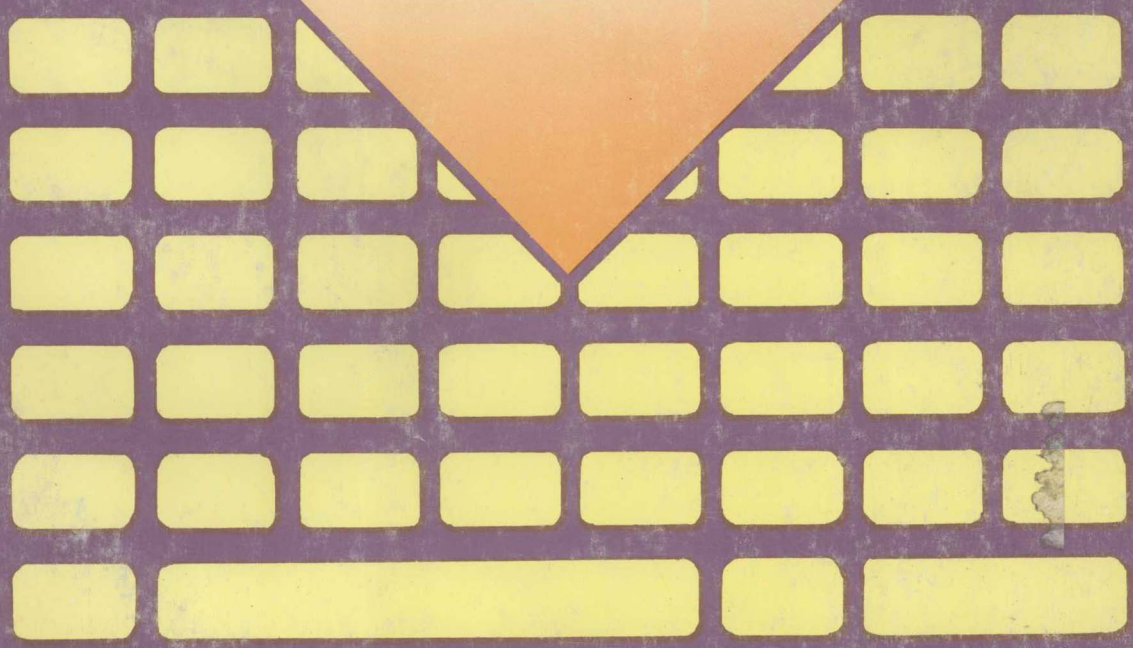
江苏省普通高校计算机等级考试系列教材

FORTRAN 77 程序设计教程

江苏省教育委员会组织编写

陈志明　叶晓风　编著
郑国樑　审阅



南京大学出版社

江苏省教育委员会组织编写
江苏省普通高校计算机等级考试系列教材

FORTRAN 77 程序设计教程

陈志明　叶晓风　编著
郑国樑　审阅

南京大学出版社

内 容 提 要

本书内容主要包括：按标准文本 FORTRAN ANSI X3.9 1978 并以 Microsoft FORTRAN 编译系统为蓝本，较完整地介绍了该语言的基本原理、编程方法、上机实习、实际应用等方面的内容。每章后附有习题、上机实习题。内容覆盖了《江苏省普通高校非计算机专业学生计算机基础和应用能力等级考试大纲》规定的二级 FORTRAN 语言的全部内容，适合作为各类理工科专业及自学考试的教材，本书文字流畅，结构合理，叙述严谨。

江苏省教育委员会组织编写
江苏省普通高校计算机等级考试系列教材
FORTRAN 77 **程序设计教程**
陈志明　叶晓风　编著
郑国樑　审阅
*
南京大学出版社出版
（南京大学校内　邮政编码：210093）
南京豪利电脑公司激光照排
江苏省新华书店发行　　[illegible]海县教育印刷厂印刷
*
开本：787×1092 1/16　印张：24.5 字数：612 千
1996 年 9 月第 1 版　　1998 年 1 月第 2 次印刷
印数 [illegible]—10000
ISBN 7-305-02950-5/TP・148
定价：23.00 元

江苏省普通高校计算机等级考试
系列教材编委会

出版说明

21世纪人类将进入信息化社会，对人才的素质和知识能力结构提出了全新的要求，高等学校面临着十分艰巨的任务。当前，计算机知识和能力不仅是本专科学生必备的知识能力，而且已成为高等教育各学科的重要组成部分，并已成为衡量人才素质的重要尺度之一。高校计算机基础教育如何主动适应经济建设和社会发展，如何提高教育质量，是一个值得重视和研究的新课题。

江苏省教委为了主动适应经济建设和社会发展以及科技进步的需要，强化跨世纪人才素质的培养，从1993年起组织全省普通高校非计算机专业学生进行计算机基础知识和应用能力等级考试，推动了计算机教学内容体系的改革，大面积提高了课程教学质量，1994年12月在江苏省普通高校计算机等级考试指导委员会下面成立了江苏省普通高校计算机等级考试系列教材编委会，根据1995年重新修改的《江苏省普通高校非计算机专业学生计算机基础知识和应用能力等级考试大纲》，组织编写、审定一套服务于我省普通高校计算机等级考试的计算机基础教育教材（第一批为六册：《计算机应用基础教程》、《FORTRAN 77程序设计教程》、《PASCAL程序设计教程》、《C程序设计教程》、《TRUE BASIC程序设计教程》、《数据库(FOXBASE +)教程》），拟在一到两年内分步出版。

这套教材，主要是编者总结了非计算机专业学生计算机基础知识和应用能力等级考试的经验在较长期教学实践基础上成稿，部分书稿是从师生反映较好的讲义中选择修改，其特点为内容丰富，结构严谨，条理清楚，通俗易懂，理论和实践紧密相结合，既可作为普通高校非计算机专业学生学习计算机的一套很好的教材，也可作为广大工程技术人员、科技工作者、管理干部自学计算机知识的宝贵丛书。

这套教材分别由南京大学出版社、东南大学出版社、苏州大学出版社承担出版工作。

我们谨向参与这套教材的编审和出版社编辑表示感谢。限于水平和经验，这套教材还会有许多缺点和疏漏之处，恳请使用这套教材的单位、广大教师和学生提出宝贵的批评或建议，以使这套教材的质量得到进一步提高。

江苏省普通高校计算机
等级考试系列教材编委会
1995年6月

前　言

算法语言 FORTRAN 特别适用于科学和工程计算，用此种语言编制的软件包和程序库很多，如医药生物领域的 BMDP、特征值系统的 ELSPACK、解大型线性方程组的 LINPACE、结构分析的 ASPV、通用的数学统计程序库 IMSL 等等。FORTRAN 语言连同由它编成的软件在世界范围内广为流行。作为理工科大学生，掌握本语言的编程技巧是他们从事数值计算的极佳选择。

本书依照 FORTRAN ANSI X3.9-1978（习惯上称为 FORTRAN 77，本书中所指 FORTRAN 即 FORTRAN 77），并采用 Microsoft FORTRAN version 5.0 编译器为蓝本，内容上覆盖了《江苏省普通高校非计算机专业学生计算机基础知识和应用能力等级考试大纲》二级中的 FORTRAN 77 语言考试要求的全部内容，并适当扩充了有利于提高编程能力的大纲以外的内容，如内部文件及其应用等等。

本课程是一门实践性很强的课程。为加强上机实习训练，本书在内容编排上作了如下考虑：在第一章讲解了直接式程序设计编程的基本要素，输入、输出操作的基本方法。在随后的章节里再逐步精细化，并逐步引入各种语言成分，如判定选择结构、循环结构、过程等等。每章备有习题和结合该章内容的实习题。习题和实习题分为两类，一类为加深对课程内容的理解；另一类为提高编程能力，试图实现从实践中加深对课程内容理解的构想。

本书由陈志明（一、二、三、五、六、八章，附录）、叶晓风（四、七章）编写，全书内容系作者在多年讲稿的基础上修改补充而成。由南京大学计算机科学系博士生导师郑国樑教授审阅。最后的定稿中吸收了他的许多宝贵意见和建议，在此深表谢意。

由于编者水平有限，疏漏之处在所难免。敬请广大读者批评指正。

编者

1996 年 6 月

目　录

第一章　FORTRAN 程序概述 …… 1
§1.1　FORTRAN 发展简史 …… 1
§1.2　算法和程序设计 …… 4
§1.3　FORTRAN 语言 …… 8
§1.4　编写 FORTRAN 程序的最基本要素 …… 9
§1.5　程序举例 …… 34
§1.6　FORTRAN 字符集和 ASCII 码序列 …… 36
§1.7　程序的形式及书写规则 …… 37
§1.8　上机实习简介 …… 41
习题一 …… 47
本章实习内容 …… 48
附表：可执行和不可执行语句 …… 49
第二章　表达式和赋值语句 …… 51
§2.1　数据类型及类型说明 …… 51
§2.2　常数和变量 …… 53
§2.3　算术表达式 …… 58
§2.4　函数 …… 62
§2.5　算术赋值语句和规则 …… 69
§2.6　各种数值数据的输入、输出 …… 70
§2.7　程序举例 …… 73
习题二 …… 77
本章实习内容 …… 78
第三章　程序的控制转移 …… 80
§3.1　转移语句 …… 80
§3.2　逻辑和关系表达式 …… 85
§3.3　有条件转移 …… 91
§3.4　程序停、暂停、结束 …… 115
§3.5　程序举例 …… 116
习题三 …… 123
本章实习内容 …… 125
第四章　数组和循环 …… 129
§4.1　数组 …… 129
§4.2　数组元素 …… 131

§4.3 循环次数已知的循环:DO 循环 …… 137
§4.4 多重 DO 循环 …… 149
§4.5 循环次数未知的循环 …… 158
§4.6 数组的输入、输出 …… 162
§4.7 变量和数组的初始化、DATA 语句 …… 166
§4.8 程序举例 …… 168
习题四 …… 176
本章实习内容 …… 177
第五章 字符型数据 …… 179
§5.1 字符型常数和变量 …… 179
§5.2 字符型数组 …… 182
§5.3 字符型表达式及赋值 …… 193
§5.4 字符型数据相关联的内部函数 …… 197
§5.5 内部文件 …… 200
§5.6 程序举例 …… 201
习题五 …… 204
本章实习内容 …… 206
第六章 输入、输出综述 …… 208
§6.1 数据输入、输出语句 …… 208
§6.2 输入、输出格式说明 …… 212
§6.3 编辑描述符 …… 214
§6.4 格式描述的其它形式 …… 230
§6.5 格式描述与输入、输出名表间的相互作用——格式控制初步 …… 231
§6.6 格式控制的进一步说明 …… 233
§6.7 走纸格式控制 …… 238
§6.8 无格式输入、输出 …… 238
§6.9 内部文件在输入、输出中的应用 …… 240
§6.10 程序举例 …… 243
习题六 …… 249
本章实习内容 …… 251
第七章 过程和程序结构 …… 253
§7.1 过程的概念 …… 253
§7.2 语句函数 …… 255
§7.3 函数辅程序:外部函数 …… 259
§7.4 子程序辅程序:子程序 …… 269
§7.5 可调数组和假定大小数组 …… 274
§7.6 哑实结合 …… 277
§7.7 全局实体和局部实体 …… 293
§7.8 外部过程的多重入口:ENTRY 语句 …… 296

§7.9 公用区和等价说明 …… 298
§7.10 程序举例 …… 309
习题七 …… 325
本章实习内容 …… 330
第八章 记录和文件 …… 334
§8.1 记录 …… 334
§8.2 文件 …… 337
§8.3 文件操作语句 …… 339
§8.4 高级输入、输出语句 …… 352
§8.5 程序举例 …… 356
习题八 …… 363
本章实习内容 …… 364
附录 A:ASCII 码字符表 …… 366
附录 B:ANSI 标准提供的内部函数 …… 368
附录 C:程序单位中语句和注解行的顺序 …… 371
附录 D:上机实习须知 …… 372
D.1 编辑器:NORTON EDITOR …… 372
D.2 编译器:Microsoft FORTRAN version 5.0 …… 374

第一章　FORTRAN　程序概述

FORTRAN 是一种程序设计语言，主要用于编写科学与工程计算方面的程序。它历史悠久，流传极广，应用广泛。通过本章的学习，读者应该初步了解此种程序设计语言的风格，知晓它最基本的要素，以及用它编写成的程序的结构。

§1.1　FORTRAN 发展简史

FORTRAN 是由 FORmula　TRANslation 两个词的开头拼合而成的，它的意思是公式翻译。这是因为最初提出的 FORTRAN 语言其目标是把数学公式翻译成计算机容易"理解"的形式，从而有了这个名称。1954 年在美国提出了第一个 FORTRAN 语言版本，1957 年首先在 IBM 704 机上实现，至今已近 40 个年头了。这期间，为实现程序设计自动化这一目标先后出现过各种各样的程序设计语言，如 ALGOL，PASCAL，COBOL，BASIC，PL/I，C 程序设计语言等等，它们中间有的颇受人们青睐蓬勃发展，有的却被冷落以至无人问津。FORTRAN 从提出至今不断发展，不断完善，始终长盛不衰。1958 年实现了称为 FORTRAN Ⅱ 的版本；1962 年实现了称为 FORTRAN Ⅳ 的版本。由于发展过程中各厂家根据需要都在这些版本中添加了自己的内容，从而很不统一。1966 年美国标准化协会(ANSI)公布了两个文件：FORTRAN X3.9—1966，它是 FORTRAN 语言的标准文本，相当于 FORTRAN Ⅳ 通常又称为 FORTRAN 66，另一文件是 FORTRAN　X3.10—1966，它是前述文本的子集，相当于上面提到的 FORTRAN Ⅱ 。在进一步的发展中，X3.10 文本被取消了。用 FORTRAN 66 文本书写过大量的程序库、软件包，它起过相当长时间的作用。由于 FORTRAN 66 文本存在许多不足之处，例如它不宜于结构化程序设计，缺少文件操作，因此使用过程中不断增加了新的内容，1977 年实现了一种新的版本，这就是 FORTRAN 77。1978 年 ANSI 公布了文件 FORTRAN X3.9—1978，它就是目前尚在流行的 FORTRAN 77。该文件中，标准也分成两部分：子集及全集。厂家根据发展的水平采用子集或全集。当然，全集的功能强于子集。Microsoft 公司的 FORTRAN 3.0 版实现的是子集，而该公司的 FORTRAN 4.0 版，5.0 版均实现了全集。FORTRAN 77 在建立其标准时考虑到当时有大量用 FORTRAN 66 写成的程序库和软件包，因此它与 FORTRAN 66 是兼容的，绝大部份用老版语言写就的软件几乎不必作什么修改就可以用新版语言的编译器做编译与连接并正确运行，少量的程序可能要略作修改，才能正确运行。我们举一个生成伪随机数的小例子来说明这一点，读者在适当的时候可以上机一试。这个例子在 FORTRAN 77 编译器上编译及连接都能顺利通过，算出的第一个伪随机数也是正确的，但显然计算出十个伪随机数中的后九个则不一定正确。而在原来 FORTRAN 66 的编译器中却正确无误。

例 1.1　(0,1)内均匀分布的伪随机数发生器如下：

```
C An example which pointed out the difference between FORTRAN 66
C and FORTRAN 77. Generate the pesudo random numbers with function RANDOM(X).
      PROGRAM MAIN
      Y=RANDOM (-1.0)
      PRINT 10, Y
   10 FORMAT (1X,' Random Number=', f10.7)
      DO 20 I=2, 10
      Y=RANDOM (0.0)
      PRINT 10, Y
   20 CONTINUE
      STOP
      END
C Pesudo random numbers generator function RANDOM(X)
      FUNCTION RANDOM(X)
      IF(X) 10, 20, 20
   20 RN=RHO * RANDOM
      RN1=AMOD(RN,BN)
      RANDOM=RN1/BN
      RETURN
   10 RHO=7.0* *13
      BN=10.0* *10
      RANDOM=-X
      GOTO 20
      END
```

下面的答案左端是用 Microsoft Fortran 5.0 版运行的结果，它是正确的；右端是用 Bull BOS Fortran 运行的结果，它显然不正确。

```
Random Number=   .6889012          Random Number=   .6889012
Random Number=   .6746961          Random Number=   .7556049
Random Number=   .5370643          Random Number=   .1334073
Random Number=   .2035631          Random Number=   .0667037
Random Number=   .9723027          Random Number=   .1334073
Random Number=   .4205452          Random Number=   .0667037
Random Number=   .0746205          Random Number=   .1334073
Random Number=   .7229909          Random Number=   .0667037
Random Number=   .0049871          Random Number=   .1334073
Random Number=   .0483198          Random Number=   .0667037
Stop - Program terminated.
```

如果将上述程序略作修改，则不论在哪种 FORTRAN 的编译器上运行，其结果都正确。修改工作是在函数 RANDOM(X)中增加了 yr=RANDOM。

```
C An example which pointed out the difference between FORTRAN 66
C and FORTRAN 77. Generate the pesudo random numbers with function RANDOM(X).
      PROGRAM MAIN
```

```
      Y=RANDOM (-1.0)
      PRINT 10, Y
   10 FORMAT (1X,' Random Number=', f10.7)
      DO 20 I=2, 10
      Y=RANDOM (0.0)
      PRINT 10, Y
   20 CONTINUE
      STOP
      END
C Pesudo random numbers generator function RANDOM(X)
      FUNCTION RANDOM(X)
      IF(X) 10, 20, 20
   20 RN=RHO * yr
      RN1=AMOD(RN,BN)
      RANDOM=RN1/BN
      yr=RANDOM
      RETURN
   10 RHO=7.0 * *13
      BN=10.0 * *10
      RANDOM=-X
      yr=RANDOM
      GOTO 20
      END
```

运行的结果如下:左端由 Microsoft Fortran 5.0 算出,右端由 Bull BOS Fortran 算出。两者都正确,且左端与上一程序计算结果一样。数值上,左右略有差异,这是两种编译器在数据十——二进制转换中所取精度不同造成的。

```
 Random Number=   .6889012          Random Number=   .6889012
 Random Number=   .6746961          Random Number=   .6746962
 Random Number=   .5370643          Random Number=   .5370649
 Random Number=   .2035631          Random Number=   .2035682
 Random Number=   .9723027          Random Number=   .9723526
 Random Number=   .4205452          Random Number=   .4210286
 Random Number=   .0746205          Random Number=   .0793039
 Random Number=   .7229909          Random Number=   .7683675
 Random Number=   .0049871          Random Number=   .4446373
 Random Number=   .0483198          Random Number=   .3080471
Stop - Program terminated.
```

在学习掌握了 FORTRAN 77 后,FORTRAN 66 编成的许多软件稍作修改都能正确使用。

FORTRAN 77 从实现、制订标准以来已经历了十几年的时间。从结构化程序设计的角度看,它仍然有不少缺陷,例如循环结构中还缺乏有效手段,因此在实现的过程中,厂商都增添了自己认为必须的内容以扩大功能。新的标准也在酝酿之中。

一种程序设计语言的生命力在于它易学好用，功能强。只有其基本概念容易弄清，便于掌握，才能为世人所接受。FORTRAN 是目前世界上流传最广的计算机语言之一。

FORTRAN 主要用于编写科学与工程计算的软件，其他方面的能力限于语言本身的结构而受到限制。虽然也有人用它写过非数值计算的软件，例如曾经用 FORTRAN 66 写过 dBASE Ⅱ（数据库管理程序），但因嫌麻烦早就放弃了。

当前流行的 FORTRAN 77 编译器版本很多，本教程将以 Microsoft Fortran 优化编译器为主要蓝本，着重介绍符合 ANSI X3.9—1978 FORTRAN 标准的内容。

在编写试用本教程期间，作者们得悉一个早就在酝酿的 FORTRAN 语言的新标准：FORTRAN 90 已基本完成。FORTRAN 90 编译器虽然尚未投入市场，但已经不再仅仅停留在学者们的实验室里。也许不久的将来这种新标准的文本及其编译器将作为商品出现在计算机市场上。作为一种面向科学与工程计算的算法语言，FORTRAN 将在不断发展中变得更加完善：功能更全、更大、更易于使用。

§1.2 算法和程序设计

在计算机上解一个问题就必须有一个解题的步骤，以便说明先做什么，再做什么，最后做什么，这个解题步骤的全体便形成为解该题的算法(Algorithm)。所谓算法就是解题过程的详尽而精确的描述，这种描述可以用一切合适的方法来完成，可以用自然语言，可以用图解，可以用框图，也可以是若干种工具联合使用，只要做到：第一，能完整地反映出解决该问题的精确过程；第二，解题者自己能比较容易理解；第三，便于和他人交流。为了说明问题，我们举判定一个奇正整数是否为素数的问题为例。用 r(A/B)表示 A 被 B 除后的余数，例如 r(5/3)=2，r(17/5)=2。

例 1.2 判定奇正整数是否素数的算法描述如下：

1. 给出一个大于 2 的正奇整数 N，
2. 令 k=1(k 将成为接连的奇数，这里是初值)，
3. 令 k=k+2(k 的原值加上 2)，
4. 若 $k>\sqrt{N}$ 则停止，
5. 若 r(N/K)=0 则停止(表明 N 不是素数)，
6. 回到第 3 步(这是在 4、5 两步尚未满足停止的前提下，重复第 4、5 步)。

至于第 4 步只要考虑 N 的平方根，那是由数学论证来决定的，它不属于算法的一部分，但它却是解题中的重要一环。(我们简单地论证一下，原因是这种考虑减少了工作量。设

$$2s+1<\sqrt{N}<2s+3$$

则当直至 2s+1 的奇数均不能整除 N 时，大于 2s+1 的奇数也不可能除尽 N。若不然，有 2k+1>2s+1 能整除 N，则有

$$N=(2k+1)j<(2s+3)^2，且\ k\geqslant s+1$$

此即 $2k+1\geqslant 2s+3$，于是 $j<2s+3$ 且它也是奇数，故 $j\leqslant 2s+1$ 这是不可能的。这种论证使得被用来测试的数据，即 k 的个数大大减少，从而节省了工作量。)

实际问题中的算法当然远比这个例子复杂，但复杂的问题可以分解成若干个较为简单的算法，而后者则可各个击破，合在一起则原问题就可解决了。我们也以一个简单例子来说明。

例 1.3 寻找前P个素数，并显示它们，这里P为正整数。此问题求解的算法为

1. 给出正整数P的值，
2. 令N＝1(N是下一个被检验是否为素数的奇数)，
3. 令Q＝1(Q表示找到的素数个数)；打印整数2(它是第一个素数)，
4. 令Q＝Q＋1(Q的作用是计数器)
5. 若Q＞P则停止(表示已找到前P个素数)
6. 令N＝N＋2(下一个奇数)
7. 检验N是否是素数(可利用上一个算法)，
8. 若N是素数则打印它并回到第4步，否则回到第6步。

这里的第7步是检验N是否为素数的全部细节，也就是前面所描述的那个算法。现在，寻找前P个素数的算法分成了两大部分，一部份用来确定等待判定的奇整数，另一部分则用来检验该奇整数是否为素数，而后一部分又是一个独立的算法。

所谓算法设计就是设计解决某个问题的步骤，开始阶段可能粗糙一些，有一个大体的设想，然后再把各个部份精细化，分解成一个个相对独立的小部分，直到具体步骤。算法设计的好坏直接关系到解题的效率，如果某个问题一时设计不出算法，那么暂时也就不能求解了。

上面所介绍的例子中我们采用的是自然语言来描写算法，此法可行但直观性尚嫌不足，如果采用框图方式，可能直观一些。以下是第二个例子的框图。

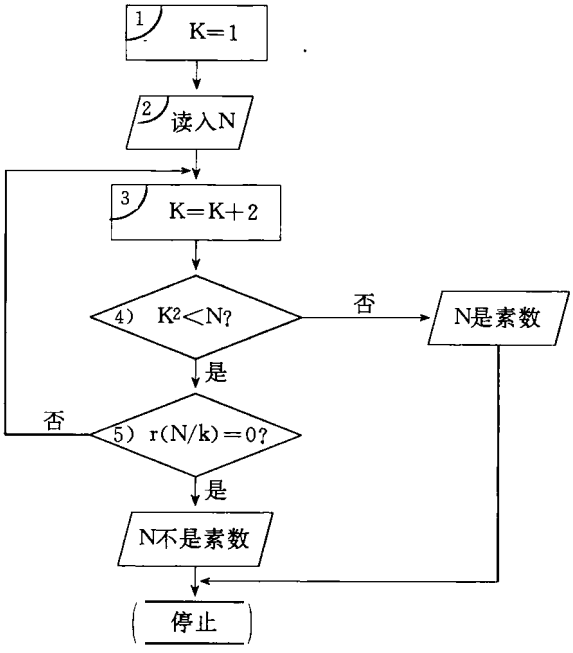


图 1.1

第三个例子的框图如下：

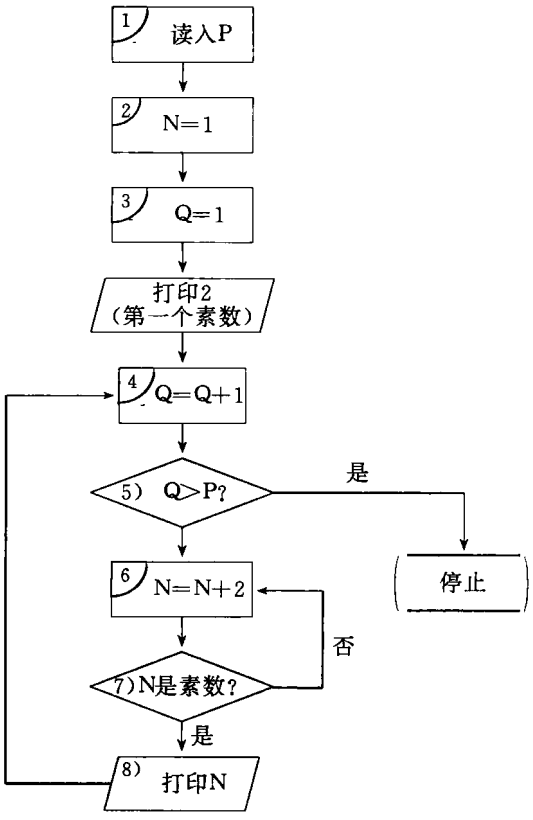


图 1.2

这里的第 7 框即为前一例子的整个框图。

以下是程序设计算法描述中常用的框图符号。

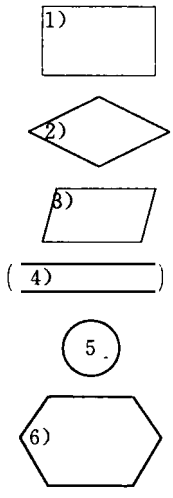


图 1.3

1)用于描述处理、加工

2)用于判定

3)用于输入,输出

4)用于开始、结束

5)用于接合,当框图在一处画不下时,只能在另一处继续画,此时用这种框表明连接的那个点

6)用于描述一些准备工作,例如描述循环

有了解一个问题的算法之后,下一步是实现该算法,这就是根据算法描述进行程序设计。所谓程序设计是设计、编制和调试程序的方法与过程。为了在计算机上解题,必须用计算机所能理解的语言来描述解题的数据和算法,即必须编制程序,这种工作称为程序设计。编程序的工具很多,可以用计算机懂得的语言——二进制指令即计算机机器语言编程序,但这对一般非专业人员是太困难了;可以用一种汇编语言来编程序,它比前者容易掌握一些,但通常也是专门从事计算机

专业的人用得多些;还可以用所谓高级程序设计语言来编程序,这是大多数非专业计算机人士采用的办法。高级程序设计语言种类繁多,常用的有 PASCAL,BASIC,COBOL,C 程序设计语言,还有本书将要介绍的 FORTRAN 语言。

对前面的例子,用 FORTRAN,编制出如下的程序:

```
      K=1
      READ*,N
2     K=K+2
      IF(K**2.LT.N)THEN
      IF(N-(N/K)*K.NE.0)GOTO 2
      PRINT *,'N isn''t a prime number.'
      ELSE
      PRINT *,'N is a prime number.'
      ENDIF
      END
```

这里不具体解释此程序的意义,细心的读者会发现它很像用英语将例 1 的算法描述了一遍,而且它与框图 1 也几乎处处可对上号,因为事实上它是用程序设计语言表达第二个例子的算法的结果。如果在第三个例子中要采用它,只须作少量的改动,因为那里只须检验 N 是否素数而不必把检验的结果显示于屏幕上。我们编出这部分程序如下:

```
      INTEGER P,Q
      READ*,P
      N=1
      Q=1
      PRINT*,2,'is the first prime number'
1     Q=Q+1
      IF(Q.GT.P) STOP
3     N=N+2
      K=1
2     K=K+2
      IF(K**2.LE.N)THEN
      IF(N-(N/K)*K.NE.0)GOTO 2
      GOTO 3
      ELSE
      PRINT,N,' is the ',Q,'th prime number'
      ENDIF
      GOTO 1
      END
```

这个程序中检验一个奇数是否为素数的部分是上一个程序略加修改而成的。当 N 被某个奇数 K 整除时,计数器 Q 不应加 1,因为此刻 N 不是素数,接着要做的事应当是准备检验下一个 N 是否素数,这就是修改的依据。

在这一节里,我们介绍了算法及其程序实现。虽然所描述的程序读者尚不能确切理解,但看一看之后大体上也了解了它的含意。这里介绍的过程具有普遍的意义,今后的任务只是更深

入地理解它们，要解决的问题比这两个例子复杂一些，究其实质则与本小节介绍者大同小异。

§1.3 FORTRAN 语言

上一小节的例子程序就是用 FORTRAN 语言编写的，它的描述十分接近于自然语言英语，读起来容易理解。但它是一种程序设计语言，因此 $k^2 \geqslant N$ 必须写成 K * * 2. GE. N，而 N 能否被 K 整除则可写成 N—(N/K) * K . NE. 0，这里 * 是乘法符号，(N/K) * K 并不一定等于 N 等等都表明了 FORTRAN 语言的特征。

FORTRAN 语言有严格的语法定义，不允许任何二义性的内容引入其语法，一条语法规则严格地只有一种解释。FORTRAN 语言的基本单位是语句(Statement)，除了语句函数定义语句及赋值语句之外，所有语句都有一个关键字(Keyword)来表明该语句的意义。而关键字只有当它应当体现出它的意义时它才成其为关键字，否则它便是一般的符号。赋值语句及语句函数定义语句是由一个称为赋值号的等号(=)来确定其意义的。此外，语句函数定义语句还由其出现于程序中的位置、形式来确定其意义。关键字，例如上一节例 1 中的 IF，END，GOTO 等等，允许作为别的符号用，例如，当作变量用：

```
GOTO=20
```

也是正确的，此时它不再具备原有的作用了。

FORTRAN 语言规定一行上只允许写一个语句，如果某语句太长，则可将一个语句写在若干行上。为了让若干行代表的是同一个语句，必须有某种措施让人及计算机了解这一点，这些具体问题留待下面的章节去讲解。

用 FORTRAN 语言写成的程序，人可以读懂，但计算机则不能理解，必须有一个称为语言处理器或编译器(Compiler)的翻译程序将它翻译成机器能理解的语言，再经过一个称之为连接编辑器(Linker)的实用程序将翻译过的语汇编辑成一条条指令，计算机才能识别，才能执行。FORTRAN 语言写成的程序和其它高级程序设计语言写成的程序都必须经过这一过程。

用 FORTRAN 语言写成的程序称为 FORTRAN 程序。一个 FORTRAN 程序通常根据需要划分成若干个程序单位。所谓程序单位是用于完成某些功能的若干语句构成的一段程序。程序单位中又区分为主程序单位、辅程序单位。每种程序单位都有它自己的特征，但它们有一个共同的特征即均以 END 语句作为结尾，它后面不可再有任何语句。一个程序单位必须有一个，也只能有一个 END 语句，它是程序单位内的比较特殊的语句，其特殊的作用在第三章介绍。一个 FORTRAN 程序必须有一个，也只有一个主程序单位。辅程序单位则依据需要可以有若干个(包括零个)。主程序单位有其区别于辅程序单位的特征。它或者有一个 PROGRAM 语句为其命名，此刻它必须是第一个语句，或者什么标志也没有，有如本章所列举的例子均属于主程序单位，它们都没有专门的标志。有关程序组织及结构的详细介绍请参见本教程第七章。

在本教程中，我们不详细地解释或介绍一条条语法规定，那是标准文本的事。我们只介绍主要语句应如何正确使用的一些规定，并通过在计算机上的实践来加深理解、熟练运用。