操作的对象就是卡片，卡片之间的关系是顺序存放的，计算机对数据的操作是按照某个特定要求（如给定书名或登录号）进行查询，找到表中满足要求的一行信息。在每张卡片上，有五列数据，每一列数据都从某一个侧面反映某本书的属性，就是说，数据与数据之间存在相互联系。我们把这种数据之间的联系形式和相互关系称为数据结构。

表 1－1　　图 书 目 录 表

书　　名	作　　者	登 录 号	分　　类	出 版 年 月

因此，用计算机进行信息处理时，必须考虑如何组织数据，以使查找和存取数据更为方便。要更好地进行程序设计，尤其是进行非数值型的程序设计，就必须研究数据结构。著名计算机科学家 Wirth 曾经提出一个有名的公式：

算法＋数据结构＝程序

就是说，在程序设计时，除了要考虑算法之外，还要考虑选择适当的数据结构。对于同一问题，可以采用不同的数据结构和算法。对不同的数据结构往往有不同的算法，它们的复杂程度也往往不同。选择适当的数据结构常常可以降低算法的复杂程度。例如，稍有程序设计经验的人都知道，将若干个数据排序时，用数组这样的数据结构就会使排序算法简单得多。

在计算机程序设计语言中，数据结构的具体体现是数据类型。数据类型是数据结构及其运算的总称。换句话说，数据类型是各种类型的数据所能取的值和所能作的运算的集合。

自从数据结构理论提出以来，各种程序设计语言都明确规定了数据类型的概念，即每个常量、变量或表达式的值都属于确定的数据类型。虽然在程序执行期间，变量的值在不断地改变，但是，变量所有可能的取值以及在这些值上所允许的操作都在程序中明显或隐含地被规定。

类型概念的显著特性可以概括为：

(1)类型决定了变量或表达式所能取值的集合。

(2)每个值属于且仅属于一个类型。

(3)任何常量、变量或表达式的类型可以从其形式或上下文推断出来，而无须了解运行时所计算出来的具体值。

(4)每种操作要求一定类型的操作数，并且得出一定类型的结果。

(5)一种类型的值及其上所规定的基本操作的性质可以由一组公理阐明。

(6)高级程序语言应用类型信息去防止或表明程序中无意义的结构，又可用于确定在计算机中的数据存贮和处理方法。

基于以上准则，每种高级语言都根据其所应用的范围规定了它所可以使用的类型。一般有如下三种不同范畴的数据类型：

(1)简单类型(基本类型)　不可再分的最基本的数据项。这种类型包括整型、实型、双精度实型、复型、逻辑型及字符型等。

(2)构造类型　由已知的简单类型通过一定的构造方法构造出来的新类型，包括数组、记录、集合等。

(3)指引元类型　用于构造各种形式的动态或递归数据结构，如链表、树、图等。

FORTRAN 语言只提供了上述六种简单类型及包括数组、记录、文件在内的构造类型，没有提供指引元类型。

在程序设计语言中使用数据类型的概念将带来如下好处：

(1)带来了程序的简明性　它明白告诉我们，变量应该取什么样类型的值，操作应该在什么样类型的数据上进行，从而可以防止许多不合法的错误。例如，编译时可以检查出给一个整型变量赋逻辑值的错误。能够进行类型匹配检查，是高级语言较之机器语言的一大进步。

(2)提高了程序的执行效率　当程序实际运行时，每个变量的当前值总是存贮在一个或多个机器存贮单元中。由于类型说明总是出现在程序可执行部分之前，所以编译系统可以决定和预先分配为存贮这些变量的值所需要的相应大小的存贮空间，而不致于等到运行时才去分配，这就提高了程序的执行效率。

第四节　结构化程序设计

在完成了建立模型、确定算法、选定数据结构的任务之后，就可以进入程序编制阶段。事实上，建立模型、确定算法、选定数据结构以及程序编制、调试、测试都是整个程序设计过程的组成部分。人们往往把程序编制称为程序设计，这是不完整的、不确切的。因为要设计一个好的程序必须将上述四个任务统筹考虑，掌握科学的程序设计方法。

一、结构化程序设计的由来

60 年代，一个程序员只要能正确理解程序设计语言的意义，经过一定的实践，掌握程序设计中的一些基本技巧，学会使用一些算法，就可以编制出适应当时要求的程序来。为此，人们称程序设计是一种艺术。程序中使用的技巧越多，运行速度越快，所占用的存贮单元越省，这个程序就越具有“美感”。到了 60 年代末至 70 年代初，出现了大型软件系统，如操作系统、数据库等，这给程序设计带来了新的问题。一个大系统常常需要花费大量的资金和人力，有时需要几千人年的工作量。当许多人分别编制的风格各异的程序连结在一起的时候，常常是可靠性差、错误多、维护和修改也很困难。尤其是维护修改别人编制的程序更是困难，隐藏的错误也不易发现和纠正。例如，有的已经在运行的大型操作系统却隐藏着几百甚至几千个错误。当时，人们称这种现象为“软件危机”。

“危机”震动了软件界，这就促使人们开始研究程序设计中的一些最为基本的问题。例如，程序的基本组成部分是什么？应该用什么样的方法来设计程序？程序设计的主要方法和技术应如何规范化和工程化等等。

1969 年，荷兰学者 Dijkstra 首先提出了结构化程序设计的概念，他强调了从程序结构和风格来研究程序设计。70 年代初，发生了要不要 GOTO 语句的争论，这场争论的实质就在于是

讲究好结构程序还是讲究效率。所谓好结构程序，指程序结构清晰、易于理解、易于验证。好结构程序从效率上看，不一定是最好的程序。但是，它能提高程序的可靠性，便于检查和维护。争论的结果，人们普遍认为，在计算机运行速度大大提高、存贮容量也很大的情况下，程序的可靠性和可维护性已成为第一要求，除了系统的核心程序以及其他一些特殊要求的程序之外，在通常情况下，宁可降低一些效率，也要保证程序的好结构。从此，人们开始总结出一套程序设计的基本原理和方法，并摸索出了一套规律，从而使程序设计不再是一种纯粹技巧性的工作，而逐渐上升为一门科学性的学科，使之成为既是一门科学，又仍保持着艺术的特点。目前，普遍使用和比较成熟的程序设计方法，是结构化程序设计方法。

二、结构化程序设计方法

结构化程序设计方法主要包括以下两方面的内容。

1. 模块化程序设计

模块化程序设计，也称“积木式”程序设计，其基本思想是在进行大的系统设计时，在总体方案阶段，采用自顶向下、逐步求精的方法，把一个大而复杂的问题分解成若干个相对独立、单一功能的模块。这些模块可以独立地被理解、编写、测试、排错及修改。因而使复杂的研制工作得到简化。由于模块的相对独立性也能有效地防止错误在模块之间扩散蔓延，当许多模块连结在一起时，减少了相互之间的干扰和影响，从而提高了系统的可靠性。

模块化程序设计的结果，使整个程序由一些相对小的程序子结构组成；每个子结构，相当于一个过程或子程序，都具有一定的相对独立性，可以分别编制、调试和修改。这样，一方面可以使许多人分头工作，缩小软件研制的周期，另一方面也使整个系统思路清晰，便于发现问题及时纠正。随着总体方案的逐步深入，一个大的模块还可以进一步细分为若干更小的模块。一般地，整个系统的模块呈树状结构，很象一棵倒立的树：“树根”是主控模块，下连第一层子模块，各个子模块下还可以有更下一层子模块，……直至最底层模块。最底层模块称为“树叶”，中间各层子模块称为“树枝”。图 1－1 表示一个三层的模块结构。

采用模块化程序设计的好处是：

(1)模块相对独立　对某一个模块的修改只会对本模块发生影响，而不会影响其他模块，因而不会造成连锁修改。

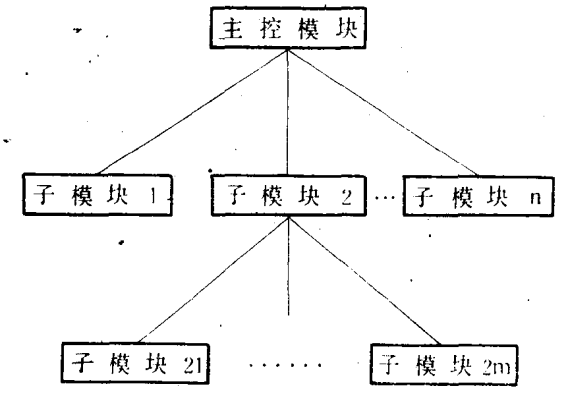

图 1－1　模块化程序结构图

(2)模块本身只有一个入口和一个出口　模块内部的数据结构，只有模块内部提供的操作

才能给以改变，因而保证了模块内部数据结构的安全性和完整性。

(3)各模块除上下级之间有接口外，相互之间无接口。因而，程序结构清晰、关系集中，使得各模块可以独立编制、调试和验证，模块的正确性比较容易保证。

2. 结构化程序设计

这里所说的结构化程序设计是指程序结构只采用几种基本的控制结构，从而使程序具有好结构。

1966 年，Böhm 和 Jacopini 证明了程序设计语言中只要有三种形式的控制结构，就足以表示出各种各样的其他形式的结构。这三种基本控制结构是顺序结构、选择结构和循环结构，每种结构用虚线框起来，分别示于图 1－2、图 1－3 和图 1－4。

(1)顺序结构　顺序地执行语句序列 S1 和 S2，只有当 S1 执行完成之后才执行 S2。

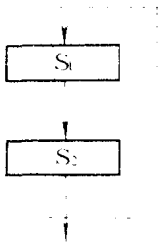

图 1－2　顺序结构

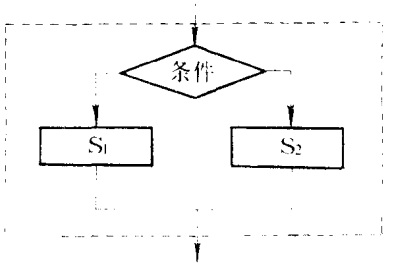

图 1－3　选择结构

(2)选择结构　对条件进行判断，当条件成立或不成立时分别执行语句序列 S1 或 S2，二者择其一。不管执行哪一个语句序列，执行结束后控制都转移到同一出口的地方。

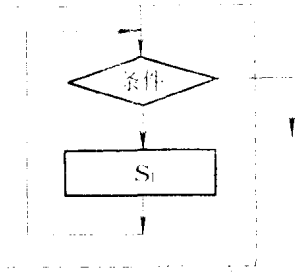

图 1－4　循环结构

(3)循环结构　反复执行语句序列 S1，直到条件不成立时终止循环，控制转移到循环体外。

在上述三种基本结构中，顺序结构和选择结构都是开型结构；当控制转入此结构执行了一串语句之后就从结构中退出，转入下一结构。开型结构不提供将控制转移返回到同一结构的入口处的办法，而循环结构则是闭型结构，它可以将控制转移返回到本结构的入口处。

从图中还可以看出，这三种基本结构有一个共同的特点，就是每种结构严格地只有一个入口和一个出口。当然，S1 或 S2 的内部又可以包含这三种结构。如果组成程序的各个分结构都只存在如此简单的接口关系，那么，就可以相对独立地设计各个分结构，静态地分析控制关系，并验证它们的正确性，而不必象 GOTO 语句那样动态地转来转去。同样，如果某一分结构需要修改，只要接口不变，就不会影响到其他分结构乃至整个程序。因此，具有这样结构的程序是好结构程序。

尽管已经从数学上证明了程序设计语言只要具备上述三种基本控制结构就可以进行结构化程序设计，但为了方便用户的使用，各种程序设计语言还是常常引进其他各种各样的控制结构，如多路选择结构、直到型循环结构、步长型循环结构等。这些结构尽管都可以用上述三种基本结构表示出来，但一般都具有更简洁的形式，直接使用更加方便。

第二章 FORTRAN 语言概述

FORTRAN 语言是世界上最早出现的计算机高级语言，从 1954 年问世到现在，历时近 40 年，仍然广泛流行，仍是科学和工程计算的主要语言。随着时间的推移，FORTRAN 语言也在不断地充实和完善，先后经历了 FORTRAN Ⅱ、FORTRAN Ⅳ（即 FORTRAN66）和 FORTRAN77 几个阶段，FORTRAN77 是美国标准化协会 1978 年公布的，也是迄今为止的最新版本。为了适应科技计算速度的需求，FORTRAN77 仍保留了相当数量的非结构化语句，因此，FORTRAN77 还不是完全的结构化程序设计语言。

为了对 FORTRAN 语言的概貌有一个大致的了解，本章先就 FORTRAN77 的源程序格式、程序运行过程、数据结构、常用语句以及输入/输出功能作一概述，这些内容在后续章节中还要进一步阐述。

第一节 FORTRAN 源程序格式

用 FORTRAN 语言编写的程序称为 FORTRAN 源程序。和其他高级语言源程序一样，FORTRAN 源程序由字符组成。若干个字符组成一个程序行，若干个程序行组成一个程序段（程序块），若干个程序段组成一个完整的程序。一个完整的 FORTRAN 源程序必有且仅有一个主程序段，其他子程序段的数目不限。和其他大多数高级语言不同的是，FORTRAN 语言对源程序的书写格式有严格的要求。

一、FORTRAN 字符集

字符是组成 FORTRAN 程序的基本元素，FORTRAN 允许使用的所有字符构成 FORTRAN 字符集，包括：

(1)字母字符 52 个(大、小写英文字母各 26 个)。

(2)数字字符 10 个(0,1,2,…,9)。

(3)专用字符 13 个。它们是空格、撇号(类似于单引号)，美元符号 $，左、右圆括号，加、减(正、负)号，斜杠，等于号，星号，小数点符号，冒号及逗号。

二、程序行

程序行包括注释行、语句初始行及续行。一般地，每个程序行从第 1 列至 72 列有效，其中，1 至 5 列为语句标号区，第 6 列为续行标志区，7 至 72 列为语句区，各区的内容必须对号入座。

(1)注释行　注释行是程序员对程序所加的注释或说明，对程序运行不发生影响。若第 1 列为字母“C”或“＊”，其他列可出现任意字符，该行即为注释行；若全行空白(即整行未出现任何字符)，也视为注释行。允许注释行出现在程序的任何地方，包括程序段的第一条语句之前和最后一条语句之后，还可以出现在初始行及其第一个续行之间，或者两个续行之间。

(2)初始行　初始行指程序语句行，其标志是在第 6 列(即续行标志区)为数字“0”或空白字符，1 至 5 列可以有语句标号，语句内容则从第 7 列或其后的某列开始书写。

(3)续行　当程序语句在初始行写不下时，可以在续行继续书写，这时，该行的第 6 列有除