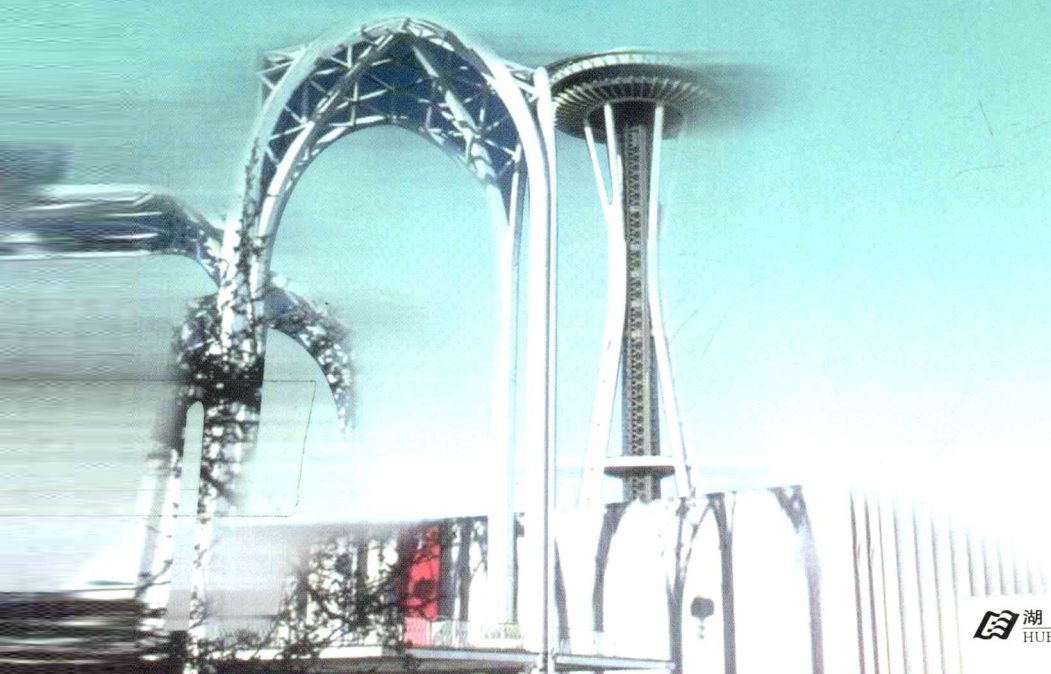
普通高中课程标准实验教科书

数学 3 （必修）

SHUXUE

教师教学用书

湖北教育出版社数学教材编写组 编著



湖北教育出版社
HUBEI EDUCATION PRESS

说　明

为了配合湖北教育出版社出版的《高级中学课程标准实验教科书·数学》的教学实践，我们编写了这套教师用书.

编写教师用书的目的在于，为教师选取素材提供资源，为设计教学提供参考，也为教师处理教学问题提供服务，帮助老师们在充分考虑数学学科特点和高中生心理特点的前提下，运用多种教学方法和手段，实现课程目标.

编写教师用书的目的也在于，与老师们沟通，呈现我们将课程标准转化为教材的心路历程；交流编写意图，特别是在贯彻基本理念、处理某些矛盾时的所思所为，从而对教科书的指导思想和主要特点形成共识，促使教师创造性地使用教材.

编写教师用书的目的还在于，以教科书为载体，从教学的基本问题出发，和老师们一起，共同领会课程标准的基本精神，立足于以人为本，发展和完善人的高度，构建现代理念下的课堂教学.

本套教师用书一般以教科书的章为单元编写，每章由教育价值、教学目标、教材结构、课时分配、内容分析、相关资源、评价建议和习题解答八部分构成.

教育价值：是课程目标在本章的具体化，也是课程设计中确定本章为教学内容的理由.

教学目标：是本章要达到的基本目标，它比课程标准中《内容与要求》要具体些.

教材结构：主要介绍三个方面：知识如何定位，教材怎样展开，有何特点.

课时分配：对每一小节所需的教学课时数作了大致的估计.

内容分析：一般按章头语、各大节逐次展开. 每大节包括三个项目：内容概述及基本要求、重难点分析和教学建议. 其中教学建议是主体，阐述教学中应该强调什么，注意什么，例题的功能及其处理. 除此之外，还涉及到旁批、交流话题、信息技术链接及教科书中课件符号标识处的教学思考等.

相关资源：在于展示本章内容的知识背景，为教学提供素材，包括重要结论的推理、证明与拓展.

评价建议：回答评价什么，如何评价等问题，并提供必要的参考案例.

习题解答：不仅包括练习、习题、复习题等基本题型的参考答案，还就《阅读与讨论》中的讨论题、《思考与实践》中的问题给出了可供参考的解决方案.

我们希望通过上述栏目的设置，既有助于解决教学设计、教学实施中的主要问题，满足教学的基本需要，又能拓展教师的视野，提升数学教学的境界.

诚然，有些想法虽然很好，却是我们力所不及的. 比如评价建议，又比如教学目标中的情感目标，如何落实和实践还有待于我们共同去研究和探索.

我们的课程改革，从理念、内容到实施，与过去相比都有较大的变化. 要实现课程改革的目标，教师是关键. 教师不仅是课程的实施者，而且也是课程研究、建设和资源开发的重要力量. 我们殷切地希望各位老师能为这套教师用书建言，更为教科书的完善献力，使它们更加有利于教师创造性地进行教学，更加有利于学生主动地学习和发展.

本套教师用书由湖北教育出版社数学教材编写组编写，主编齐民友，副主编裴光亚，徐学文，郭熙汉. 本册主要编者是陈应保，费太生.

目　录

第1章　算法初步

一、教育价值

算法是计算科学的基础，运用算法求得问题的数值结果是数学的主要目标之一. 在现代科学技术，特别是计算机技术飞速发展的今天，作为数学及其应用的重要组成部分，算法在科学技术、社会发展中的作用已被越来越多的人们所认识.

中学生了解一些算法的思想，学习一些初步的算法知识，可以发展有条理的思考与表达能力、提高逻辑思维能力，可以激发对信息科学、计算机科学、计算科学的兴趣，可以培养学生大胆探索、认真求解的钻研精神，这对于学生的全面发展和今后的大学学习都是非常有益的.

二、教学目标

1. 知识与能力

(1)要求学生通过例子体会算法的含义和学习算法的重要性.

(2)要求学生熟悉流程图的符号，能用流程图表示简单的算法.

(3)要求学生了解 BASIC 语言的基本语法，能读懂简单的 BASIC 程序，能用 BASIC 语言编写简单的程序. 有条件的地方要求学生能完成用 BASIC 语言编程上机求解问题的全过程.

(4)要求学生掌握课本中所给出的算法，并利用这些算法去解决相应的实际问题.

(5)要求学生尽可能体会教材中所给出的算法的思想，并将这种思想融会贯穿于今后的数学学习中.

2. 过程与方法

(1)由解一元二次方程的计算步骤出发，指出学习算法的重要性，激发学生的兴趣.

(2)从一个实际问题和一个数学问题出发，引出算法的概念，给出算法的特征.

(3)通过例子讲述如何用日常的自然语言来描述算法步骤.

(4)举例说明同一个问题可以有多种算法，并对算法进行比较，指出应尽可能寻求计算次数少的简便算法. 课本中给出计算多项式值的秦九韶法就是减少计算次数的典型，它体现了中国人在数学方面的聪明才智.

(5)本章 1.2 节首先指出用自然语言描述算法的缺陷，由此引入表示算法的重要工具流程图. 接着介绍了流程图中采用的记号，并由浅入深逐步介绍了顺序结构、分支结构、循环结构. 以上的阐述始终结合例子进行，始终贯穿算法的思想.

(6)通过介绍一种最简单的计算机语言：BASIC 语言. 使学生掌握一种在计算机上实现算法的工具，能将自己设计的算法付诸实现.

(7)通过一些典型算法的例子让学生了解一些常用的算法，从中体会算法的思想算法都是在分析问题的基础上给出的，每个算法都给出了程序. 为了清晰地表示算法，许多算法都给出了流程图，为了鼓励学生思考，许多程序给出了旁批.

3. 情感、态度与价值观

(1)自编程序,在计算机上一遍遍地检查、修改程序,最后成功地实现算法,得到正确的结果,这对中学生来说,是一件极有诱惑力的活动.它能极大地激发学生对学习算法、学习数学、学习计算机科学的兴趣,并将算法的钻研融入今后的数学学习中,有利于培养学生刻苦的钻研精神和严格认真的科学态度.

(2)一个复杂的问题,从分析问题、确定算法,到编写程序、上机调试得到结果,最后整理结果,往往需要同学间互相讨论,这种实践活动对培养学生的合作意识是十分有利的.

(3)了解一些常用算法、体会算法的思想、编写程序,这些对培养学生有条理的思考和表达能力,发展学生的逻辑思维能力,提高学生的数学素质都是非常重要的,并能在学生今后的学习生活中产生明显的效果.

(4)通过对古代算法典型例子的了解,学生能体会中国人对世界数学发展所作出的贡献,从而增强民族自豪感,增强学好数学的自信心.

(5)将实际问题化为数学模型,再设计适当的算法加以解决,这一过程是学生将所学知识融会贯通,灵活应用的过程,它能使学生提高对数学应用价值的认识,有助于提高学生将所学知识用于实际的意识和能力,有助于学生进一步树立唯物主义的世界观.

三、教材结构

设计本章内容时,力图使学生通过算法实例的学习和上机实践初步体会算法思想,了解一些常用的算法,能读懂和书写简单的程序,培养有条理思考的习惯.

章头语首先突出算法的重要性,接着指出从数学发展史来看,算法的思想是贯穿始终的,它已成为数学及其应用的重要组成部分.

全章共分 4 节.

第一节由两个例子引出算法的概念,将算法理解为"为解决一类特定问题而采取的确定的、有限的步骤".接着指出算法的三个特征:确定性、有穷性和有效性.

第二节介绍描述算法的流程图.首先介绍了流程图中使用的记号,然后给出了一些例子,伴随着由浅入深给出的例子,介绍了顺序结构、分支结构和循环结构.

第三节结合例子简明扼要地介绍了一种简单的程序设计语言——BASIC 语言.

第四节介绍常用算法,用例子阐述了求最大公因数、判断素数、二分法求方程的近似解、求一次方程组的解、穷举法、顺序查找法、求一组数中的最大数、排序、累加求和、累乘求积等.

四、课时分配

本章教学时间约需 12 课时,具体分配如下(仅供参考):

1.1 算法的特征	约 2 课时
1.2 描述算法的流程图	约 2 课时
1.3 用 BASIC 语言实现算法	约 4 课时
1.4 算法举例	约 4 课时

五、内容分析

本章的重点是算法,首先是通过例子理解算法思想,然后是学习如何针对具体问题来设计算法,如何用流程图来表示算法,如何编写程序上机实现算法.

本章的难点是如何依据算法正确地画出流程图和编写程序. 关键是先针对问题分析问题是属于哪一类的,有没有什么特殊性,用什么样的算法解决,程序的大体结构如何.

在组织本章教学时应注意将重点放在算法上,切不可将本章当做计算机语言课来讲授. 即使在讲授语法规则时,也应尽可能融入算法思想. 有条件的学校应提倡学生多上机,开始可运行书中的程序,稍后应鼓励学生自编程序上机. 让学生经历一次次的失败,一次次分析失败的原因,一次次修改程序,直至成功的过程. 这个过程中老师加以适当的指点和引导,可激发学生自主探求的兴趣,培养学生坚忍不拔的毅力. 对课本中的例题和习题,教师可灵活变换问题要求,引导学生多思考. 鼓励学生对已有的程序进行修改,或另辟蹊径给出新的算法或程序. 这对培养学生的创新意识是非常有利的. 在教学中应注重培养学生主动与人合作交流的意识,鼓励学生针对算法设计进行讨论和争论,共同体验协作成功的乐趣. 现将本章内容作如下分析.

章头语

首先提出一个问题:看似万能的计算机为什么这么神奇呢? 这样很自然地呈现了算法的重要性;接着指出从数学发展史来看,算法的思想是贯穿始终的,它已成为数学及其应用的重要组成部分. 这样的章头语就是想激起学生了解算法的兴趣,抱着强烈的求知欲来学习后面的内容.

1.1 算法的特征

1. 内容概述及基本要求

本节由两个例子引出算法的概念. 第一个例子是查字典,属处理日常事务的例子. 第二个例子是解一个具体的一元一次方程,属解决数学问题的例子. 从这两个例子可看出,解决任何问题都应有一定的方法和步骤,从中引出算法的概念:我们可将算法理解为"为解决一类特定问题而采取的确定的、有限的步骤". 接着指出算法的三个特征:确定性、有穷性和有效性.

课程中讨论的算法主要指数学方面的算法. 为了在计算机上以实现算法,有必要学会详细描述算法的步骤. 为此课本中给出了三个例子. 第一个例子是求 50 个数的和. 第二个例子是将两个输入到计算机中的数按从大到小的顺序输出,给出了两种算法,并将两种算法进行了比较. 第三个例子是计算多项式在某点的值,也给出了两种算法,并将两种算法中作乘法的次数进行了比较. 指出其中一种由我国古代数学家秦九韶最先使用的方法能显著地节省运算次数.

学生通过本节的学习,应能理解算法的概念,认识算法的特征. 通过本节的学习,学生能对较简单的问题设计算法,并能用自然语言正确地描述算法.

2. 重、难点分析

本节的重点是对算法的概念及特征的理解. 教材中的处理是先由例子引出算法概念,然后列出它的几点特征,在讲述每一特征时穿插例子以帮助学生理解.

本节的难点是用自然语言描述算法. 教材中不是空泛地指出如何描述,而是运用例子作示范,让学生去体会、模仿.

3. 教学建议

限于篇幅,教材中例子不多,为了帮助学生理解算法的概念和特征,教师可补充例子,但补充的例子必须针对性强、准确,避免误导. 在描述算法步骤的部分共举了三个例子. 例 1 中对求 50 个数和的算法用自然语言详细描述了算法步骤. 在这个例子中首次提到了变量的概念,这样可使对算法的描述更简洁明确.

例 1 中使用的算法为累加法，其中设置了一个求和变量 S，其初值为 0，以后每次加入一个数，它的值随之改变. 加入 50 个数后，S 存放的就是 50 个数的和了. 这里的第二步至第六步是重复的操作，每次操作都是将一个数读入变量 x，然后加到 S 中. 为了控制操作次数，设置了一个计数变量 n，其初值为 1，每做完一次上述操作，其值在原有基础上加 1. 做下次操作前都要先检查 n 是否超过 50，未超过 50 时重复如上操作，否则将 S 中的值打印出来，实际上这是第 2 节中要介绍的计数型循环结构.

给出例 2 的目的是想说明针对同一问题可以有不同的算法. 例中给出的两种算法都很容易理解，教材中将它们进行了比较，说第一种算法比第二种简便，且占内存较少. 应当指出，这个结论是对问题本身的要求而言的，如果进一步要求将较大的数加 1，则在第二种算法中只须再将 A 加 1 即可，这就显示了第二种算法的优越性了. 所以我们这里避免列出几条衡量算法“好坏”的标准，只是让学生感觉到针对同一问题的算法可以有多种，对它们的难易程度、所占内存大小可以比较.

例 3 是计算多项式在某点的值. 这里也给出了两种算法，并对两种算法的乘法次数进行了比较，指出其中的第二种算法（秦九韶法）能显著地节省运算次数. 引入这个例子是让学生对算法中的运算次数有所认识，并对我国古代数学家在算法方面的贡献有所了解.

用自然语言描述算法本身就是有难度的，教师可再选一些教材外的例子进行算法描述，目的是让学生多认识一些算法，培养他们的逻辑思维能力. 举例时不宜太复杂，尽量选取不同类型的例子，也可对同一问题让不同学生设计并描述算法.

1.2 描述算法的流程图

1. 内容概述及基本要求

本节首先指出上节中用自然语言描述算法的弊端，自然引出一种能清晰地表示算法的工具——流程图.

首先给出一个简单的顺序结构的流程图，对图中所用的记号逐一解释. 随后在例 1～例 3 中举出出现了分支结构和循环结构的流程图. 这个过程中对三种基本结构作了简要介绍. 从例 4 起介绍出现嵌套情形的流程图：例 4 中是分支套有分支；例 5、例 6 是循环中套有分支；例 7 是循环中套有循环. 这样就能使学生对流程图有较全面的了解.

学生通过本节的学习，应能了解流程图的概念，熟悉流程图的符号，读懂流程图，并能初步学会用流程图表示三种基本结构和简单算法.

2. 重难点分析

本节的重点是在对算法进行分析的基础上画流程图，难点是流程图中各框相对位置的确定. 学生一般知道应该有些什么样的框，但各框关系如何，如何将它们组织成流程图，这些都需要教师的指导.

3. 教学建议

教师应在讲解每个例子的流程图前先引导学生进行分析：针对问题提出的任务，大致需用什么样的步骤完成；流程图采用什么样的结构；应该用一些什么变量，它们各有什么作用；需要什么样的输入、输出等. 只有脑子清楚，画出的流程图才能脉络清晰；如果脑子里像一锅浆糊，画出的流程图必然乱七八糟. 下面按例子的先后次序分述之.

引入流程图记号的例子是求某学生语文、数学、英语三科成绩的平均值，将三科成绩分别输入到变量 A，B，C 中，然后将 A，B，C 值相加除以 3 放入变量 $AVER$，再将 $AVER$ 打印出来.

这个框图中出现了输入、输出框，处理框和起止框，是一种没有分支、没有循环的结构——顺序结构，其方向自上而下，是一种最简单的情形. 教师应指出的是，一般程序都应有输入、输出，其作用是实现人与机的数据交换，我们后面要讲到 BASIC 语言中实现输入的三种语句——赋值语句、读数与置数语句(在流程图中它们可对应地画成矩形框)和键盘输入语句(在流程图中对应地画成平行四边形框)，还要讲到实现输出的打印语句.

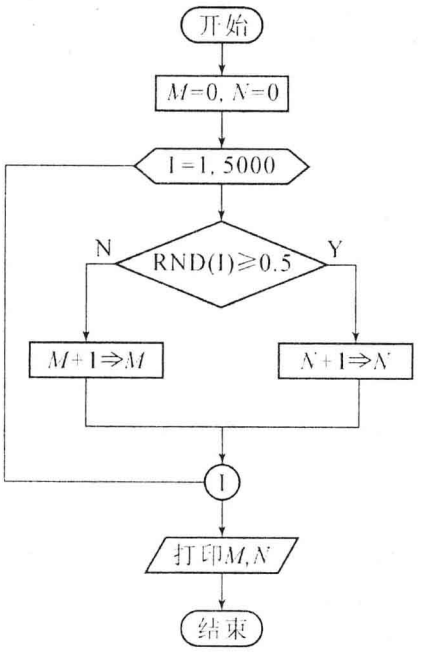
第二个例子是要求画出求一元二次方程根的流程图. 这里流程图中出现了分支，从而引出了分支结构的概念. 整个流程是：输入方程二次项、一次项、常数项的数 a,b,c 后，计算判别式 b^2-4ac 的值放入变量 d，检查 d 的值是否大于或等于 0，是则按求根公式求出两个根分别放入 x_1,x_2，否则打印一串方程无实根的信息. 这个流程图表示了用计算机求解数学问题的一种典型算法. 应当指出，我们这里画的输入框为平行四边形框，表示 a、b、c 的值由键盘输入，这样，据此编出的程序，每次运行可输入不同的系数，因而具有通用性，可以解任何一个二元二次方程.

例 2 中要求画出求十个家庭年均收入的流程图，因为家庭数较多，因此我们不像本节引例中那样求和，而是用累加法求和. 累加法已在 1.1 节例 1 中用自然语言描述过，这里用流程图描述算法显然更清楚. 在这个例子中先给出流程图 1-3，并据此讲解了循环的执行过程，这个过程能作为循环过程的一般代表，应详细讲解. 这个例子的算法因为是用计数型循环完成的，因此也可将流程图用图 1-4 表示. 图 1-4 比图 1-3 简单，学生更易掌握.

选用例 3 的目的是想描述使用“当循环”的算法. 算法中采用了“终止标志”的技巧. 虽然与例 2 一样，此例仍然是求若干家庭的年均收入，但由于不知家庭个数，采用计数型循环就不合适了，从此例可看出，当循环有时比计数型循环更灵活，教师可适当补充例子.

例 4 是分支中套有分支的例子. 讲这种流程图时，教师应分析每层条件满足(或不满足)的情况下应做的工作，否则会产生逻辑上的混乱.

例 5 的算法模拟投掷硬币，统计正、反面出现的次数. 它是循环结构中有分支的情形. 因为是一个计数型循环，也可使用循环框(六边形框)将流程图画为



例 6 也是循环结构中有分支的情况. 这里的循环仍是计数型的. 应注意的是，这里的分支中的条件 $A+B=C$ 不满足时，下面没有任何处理框，表示不执行任何操作.

例 7 要求画“九九乘法表”的流程图，这是一个计数型循环中套有另一个计数型循环，教师可引导学生详细探讨循环的执行过程以加深理解. 应该说两个计数型循环的嵌套是比较简单的情形，计数型循环与当循环也可彼此嵌套.

在正文阐述主要内容的同时，本节中的旁批引导学生做进一步的思考. 例如例 2 的旁批引导学生思考流程图中各框的逻辑关系；例 3 的旁批引导学生思考终止标志如何选取；例 5 的旁批引导学生如何对流程图进行优化；例 6 的旁批可引导学生针对问题条件的变化，如何在原有的流程图上进行修改；例 7 的流程图引导学生计算循环次数等等. 这些旁批都给学习内容的展开留下很大的空间，教师可据此引导学有余力的学生参考有关书籍，引导他们进一步深入钻研有关知识.

1.3 用 BASIC 语言表示算法

学习算法不能光“纸上谈兵”，在计算机上将算法付诸实现，能使学生产生成就感，激发学习兴趣，但要做到这一点，必须让学生了解一种计算机语言. 要讲授语言规则，但又不能“喧宾夺主”，这就是选择最简单的 BASIC 语言的原因.

1. 内容概述及基本要求

本节分两小节介绍 BASIC 语言的语法

1.3.1 节介绍 BASIC 语言的特点，然后介绍 BASIC 中变量、常量、函数的概念及用法，最后介绍 BASIC 的各种表达式：算术表达式、关系表达式、逻辑表达式的组成、书写和使用等.

1.3.2 节介绍 BASIC 语言中各种语句：输入输出语句、结束语句、条件语句、计数型循环语句、当循环语句、数组定义语句等的语法和书写、使用的说明、介绍结合例子，由易到难进行.

学完本节后学生应能做到：

①能写出合法的 BASIC 变量名，能识别科学计数法写出的常数；

②能写出常用函数的 BASIC 形式，能正确使用 BASIC 的标准函数；

③能正确书写 BASIC 的各种表达式并求值；

④能写出合法的 BASIC 语句；

⑤能将 BASIC 语句组成正确的 BASIC 程序；

⑥能根据流程图写程序，或画出程序的流程图.

2. 重点、难点分析

本节的重点是变量、函数表达式的正确书写和使用，各种 BASIC 语句的语法规则及正确使用. 只有正确的掌握 BASIC 语言，才能正确地书写 BASIC 语句，才能正确地将算法转化为程序.

本节的难点一是 BASIC 表达式的准确书写；二是某些函数的正确使用；三是各种语句的正确书写和组织. 为此教材中举了不少例子，并配备了相当数目的练习、习题，教师也可自拟题目让学生进一步熟悉语法规则.

3. 教学建议

1. 程序设计语言的语法规则是十分严格的，不合语法的程序上机时根本通不过，学习严格的语法规则有助于培养学生严谨的作风，教师务必向学生强调这一点.

2. 取整函数和随机函数的使用十分广泛，它们的正确使用是本节难点之一. 教师应结合

书中例子详细讲解.

3. 算术表达式的正确书写是很重要的,教师可让学生多做练习,将容易出错的地方反复强调.

4. 逻辑表达式的正确书写及其求值也是本节的重、难点之一. 教师可针对具体例子讲清逻辑关系以帮助学生掌握.

5. 每个语句的语法和使用说明都应在理解的基础上记牢,这样用起来才得心应手.

6. 讲解三种输入语句时,可以让学生对同一问题列出三种输入语句并比较.

7. 对条件转移语句和循环语句的讲授应着重于功能部分,尤其是循环语句的执行过程应结合例子详讲.

8. 教材中对当循环语句没有举例. 教师可结合具体情况举例,以帮助学生理解. 要注意引导学生比较两种循环语句的异同. 但这节中对学生的要求不宜过高,学生将在做练习、习题的过程中,在对第 4 节算法的学习中逐渐学会对各种语句的灵活使用.

1.4 算法举例

1. 内容概述及基本要求

1.4 节算法举例分两小节,第 1 小节介绍初等数学中的一些算法;第二节讲解一些常用算法. 算法都是结合例子给出的,一般先简要介绍算法思想和变量的含义,然后给出流程图,最后给出程序. 有些程序给出了旁批,力图引导学生思考问题,更深入地理解算法思想.

学习本章后要求学生能正确领会各种算法的思想;能画出描述常用算法的流程图;能写出较简单的常用算法程序;能较自如地运用上节所学的各种语句编写程序.

2. 重难点分析

各种算法的思想及实现是本节的重点;难点是算法的思想付诸实现,即程序的编写,初学者编写的程序往往会有漏洞. 解决之道一是要做到心中有数,即对算法步骤做到心中有数;二是要多看、多写、多练习,做到"熟能生巧".

3. 教学建议

教材中每种算法都由具体例子给出,它的适用范围一般并不局限于例子本身. 因此,教师应着重引导学生理解算法的思想,只有这样学生才能举一反三,灵活运用这里的算法解决其它问题.

第 1.4.1 节先介绍求两个正整数最大公因数的算法——辗转相除法. 算法的步骤用流程图表示得非常清楚,教师可引导学生进一步思考一些问题,比如将循环用 IF-THEN 和 GOTO 语句配合完成,应该如何修改流程图和程序,为什么有两个完全相同的语句(30 语句和 70 语句)能不能去掉一个等.

本小节介绍的第二个算法是判断一个正整数是不是素数. 这里没有给出流程图,直接给出了 BASIC 程序. 此处教师可引导学生分析程序的结构,也可向学生提一些问题,比如 30 语句行中的 GOTO 语句能否去掉,如果要挑出某个范围内的所有素数,应该怎样修改程序等.

本小节的第 3 个算法是用二分法求方程 $f(x)=0$ 的近似解. 这是一个很有用的算法,它的结构为循环内套有分支. 算法的思想较容易理解. 教师带领学生上机时,可将 20 语句——自定义函数语句中的函数加以修改,针对不同函数从键盘上输入相应的区间,使学生体会实际应用算法的乐趣. 应指出的是,这里总假设 $f(a)\cdot f(b)<0$,即 $f(x)=0$ 在 $[a,b]$ 一定有解. 可以让学生考虑如何将 $f(a)\cdot f(b)$ 是否小于 0 作为判断条件加在程序中,这时如何修改流程图和

程序.

本小节的第4个算法是求一次方程组的解.这里介绍的算法是顺序消元法.教材中首先较详细的介绍了解三元一次方程组的消元过程,然后列出了程序.这里仍然应将算法的讲解作为重点.将消元法的思想转化为程序,是这里的难点.因程序较长,学生理解可能会有困难,教师可将程序分为输入、第一次消元、第二次消元、回代、打印结果几个部分来讲解.应该指出这里的顺序消元法的实现是有条件的,这个条件就是 A 的各阶顺序主子式不为0(反映到程序中的40、100、130语句),当条件不满足时,方程组无法用此法求解,这时计算机打印出一串 *.

1.4.2节介绍常用算法中的几种:穷举法、顺序查找法、分类统计法、求最大(最小)法、排序法、累加(乘)法.这些算法都很重要,是本章学习的重点,分述如下:

例1是解古代数学家给出的百鸡问题,它归结为求一个不定方程组的非负整数解.教材中已给出了流程图和程序,教师可引导学生分析程序的结构,也可举其它同样采用穷举法的问题作例子,以加深学生的理解、记忆.例2表明的是顺序查找的算法,程序中首次使用了字符型数组 $A\$$、$B\$$,分别表示某班学生的学号和姓名.数组是一种非常重要的数据类型.此例中的循环是用下标变量(即数组元素)的下标为循环变量的,这种处理在应用数组的程序中带有普遍性.应当指出,因顺序查找的思想简单,故编程容易,但它不是一种效率高的方法.例3用一个社区各年龄段人口统计例子介绍分类统计的算法,算法的关键部分用一个当循环完成,这里程序的两个旁批可帮助学生理解算法和程序.应当指出,这里给出的分类统计算法适合于某些段的间隔相等的情形,当各段间隔无规律可循时,可用多个条件语句并列实施分类统计.例4要求100个数中的最大者,这也是许多情况下都会出现的普遍问题,解决这类问题的方法可称之为"打擂台",教材中先介绍了算法的思想,然后给出了流程图和程序.这个算法的思想较易理解,较难的是如何灵活地运用,当问题的要求有改变时应当怎样修改程序?程序的旁批就是针对问题要求改变时给出的,与此类似,教师还可提出其它问题:如何求若干个数中绝对值最大者?如果已知数的个数事先不知道,怎么办?如果将数改为字符串,怎么处理?

排序也是我们日常生活、工作中常遇到的问题,用计算机解决这类问题有许多方法.教材中给出的比较排序法是其中比较容易理解的一种.教材中先介绍算法的思想,然后给出了流程图和程序.算法的思想和程序结构是这里的重点,难点是学生对算法的理解.教师在讲授时可先用十个具体数来排序,让学生清楚地看到排序过程,此外两重循环中初值和终值的选取这些细节也要讲清楚.除旁批提出的问题外,教师还可引导学生思考下面一些问题:如果按数的绝对值大小排序,应如何修改程序?如果是若干字符串排序,又应如何修改程序?如果每次参与排序的数的个数不同,应将程序做怎样的修改?

第1.4.2节最后介绍累加(或累乘)的算法.数学中经常要用到累加或累乘问题,比如求 $1+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{10!}$,其中既有累乘(求阶乘)也有累加问题.因为累加问题和累乘问题有相同的规律,所以把它们放在一起介绍.例6是解决累加问题的例子,因为不知道参与求和的项数,所以采用当循环.这里的关键是找出WHILE语句中的条件;二是找出和式中各项的通项公式或找出相邻项间的关系,例6中通项为 I^3(i=1,2,…),而和式 $1+\frac{1}{2!}+\frac{1}{3!}+\cdots+\frac{1}{10!}$ 中,若相邻两项中的前一项用 P 表示,后一项则为 $P\times\frac{1}{I}$($i=2,3,\cdots,10$).当求和的项数已知时,用计数型循环会更简单.累乘求积时的程序与累加求和的程序基本相同,注意此时给放乘积的变量赋数值时一定不能赋为0.

六、相关资源

在这里我们介绍三方面的知识，即结构化程序设计与N－S流程图，数值算法及误差，BASIC语法补充.

1. 结构化程序设计与N－S流程图

我们已经见过了不少程序.自然就会提出一个问题：什么样的程序才是一个好的程序呢？在计算机发展的初期，由于硬件价格昂贵，内存容量和运行速度都受很大的限制.人们在编制程序时，往往把占内存小，运行次数少作为衡量程序质量的主要标准.不少程序设计者为了节省内存和提高运行速度，人为地设置一些“技巧”，给阅读程序的人带来了困难.

随着计算机硬件价格的降低，已没有必要为节省一点内存和运行时间而采用难懂的“技巧”，况且，一个难懂的程序作为产品推广在社会上将产生巨大的人力浪费，最终是得不偿失的.目前计算机应用已普及到社会的各个方面，对软件的需求也飞速增长，降低软件的生产成本，将程序设计的手工方式改变为大生产的方式，将软件的编制视为一项“工程”，按严格的规范和一定的步骤来操作是时代的要求.结构化程序设计的思想由此诞生.

结构化程序设计把“具有良好的结构，容易阅读和理解作为衡量程序好坏的首要标准（当然，运行结果必须正确），其次才考虑运行时间和所占内存.为了提高程序的易读性，荷兰学者迪克特拉(Dijkstra)等提出了“结构化程序设计方法”，这种方法规定了几种具有良好特性的基本结构，这种结构的特点是①只有一个入口；②只有一个出口；③没有永远执行不到的语句；④没有永远执行不完的循环.

结构化程序设计程序开发的方法是“自顶向下，逐步细化和模块化，直到分解为上述基本结构为止.

严格说来，我们前面所说的顺序结构、分支结构、循环结构并不完全满足上述“具有良好特性的基本结构”的要求，第②条教材中有的程序就不满足.例如，判断一个正整数是不是素数的程序中原30语句行为

```
30  IF A/I=INT(A/I)  THEN PRINT A;"is not a prime number.";GOTO 60
```

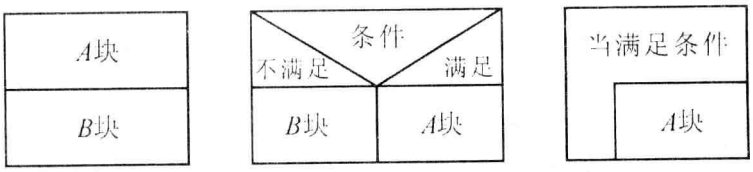
最后的GOTO 30的作用是确定某数不是素数后打印出这个信息，然后转到循环外的60语句（结束语句）.这样程序本身虽无错，但这个循环结构有了“两个出口”，不符合“具有良好结构”的要求.改善的办法是用一个标志变量F，其取值为0，当数A不是素数时将F赋值为1，并立即转到循环外，最后由F的值是0还是1判断A是不是素数.程序可修改如下：

```
10  INPUT "Enter an integer(>2)";A
15  F=0
20  FOR I=2 TO A-1
30  IF A/I=INT(A/I) THEN F=1
40  NEXT I
50  IF F=1 THEN PRINT A;"is not a prime number."
    ELSE PRINT A;"is a prime number."
60  END
```

这个程序中的循环只有一个入口，一个出口，符合结构化程序结构的要求.可见，BASIC语言编写的程序能符合结构化程序设计的要求.

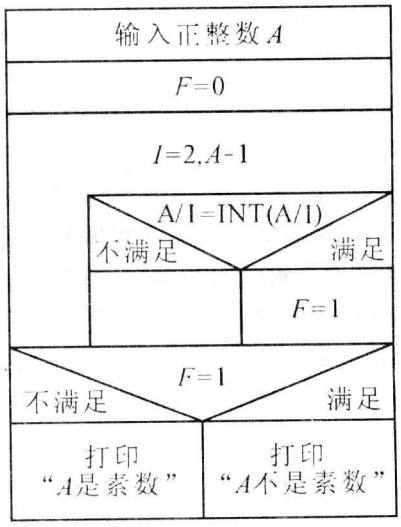
由于结构化程序设计的上述特点，其程序流程图中可取消流线，称之为“N－S图”，三种

基本结构图示为



顺序结构　　分支结构　　循环结构

比如上面程序流程图为



2. 数值算法及误差

在数学问题中，有很多情况下要求得到数值结果，我们称这类数学问题为数值问题. 讨论一个方程有没有解是一个数学问题，但不能称为数值问题. 在方程有解的情况下求出解，这就是数值问题了.

可以说数值问题是指：从一组原始数据出发，求得一组结果数据，使这两种数据间满足事先确定的某种关系.

解数值问题的算法称为数值算法. 我们首先讨论一下误差的来源.

用数学工具来解决实际问题，一般要先建立数学模型(这个过程简称数学建模)，在建立模型的过程中要对问题作一些简化，舍弃一些次要因素. 因此，数学模型本身就包含着误差，我们称之为“模型误差”. 例如用公式

$$s=\frac{1}{2}gt^2$$

可以计算自由落体运动中物体在 t 秒内下落的距离，这就是自由落体运动的数学模型，它是忽略空气阻力得到的. 因此，用这个公式求得物体的下落距离，与实际下落的距离间有误差，这个误差就是模型误差.

在数学模型中通常包含一些观测数据，这些观测数据往往带有误差，称之为“观测误差”. 例如用公式

$$s=\frac{1}{2}gt^2$$

计算自由落体在 t 秒内下落的距离时要测量时间 t，测量过程中产生的误差为观测误差.

在解较复杂的数学模型时，往往没有解的精确表达式，必须用一种近似方法（或称数值方法）求解，模型的准确解与数值方法的准确解之间的差称为“方法误差”. 由于方法误差往往是将无穷多项的和“截断”为有穷多项的和的过程中产生的，所以有时又称“截断误差”. 例如通过公式

$$\frac{\pi^2}{6}=\frac{1}{1^2}+\frac{1}{2^2}+\cdots+\frac{1}{n^2}+\cdots$$

计算 π 的值，可先求 $\frac{\pi^2}{6}$ 的值，再求 π 的值. 此公式的右边有无穷多项，显然不能在有限步内求得精确值，我们取前 n 项的和作为 $\frac{\pi^2}{6}$ 的近似值. 这个近似值与 $\frac{\pi^2}{6}$ 的真值间的误差为截断误差.

用计算工具进行计算时，只能取有限位数字进行运算. 因此有些数（比如 π）在参与计算时，必须经过舍入得到一个与它近似的有规定位数的数. 这个过程中产生的误差称为“舍入误差”. 例如 $\pi=3.14159265\cdots$，我们四舍五入取小数点后 5 位得 3.14159，则误差为 $3.14159-3.14159265\cdots=-0.00000265\cdots$，这就是近似数 3.14159 的舍入误差.

上面我们是从误差的来源来给误差分类的，一共分为四类（模型误差、观测误差、方法误差、舍入误差）. 在算法设计中，我们考虑的是后面两种，即设计数值算法时，如何使方法误差小、舍入误差积累少.

严格地说，设 x 是某个量的精确值，x^* 是它的近似值，我们称 x^*-x 为近似数 x^* 的**误差**. 因为有时精确值 x 是求不到的，也求不到误差 x^*-x，因此在实际应用中我们估计近似数 x^* 的**误差界**（如果能找到一个正数 ε，使 $|x^*-x|\leqslant\varepsilon$，我们称此正数为近似数 x^* 的误差界）. 显然，如果一个近似数是由精确数经四舍五入得来，则这个近似数的误差界可取为它最后一位数字所在数位的半个单位. 例如分数 $A=0.3333333333333\cdots$ 放入取小数点后八位的计算机时要取为 $A^*=0.33333333$，A^* 最后一位上的单位是 0.00000001，于是

$$|A^*-A|=0.00000000333333\cdots\leqslant\frac{1}{2}\times 0.00000001.$$

误差和误差界常常有量纲，通俗地说就是“带单位”. 随所带单位不同，近似值的误差和误差界的数值就不同. 例如，用最小刻度为毫米的尺量桌子的长度，将尺的零刻度处与桌子的一端对齐，尺上离桌子另一端最近的刻度数就认为是桌子的长度，这样测量的误差界为 0.5 毫米，也可以说误差界为 0.05 厘米. 随所带单位不同，近似值的误差界所取数值就不同.

有时用误差或误差界还不足以说明近似数的精确程度，比如，有两只手表，第一只每小时快一分，第二只每天快一分，误差都是 1 分，显然后面这一只走得比较准. 为了说明这一点，我们引入相对误差的概念：

近似数的误差与精确值 x 的比值

$$\frac{x^*-x}{x}$$

称为近似数 x^* 的**相对误差**.

因为精确值 x 不易求得，所以有时称误差与近似值 x^* 的比值 $\frac{x^*-x}{x^*}$ 为近似数 x^* 的相对误差. 类似地我们可以称相对误差绝对值的上界为**相对误差界**. 相对误差和相对误差界是不带单位的，常常用百分数表示.

例 求上面说到的两只表的相对误差.

解:第一只表每小时快一分,所以相对误差为

$$\frac{1\text{分}}{60\text{分}}\approx 1.67\%.$$

而第二只表每天快一分,所以相对误差为

$$\frac{1\text{分}}{(24\times 60)\text{分}}\approx 0.069\%.$$

为了减少误差,在设计算法时我们应该注意下面四个问题:

(1)要避免两个相近的数相减.

例如,设 x^*、y^* 分别是 x、y 的近似值,则 x^*-y^* 是 $x-y$ 的近似值,它的相对误差为

$$\frac{(x^*-y^*)-(x-y)}{x-y},$$

当 x 与 y 相近时,上式的分母很小,在计算机上计算时,可能导致此分数值很大,即 x^*-y^* 的相对误差可能很大.

防止两个相近的数相减的办法之一是利用恒等式. 比如计算 $\frac{1}{961}-\frac{1}{962}$,因为 $\frac{1}{961}$ 与 $\frac{1}{962}$ 是两个相近的数,而 $\frac{1}{961}-\frac{1}{962}=\frac{1}{961\times 962}$,所以我们用右边的式子计算它时能减少误差.

(2)两个相差很大的数相加、相减时,要防止大数"吃掉"小数.

例如用因式分解容易知道一元二次方程

$$x^2-(10^9+1)x+10^9=0$$

的两个根是 $x_1=10^9$ 和 $x_2=1$,但如果我们用只能将数表达到小数点后八位的计算机,按照求根公式计算,求得的两个根分别是 $x_1=10^9$ 和 $x_2=0$,这就是因为在计算机上计算判别式 b^2-4ac 时出现了大数"吃掉"小数的现象. 改进的方法是按照求根公式计算出第一个根 $x_1=10^9$,在求第二个根时,利用根与系数的关系 $x_1x_2=10^9$ 得 $x_2=10^9/10^9=1$.

(3)避免用绝对值很小的数作除数.

用绝对值很小的数作除数时,往往也会引起结果的较大误差,下面给出一个例子.

例 取小数点后 4 位计算,用消元法解二元一次方程组

$$\begin{cases}0.0003x_1+3.0000x_2=2.0001,\\1.0000x_1+1.0000x_2=1.0000.\end{cases}$$

分析:若用 $-\frac{1.0000}{0.0003}$ 乘第一个方程加到第二个方程,消去第二个方程中含 x_1 的项,得到方程组

$$\begin{cases}0.0003x_1+3.0000x_2=2.0001,\\9999.0x_2=6666.0,\end{cases}$$

因此得 $x_2=0.6667$,代入第一个方程得 $x_1=0$. 显然与此方程组的精确解 $x_1=\frac{1}{3}$,$x_2=\frac{2}{3}$ 相差很大,原因是作除数的数0.0003太小.

如果用 $-\frac{0.0003}{1.0000}$ 乘第二个方程加到第一个方程,消去第一个方程中含 x_1 的项,再求解得 $x_2=0.6667$,$x_1=0.3333$,此结果就相当精确.

(4)注意减少运算次数.

减少运算次数可缩短计算机的运算时间，减少误差积累. 因此我们设计算法时，应尽量减少运算次数. 教材中计算多项式值的秦九韶法就是减少计算误差的典型例子.

3. BASIC 语言补充

(1)暂停语句和注释语句

暂停语句在调试程序时非常有用，它的语句形式特别简单. 即

行号　STOP

其作用是暂停程序的执行. 用户可在此时做查看变量的值等其它工作，在需要继续运行时只要打入 CONT 加回车键即可. 例如下面的程序

```
10  READ X
20  PRINT X
30  STOP
40  PRINT SQR(X)
50  GOTO 10
60  DATA……(若干数)
70  END
```

中的 30 语句为暂停语句，这时可观察 20 语句打印出的变量 X 的值，如果值大于等于 0 则打入 CONT 继续执行.

应指出 STOP 语句与 END 语句是不同的. 执行 STOP 语句后，各变量保持原值；而执行 END 语句后，各变量所取得的值不再保留.

注释语句又称 REM 语句，它是非执行语句，即此语句不使计算机执行任何操作，它的作用是对下面的程序或程序段作出注释，往往是对程序段或变量的功能作出交待，使程序可读性增加. 例如

```
REM  A GAME PROGRAM
```

(2)转子语句与返回语句

在 BASIC 程序中可将反复执行的部分编成一个程序段，称之为子程序，在需要这段程序时就调用它. 这样可增加程序的结构性. 转子语句的形式为

行号　GOSUB 行号

后面的行号是子程序开始的行号. 程序运行到 GOSUB 语句时，会转向所指行号的子程序，并执行子程序，同时记下 GOSUB 语句下一行的行号，作为返回的位置.

转子语句要与返回语句配合使用，返回语句的形式为

```
RETURN
```

返回语句出现在子程序中，当执行到此语句时，结束子程序的运行，返回到调用此子程序的程序中的返回位置继续执行. 例如程序

```
10  FOR I=1 TO 10
20  A=INT(10*RND):B=INT(10*RND)
30  PRINT A;"*";B;"=";
40  INPUT C
50  GOSUB 200
60  NEXT I
200 PRINT "*****"
```