高等学校教材

程序设计方法学

（1992年修订本）

胡正国　蔡经球　编著

西北工业大学出版社

高等学校教材

程序设计方法学

(1992修订本)

胡正国　蔡经球　编著

西北工业大学出版社

1997年2月　西安

(陕)新登字 009 号

【内容简介】本书主要介绍程序设计方法学这一新兴学科的主要内容，即结构化程序、程序正确性证明、结构化程序的正确性证明、递归程序及其正确性证明、程序的形式推导技术、程序变换技术．这次修订再版，除对原有内容做了一些修改和调整外，补充了模块化程序设计、面向对象的程序设计方法和快速原型技术的内容．

本书可供大专院校计算机专业学生使用，也可供硕士研究生及从事计算机工作的科技人员参考．

高等学校教材
程序设计方法学
（修订本）
胡正国　蔡经球　编著
责任编辑　王夏林
责任校对　钱伟峰
*

（710072　西安市友谊西路 127 号　电话 8493844）
陕西省新华书店发行
西安向阳印刷厂印装
ISBN 7－5612－0937－1/TP・123(课)
*
开本：850×1168 毫米 1/32　印张：8.625　字数：210 千字
1987 年 2 月第 1 版　1997 年 2 月第 2 版第 2 次印刷
印数：11 001—14 000 册　定价：9.00 元

前 言

程序设计方法学是60年代末、70年代初形成和发展起来的计算机科学领域中的一个新兴的学科．近年来，这一学科发展是比较迅速的，同时也取得了不少令人鼓舞的成果．为了介绍这一学科的一些基本内容，我们在多年进行这方面教学的基础上编写了这本教材．

由于我们编写这本教材的目的是向读者介绍这一学科的一些最基本的内容，因而在讲述时我们尽量避免一些严格的形式化系统．根据我们以往的教学经验，我们相信读者在掌握了这些基本内容以后，可以较顺利地阅读其它有关的专著，以求对这一学科有更深入的了解．

另外，本书中没有采用统一的语言描述程序．我们感到，这样做尽管从表面上看不够统一，但是可以使读者接触较多的控制结构，而且有利于读者阅读有关专著．因而，权衡利弊我们做出了这样的选择．

这本教材初次出版后，受到了许多兄弟院校的欢迎，并且提出了一些宝贵的意见和建议．在此我们向这些同志表示深深的谢意．这次修订再版，我们除了对原有的内容做了一些修改和调整外，补充了模块化程序设计及面向对象的程序设计方法两章．另外，考虑到快速原型技术的发展，在第十章简要地介绍了这一新的软件开发方法的基本思想．本教材可供计算机专业高年级学生及硕士研究生使用，讲授时数约60学时，也可供计算机工作者参考．

本书的第一章到第五章、第七章及第九章由西北工业大学计

算机科学与工程系胡正国编著，第六章、第八章和§10.1由厦门大学计算机科学系蔡经球编著。如果我们的合作能对读者了解和掌握这一新兴学科有所帮助的话，我们将感到十分高兴。西北工业大学计算机科学与工程系陈小群同志参加了第三章和第九章的讨论和部分编写工作。另外，§10.2是根据该系严瑜同志的一篇论文修改而成的。在此向他们表示感谢。

西安电子科技大学陈家正教授在百忙中认真审阅了全书的内容，并提出了宝贵的意见，在此也表示衷心的感谢。

由于时间仓促，加之编者水平有限，不妥之处在所难免，诚恳希望同行的专家及广大读者提出宝贵意见。

编　者

1991年10月于西安

目　录

第一章　程序设计方法学简介

§1.1　程序设计方法学的产生

我们知道，随着计算机硬件技术的不断发展，程序设计方法也在不断地改进和发展。

在计算机发展的早期阶段，由于计算机硬件功能较弱，即运算速度较慢、存储空间较小，因而主要采用机器语言或汇编语言编写程序。这时，程序设计方法的重点是如何尽可能多地使用一些技巧，以节省内存空间，提高运算速度。在本世纪50年代后期和60年代初虽然也相继出现了一些高级语言，例如FORTRAN语言，ALGOL语言等，为提高程序员的算法表达能力，减轻一些繁琐的劳动创造了一定的条件。但是，由于在这一时期计算机主要是应用于科学计算，而且程序的规模一般都比较小，因而从程序设计方法上看并没有发生根本的变化。总的来说，在这一阶段，程序设计被看作是一项技巧性很强的工作，也可以说，采用的是一种手工艺式的设计方法。

本世纪60年代以来，随着硬件技术的迅猛发展和计算机应用领域的急剧扩大，不仅绝大多数计算机都采用高级语言编写程序，而且计算机的一些规模较大的应用软件和系统软件也采用某些高级语言来编写。这时，由于一般要编写的程序的规模都比较大，因而对这些程序来说，运行时间和占用存储空间的大小已经不是编写者要考虑的主要问题，主要问题已经逐渐转化为希望编写出的程序结构清晰、容易阅读、容易修改、容易验证，即希望得到好结构的程序。而要做到这一点，就必须改变传统的程序设

计方法的观念，采用一种新的策略和方法来进行程序设计。这就是所谓的“软件工程”的方法，或者说工程化的设计方法。

程序设计的这一发展过程，可以简单地概括为：

手工艺式的设计方法→工程化的设计方法

程序设计方法的这一发展是方法论上的一个飞跃。为什么会产生这一飞跃呢？这是由于通常所说的“软件危机”引起的。所谓软件危机是指：自 60 年代末到 70 年代初，随着计算机硬件技术的发展和计算机应用范围的不断扩大，需要研制一些大的软件系统，例如，操作系统、程序库等。而一个大的系统的编制周期是较长的，工作量是很大的（常常需要几百到几千人年），并且由于传统的程序设计方法的局限性，设计出的软件系统往往隐藏着许多错误，这就给软件的使用和维护带来很大的困难。一方面，客观上需要研制大量的软件；另一方面，按照原有的方法研制软件周期长，可靠性差，维护困难。这就是“危机”之所在。为了克服这一“危机”，一方面需要对程序设计方法、程序的可靠性等问题进行系统的研究，另一方面，也需要对软件的编制、管理和维护的方法进行研究。这就是程序设计方法学产生的历史背景。

1968 年，E.W.Dijkstra 首先提出“GOTO 语句是有害的”，向传统的程序设计方法提出了挑战，从而引起了人们对程序设计方法讨论的普遍重视。许多著名的计算机科学家都参加了这种讨论。程序设计方法学这一学科也正是在这种广泛而深入的讨论中逐渐产生和形成的。

什么是程序设计方法学呢？简单地说，程序设计方法学是讨论程序的性质以及程序设计的理论和方法的一门学科。

由于程序设计方法学是一门新兴的发展较为迅速的学科，因而它包含的内容是比较丰富的。例如，结构程序设计、数据抽象与模块化程序设计、程序正确性证明、程序变换、程序的形式说明与推导、程序综合技术、面向对象的程序设计方法等。由于篇

幅限制，本书仅介绍其中的一些主要内容。

在程序设计方法学中，结构程序设计占着十分重要的位置。可以说，程序设计方法学是在结构程序设计的基础上逐步发展和完善起来的。因而，了解和掌握结构程序设计的概念以及它所讨论的一些主要问题对了解什么是程序设计方法学是很有帮助的。下面，首先简要介绍这方面的内容。

§1.2 结构程序设计及其讨论的一些主要问题

什么是结构程序设计呢？到目前为止，虽然人们从不同的角度给出了不少的定义，但是还没有一个很严格的，又能为大家普遍接受的定义。

1974年，D.Gries教授将已有的对结构程序设计的不同解释归纳为13种[3]。其中，比较有代表性的有以下几种：

(1) 结构程序设计是避免用GOTO语句的一种程序设计。

(2) 结构程序设计是自顶向下的程序设计。

(3) 结构程序设计是一种组织和编制程序的方法，利用它编制的程序是容易理解和容易修改的。

(4) 程序结构化的一个主要功能是使得正确性的证明容易实现。

(5) 结构程序设计允许在设计过程中的每一步验证其正确性，即自动导致自我说明与自我捍卫的程序风格。

(6) 结构程序设计讨论了如何将任何大规模的和复杂的流程图转换成一种标准的形式，使得它们能够用几种标准的控制结构(通常是顺序、分支和重复)通过重复和嵌套来表示。

这些定义（或解释）从不同的角度反映了结构程序设计所讨论的主要问题。综合这些定义，可以使我们对结构程序设计有一

个概貌性的了解。为了以后讨论方便起见，我们给出下面的定义：

结构程序设计是一种进行程序设计的原则和方法，按照这种原则和方法设计出的程序的特点是：结构清晰，容易阅读，容易修改，容易验证。

按照结构程序设计的要求设计出的程序设计语言称为结构程序设计语言。

利用结构程序设计语言，或者说按照结构程序设计的思想编制出的程序称为结构化程序，或者好结构的程序。

下面，简要介绍结构程序设计所讨论的一些主要问题，在第二章我们还将对有些问题作进一步的介绍。

一、关于 GOTO 语句的问题

前面已经提到，关于程序设计方法的讨论是由避免使用 GOTO 语句引起的。因而，首先对这一问题作一些简单的介绍。

在 60 年代末和 70 年代初，关于 GOTO 语句的争论是比较激烈的。

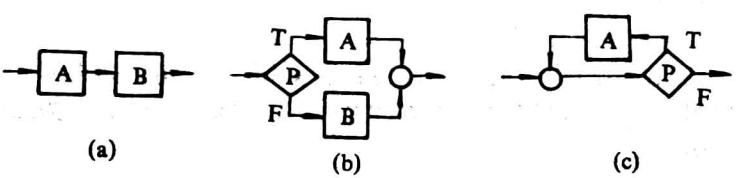
主张从高级语言中去掉 GOTO 语句的人认为，GOTO 语句是对程序结构影响最大的一种有害的语句。他们的主要理由是：GOTO 语句使程序的静态结构与它的动态执行之间有很大的差别。这样使程序难以阅读，难以查错。对一个程序来说，人们最关心的是它运行的正确与否，去掉 GOTO 语句以后，可直接从程序结构上反映出程序运行的过程。这样，不仅使程序的结构清晰，便于阅读，便于查错，而且也有利于程序的正确性证明。

持不同意见的人们认为，GOTO 语句使用起来比较灵活，而且有些情形能提高程序的效率。如果一味强调删去 GOTO 语句，有些情形反而会使程序过于复杂，增加一些不必要的计算

量.

1974 年，D.E.Knuth 对于 GOTO 语句的争论作了全面的公正的评述[2]。他的基本观点是：不加限制地使用 GOTO 语句，特别是使用往回跳的 GOTO 语句，会使程序结构难于理解，这种情形应尽量避免使用 GOTO 语句．另外，为了提高程序的效率，同时又不破坏程序的良好结构，有控制地使用一些 GOTO 语句是有必要的．用他的话来说："有些情形，我主张废除转向语句；另外一些情形，则主张引进转向语向。"

进一步讲，GOTO 语句能不能消除呢？或者说 GOTO 语句这一语言成份能不能从程序设计语言中取消呢？回答是肯定的，1966 年，G.Jacopini 和 C.Bohm 从理论上证明了：任何程序都可以用序列结构、条件结构和循环结构表示出来。具体一些，即任何程序都可以用图 1.1 所示的三种结构表示出来（这一结论将在第二章证明）。



(a) (b) (c)

图 1.1

更进一步，如何从一个具体的程序中消除 GOTO 语句呢？通常可以采用增加辅助变量，或者改变程序执行顺序的方法来消除 GOTO 语句.

例 1.1 考查下面一段程序:

L1: **if** B_1 **then go to** L2 **fi**;

S_1;

```
        if B2 then go to L2 fi;
        S2;
        go to L1;
    L2: S3;
```

需要指出的是，程序中符号 fi 和相应 if 配对使用，表示条件语句的结束。这个记号是由 D.E.Knuth 提出来的，采用它可以省略一些语法括号 begin…end。同时，使程序结构更为清晰。例如，语句

```
if p then begin S1; S2 end
```

可以表示为

```
if p then S1; S2 fi
```

语句

```
if p then begin S1; S2 end
     else begin S3; S4 end
```

可以表示为

```
if p then S1; S2; else S3; S4 fi
```

显然，例 1.1 中的程序段包含有 GOTO 语句。如果引入一个逻辑变量 p，则可以消除 GOTO 语句，建立下面的一段与它等价的程序：

```
p:=true;
while p do
  if B1 then p:=false;
    else S1; if B2 then p:=false;
    else S2 fi fi;
 S3;
```

例 1.2 (查表程序)在一个表中有 m 个不同的数 A[1], A[2], …, A[m]。现在，要编写一段程序，在该表中查找数 x。若找到 x，将该数以及它在表中的位置打印出来；否则，将该数

加到这个表中去．这一工作可以出下面的一段程序来完成:

```
      ......
      for i:=1 to m do
        if A[i]=x then go to 1 fi;
      m:=m+1; A[m]:=x; go to 2;
1:  write(i, x);
2:  ......
```

这段程序包含有两个 GOTO 语句，和上例类似，可以用增加辅助的逻辑变量 p 的方法消除 GOTO 语句，得到下面的等价程序:

```
      ......
      p:=false;
      for i:=1 to m do
        if A[i]=x then p:=true; y:=i fi;
      if p then write(y, x);
         else m:=m+1; A[m]:=x fi;
      ......
```

另外，也可以用改变程序执行顺序的方法消除 GOTO 语句，得到下面的等价程序:

```
      ......
      i:=1;
      while (A[i]≠x)∧(i<m) do i:=i+1;
      if i>m then m:=m+1; A[m]:=x;
          else write(i, x) fi;
      ......
```

上面通过两个例子介绍了消除 GOTO 语句的一些方法。但是，对于比较复杂的程序，要采用类似的方法往往是比较困难的。这时，常采用另一个比较行之有效的方法，即给语言中增加

一些新的控制结构．例如，结构 FORTRAN 语言，结构 COBOL 语言等就是在原有的高级语言的基础上增加了若干个新的控制结构而建立的．另外，一些常用的结构程序设计语言(例如，PASCAL 语言，BLISS 语言)，以及问世不久的 ADA 语言，除了数据结构上作了若干新的扩充以外，也都分别设计了多种控制结构，为避免使用 GOTO 语句创造了条件．关于控制结构的研究，70 年代以来已经取得了不少成果，有关这方面比较详细的内容将在第二章中介绍．在这里需要指出的是，虽然关于结构程序设计的讨论是从废除 GOTO 语句开始的，但是，绝不能认为结构程序设计就是避免 GOTO 语句的程序设计方法．事实上，结构程序设计讨论的是一种新的程序设计的方法和风格，它所关注的焦点是所得到的程序的结构的好坏，而有无 GOTO 语句并不是一个程序结构好坏的标志．这也就是说，限制和避免使用 GOTO 语句是得到结构化程序的一个手段，而不是我们的目的．

二、程序的结构

前面已经谈到，结构程序设计的目标是得到一个好结构的程序．再具体一些，好结构的程序是由一些什么样的语法结构组成的程序呢？一般地说，结构化程序是由下面七种结构组成的．

1. 序列结构

如图 1.2 所示．图中 A，B 可以是一个语句(甚至是空语句)，也可以是这里将要介绍的七种结构中的一种结构．当 A 或 B 是一个结构时，称它为这一序列结构的子结构．对下面几种结构亦有完全类似的情况．

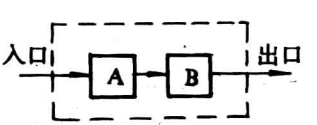


图 1.2

2. 选择结构

有以下三种：最常用的两种选择结构如图 1.3 所示(它们通常也称为条件结构)．更一般的选择结构如图 1.4 所示．

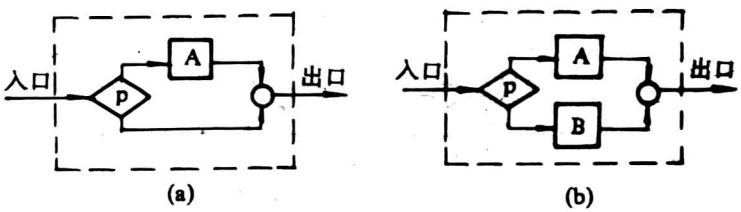


图 1.3

显然，图 1.3(a)的结构是图 1.3(b)的特殊情况，而且图 1.4 所示的结构可以通过嵌套使用图 1.3 的结构表示出来，但是为了更灵活地编写程序，我们仍然将它们看作不同的结构．

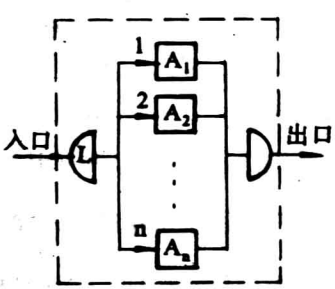


图 1.4

在以上四种结构(序列及三种选择结构)中，都没有出现将控制转移到本结构入口的情形，这样的结构通常称为开型结构．

3. 循环结构

有以下三种(图 1.5)：这三种循环通常依次称为 WHILE 循环、REPEAT 循环和 N+1／2 循环．和选择结构类似，虽然前两种循环可以看作是第三种循环的特殊情形，但为了构造程序方便起见，仍然将它们看作不同的结构．