Mc Graw Hill Education

（第三版）

Fortran 95/2003 程序设计

```
END PROGRAM

SUBROUTINE compl
!
!  Purpose:
!    Subroutine to add two complex numbers (x1, y1) and
!    (x2, y2), and store the result in (x3, y3).
!
IMPLICIT NONE

REAL, INTENT(IN) :: x1, y1, x2, y2
REAL, INTENT(OUT) :: x3, y3
x3 = x1 + x2
y3 = y1 + y2

END SUBROUTINE complex_add

SUBROUTINE complex_divide ( x1, y1, x2, y2, x3, y3 )
!
!  Purpose:
!    Subroutine to divide two complex numbers (x1, y1) and
!    (x2, y2), and store the result in (x3, y3).
!
IMPLICIT NONE

REAL, INTENT(IN) :: x1, y1, x2, y2
REAL, IN         :: x3, y3
REAL ::
denom = x2      y2**2
x3 = (x1 * x2 + y1 * y2) / denom
y3 = (y1 * x2 - x1 * y2) / denom

END SUBROUTINE complex_divide
```

Stephen J. Chapman 著
刘 瑾 庞岩梅 赵 越 等 译
章小莉 审校

中国电力出版社
www.cepp.com.cn

（第三版）

Fortran 95/2003 程序设计

Stephen J. Chapman 著
刘 瑾 庞岩梅 赵 越 等 译
章小莉 审校

```
END PROGRAM
SUBROUTINE complex_add ( x1, y1, x2, y2, x3, y3 )
!
!  Purpose:
!    Subroutine to add two complex numbers (x1, y1) and
!    (x2, y2), and store the result in (x3, y3).
!
IMPLICIT NONE
REAL, INTENT(IN) :: x1, y1, x2, y2
REAL, INTENT(OUT) :: x3, y3
x3 = x1 + x2
y3 = y1 + y2
END SUBROUTINE complex_add
SUBROUTINE complex_divide ( x1, y1, x2, y2, x3, y3 )
!
!  Purpose:
!    Subroutine to divide two complex numbers (x1, y1) and
!    (x2, y2), and store the result in (x3, y3).
!
IMPLICIT NONE
REAL, INTENT(IN) :: x1, y1, x2, y2
REAL, INTENT(OUT) :: x3, y3
REAL :: denom
denom = x2**2 + y2**2
x3 = (x1 * x2 + y1 * y2) / denom
y3 = (y1 * x2 - x1 * y2) / denom
END SUBROUTINE complex_divide
```

中国电力出版社
www.cepp.com.cn

内 容 提 要

Fortran 是计算世界最早出现的高级程序设计语言之一，随着面向对象编程时代的到来，Fortran 语言不仅保持了发展的步伐，而且继续在科学计算方面领先。

本书在第 2～7 章介绍了 Fortran 语言基础知识，为初学者提供入门学习资料；在第 8～15 章介绍了 Fortran 语言高级特性，为深入用好 Fortran 语言提供支持；在第 16 章讲述了 Fortran 语言面向对象编程方法，支持代码的复用实现；在第 17 章简述 Fortran 95 版本宣布废弃的特性，支持人们对新变化的认识。在附录 B 详细列出内置函数功能说明，使本书犹如 Fortran 语言编程指南。

书中语言浅显易懂，例题详细展示知识的用法，测验帮助读者检验学习效果，涉及各学科实际工程计算作业有利于培养解决科学计算方面问题的能力。

本书是一切乐于用 Fortran 语言进行计算的读者的首选。

图书在版编目（CIP）数据

Fortran 95/2003 程序设计：第 3 版 /（美）查普曼（Chapman，S.J.）著；刘瑾等译. —北京：中国电力出版社，2009

书名原文：Fortran 95/2003 for Scientists and Engineers

ISBN 978-7-5083-8670-6

Ⅰ. F… Ⅱ. ①查… ②刘… Ⅲ. FORTRAN 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2009）第 049240 号

著作权合同登记号　　图字：01-2009-1830

中国电力出版社出版、发行

（北京三里河路 6 号　100044　http://www.cepp.com.cn）

汇鑫印务有限公司印刷

各地新华书店经售

*

2009 年 8 月第一版　2009 年 8 月北京第一次印刷

787 毫米×1092 毫米　16 开本　50.75 印张　1249 千字

印数 0001—4000 册　　定价 **88.00** 元

译 者 的 话

Fortran 是最早出现的高级程序设计语言之一，它主要是适合用来解决科学计算方面的问题。今天即使是已经进入到面向对象的编程时代，本书的第 16 章专门讲述了 Fortran 语言中新添的面向对象特性，但是该语言依然保持住了很好地解决科学计算的优良特性。如，有专用的 COMPLEX 数据类型，还有大量专门用于完成计算的函数（见附录 B）。这一切都使得每一个应用 Fortran 语言编程的人们乐于购买本书，掌握 Fortran 语言的新特性，了解 Fortran 语言的新发展。

本书其实不单单是把一个 Fortran 语言的老手作为读者对象。因为它在第 1 章从计算机基本结构出发，讲述了计算机中数据的表示、Fortran 语言的发展历史以及编程的好习惯如何培养。在第 2～7 章介绍了 Fortran 语言的基本知识。这些为新上路的人学习 Fortran 语言、掌握编程方法提供了可能。

此外，在第 8～15 章介绍了 Fortran 语言的高级特性，为读者深入用好 Fortran 语言提供了强有力的支持。

最后，本书的第 17 章还一一简述了 Fortran 95 版本已经宣布废弃的、但是在遗留的旧版 Fortran 程序中常常会遇到的 Fortran 特性。

作者所用语言浅显易懂，每章每节的学习目标明确，时而穿插有测验，帮助读者检验自己的学习效果。各章节中给出的例题更是展示了所学知识的具体用法、重点提示了良好编程技巧与注意事项，对培养好的编程方法大有益处。

本书既适合初学 Fortran 语言程序设计的人们选用，又适合已经掌握 Fortran 语言，但希望了解 Fortran 2003 新特性，掌握 Fortran 面向对象编程技术的工程技术人员和科学研究人员选用。还有一直耕耘在科技界与工程界、用 Fortran 语言程序帮助解决计算问题的人们，能够在第 17 章的帮助下继续使用已有程序开展工作。本书作为工科院校的教材也是很好的选择。

本书从始至终强调读者树立实际工程编程思想，致力于培养读者编写可读性好、维护性强和完整性好的程序能力。书中除配有大量的工程实例外，还在每章后面的习题中配有很多练习，其中不少练习涉及各个学科的实际工程计算。例如，相对论、电子工程、万有引力、逃逸速度、双曲余弦、振动周期、无线电接收机等的编程。这非常有利于读者明白所学知识在实际工程中的运用，掌握所学知识、增加学习兴趣。本书还在扉页给出知识点的快速查找索引表，在封底列举常用属性的简介，引导读者快速了解和查阅所需知识点。

本书主要译者是刘瑾、庞岩梅、赵越、王艳红、章小莉等，还有孙鹏飞、温洪波、胡齐、纪旭东、刘宇栋、李冬冬、何洪林、杨丹、周志全、曹长宏、纪睿琪、章晓盛、章地等也参与了翻译工作。在此对大家的精诚合作表示深深地感谢！

由于时间仓促，译者水平有限，译文中难免存在不足之处，尽请读者批评指正。

译 者

2009 年 6 于北京

前　言

本书第一版是我编写维护国防和地球物理领域的大型 Fortran 程序的产物。我在工作期间，成功编写了大型程序，这取决于策略和技术，很显然，那时候对于一个年轻的工程师来说，维护 Fortran 程序与学校学习 Fortran 程序的编写是完全不同的。一旦程序投入服务，维护和修改大型程序绝对需要高昂的令人难于置信的费用。因为编程的人容易理解程序，而修改程序的人相对于原程序员来说就很难明白它们的内容了。我编写本书的目标是，既传授 Fortran 语言的基础知识，又较好地传授编写和维护程序的技术。另外，还希望本书对今后打算从事编程工作的学生有一定的参考价值。

在学生学习程序设计过程的早期，要教会他们花更多的精力来保证程序的可维护性是非常难的。因为课堂上的编程任务都很简单，一个人足可在短期内完成编程任务，且这些程序也不需要多年的维护。也因为项目简单，一个学生只要参加了课程学习，完成所有编程任务，通过考试，即使始终不学实际工作中参加大型编程项目时需要的经验，常常就能胜任从“编写”到产生代码这一整个过程的任务。

本书一开始传授编写的 Fortran 程序就适用于大型项目开发，这样可以强调在编代码前认真进行设计的重要性。其中设计过程使用的技术是自顶向下设计技术，即把大型程序分解为可以单独实现的若干个逻辑部分。书中还强调完成单个逻辑部分编写过程的重要性和在将各个独立部分集成为最终产品的过程开始之前测试单元的重要性。

另外，本书传授的 Fortran 程序是工程师和科学家在实际工作与研究中都会遇到的。在所有编程环境中一个生活事实是很常见的：必修维护大量遗留代码。在特殊场合的遗留代码起初是用 Fortran IV（或甚至更早版本）编写的，今天这些程序的结构已不再被使用。例如，这些代码可以通过使用 IF 语句来完成或计算转到或赋值转到 GO TO 语句。第 17 章将介绍语言中那些不再被使用但是在遗留代码中还是会遇到的旧特性。并强调在新程序中应该永不再用那些旧特性，教会学生在遇到这些特性时，如何处理它们。

第三版的变化

本书第三版是直接基于 Fortran 90/95 （适于科学家和工程师） 第二版而写的，保留了本书第一版本的编写结构，但全书都穿插有 Fortran 2003 的新知识，本书大多数内容既适用于 Fortran 95，也适用于 Fortran 2003。Fortran 2003 专有特性用加阴影的背景标识。

从逻辑上来说，Fortran 2003 大部分新增技术是对 Fortran 95 的扩展，每章的相应位置都汇总了这些新技术。但是，Fortran 2003 面向对象的编程技术完全是新的，且第 16 章是重新编写的。

绝大多数 Fortran 语言课程授课时间限定为三个月或一个学期，学生从中掌握 Fortran 语言的基础知识和编程基本概念。这些课程的内容为本书第 1～7 章内容，加上可选学的第 8、9 章。给学生打下良好基础，以便他们在实践中用好 Fortran 语言。

有能力的学生、工作中的科学家和工程师会需要第 11～15 章的 COMPLEX（复数）、

派生数据类型和指针知识。工作中的科学家和工程师几乎肯定需要第 17 章中陈旧、丰富和已删除的 Fortran 特性，这些知识很少在教室里传授。但是在本书中包含有这些知识，目的在于面对实际中用 Fortran 语言解决真实问题时，本书依然有参考价值。

本书特点

本书设计了很多特性来强调如何用恰当的方法编写可靠性高的 Fortran 程序。这些特性对于首次学习 Fortran 的学生和实际工程中的人们都很有用。它们是：

1. 现代技术

本书在例题中始终用的是最新特性。Fortran 95/2003 特性中不仅一直保留着 Fortran 语言旧版本的特性，也有了新的可取代它们的新特性。在这种情况下，例题中用的是现代新技术。旧技术的使用大部分被移到第 17 章中讲述，在那里强调了它们是旧版本的、不受欢迎的。留有旧版本 Fortran 特性的 Fortran 95/2003 特性有使用模块替代 COMMON（通用）块来实现数据共享，DO…END DO 循环替代 DO…CONTINUE 循环，内置过程替代语句函数、CASE 结构替代计算转向 GOTO 语句。

2. 强制类型

全书一直使用 IMPLICIT NONE 语句来强制每个程序中的每个变量类型要显式给出，以便编译时捕捉到常见的打错字。与程序中每个变量显式声明一样，书中强调创建数据字典的重要性。该字典描述程序中每个变量的作用。

3. 自顶向下设计方法

本书第 3 章介绍了自顶向下设计方法，并且随后的其他章节一直在用该方法。这一方法鼓励学生在开始编代码前，仔细思考，对程序进行好的设计。强调明确定义问题的重要性，以便在开始任何其他工作之前准备好需要的输入和输出数据。一旦问题被恰当定义，紧接着就教授学生逐步细化问题，即将问题分解得更小，把单个子任务设计为子程序或函数，最后告诉学生每个阶段中测试的重要性，包括关于构建程序的单元测试和最终产品的集成测试。书中给出了几个程序样例说明如何进行测试，这些程序可以在一些数据集上正确运行，而在另一些数据集上运行时却失败。通过学习本书可以知道标准的程序设计过程是：

（1）清晰地说明要解决的问题；

（2）定义程序需要的输入和将产生的输出；

（3）描述打算用于程序的算法，这一步涉及到自顶向下、逐步分解、伪代码或流程图；

（4）把算法转换成 Fortran 程序；

（5）测试程序。这一步包括对于特定子程序的单元测试，也包括用许多不同数据完成最终程序的集成测试。

4. 过程

本书强调用子程序和函数来实现在逻辑上由大任务分解出来的子任务，并利用过程隐藏数据，还强调要重视在将子任务集成为最后的程序之前的单元测试。另外，书中还介绍了使用过程时常见的错误，以及怎样避免这些错误（参数类型不匹配、数组长度不匹配等）。书中强调对过程要用好显式接口，因为它使得在对 Fortran 编译时，编译器能尽可能多地捕捉常见的编程错误。

5. 简版和标准版 Fortran 95/2003

本书强调编写简洁 Fortran 代码的重要性，因为这样才可以很容易地实现代码在不同类型计算机之间的移植。书中还教授学生在自己的程序中一定要用标准版的 Fortran 95/2003 语句，以便代码达到最大限度的简洁。另外，书中还教授要多用像 SELECTED_REAL_KIND 以及 ACHAR 和 IACHAR 这类函数，前者能保证程序在计算机间移植时更精确和融合得更好，后者能避免移植时发生将 ASCII 转换为 EBCDIC 的现象。

本书也教授学生不要在少数特殊的过程中用与机器型号有关的代码（如调用与机器系统有关的类库文件），以免在移植程序时不得不重写这些代码。

6. 良好的编程习惯

当介绍良好的编程习惯时，为便于对学生强调它们，这些知识点被突出表示，以示强调这是好的编程做法。另外，每章介绍的“良好编程习惯”在本章之后都进行了汇总。下面举例说明书中如何标示“良好的编程习惯”知识点。

良好的编程习惯

在书写代码时，请保证用多个空格来缩进 IF 语句的语句体，从而使代码的可读性好。

7. 编程警示

书中对编程时需要注意的事项进行了突出表示，以提示要避免它们的发生。下面举例说明书中如何标示“编程警示”知识点。

编程警示

要关注整型数运算，因为整型数除法常常会得出难于预料的结果。

8. 指针和动态数据结构

第 15 章详细讨论了 Fortran 指针，包括指针使用不正确可能带来的问题。如内存不足，指针指向的空间将得不到分配。书中给出了很多动态数据结构样例，包括链表和二叉树。

第 16 章讨论了 Fortran 的对象和面向对象编程，包括涉及程序多态性的动态指针的使用。

9. 注意事项

整本书中有很多的注意事项，这些注意事项列出的是学生可能感兴趣的其他信息。某些注意事项实质上展示了 Fortran 的发展历史。例如，第 1 章有一条注意事项描述了 IBM Model 704，第一台运行 Fortran 的机器。另有一条注意事项是对书本知识的补充。再如，第 9 章的一条注意事项回顾和概述了 Fortran 95/2003 中数组的多处不同。

10. 知识点完整参考资料

最后，书中给出了 Fortran 95/2003 语言的完成参考资料，以便读者在实际应用中能快速查找到需要的有关知识点。这里把特别关注点列入在特性表中，很容易查阅，其中包括晦涩和难于理解的特性，如通过地址传递实现函数名的递送，在表控输入语句中的默认取值等。

关于 Fortran 编译器的注意事项

在写作本书的时候，我用过三种 Fortran 编译器：Lachey/Fujitsu Fortran 95 编译器、Intel Visual Fortran Version 9.1 和 NAGWare Fortran 95 编译器。在本书网址（参见下一页）中可

以找到所有这些编译器的零售商联系方式。

书在编写时，这三种编译器还不支持任何一个 Fortran 2003 专有特性，所以书中关于 Fortran 2003 专有特性的一些样例没有经过编译器的“实际测试”，它们有可能存在错误。一旦编译器添加了对 Fortran 2003 专有特性的支持，McGraw-Hill 和我就会立即测试所有与 Fortran 2003 专有特性有关的代码。若发现问题，我们会把正确的代码发送到本书网站，在本书以后的印刷版本中我们也会力图修正存在的这类错误。

在此要特别感谢 Ian Hounam 和 NAG 公司的开发团队，他们允许我在修订本书时使用 Beta 版的 NAGWare Fortran 95 编译器 5.1 版。NAG 领先于其他 PC 版 Fortran 编译器零售商实现 Fortran 2003 面向对象的特性，且它的工具无偿地提供给我作为准备本书的新资料。

使用本书的最后注意事项

无论我多么努力地审校本书稿件，但是书中还是存在一些印刷和打印错误，如果读者发现这些错误，请通过出版商告诉我，以便后续再版和改版时修正它们。对于您的帮助与支持我深表感谢。

在本书网站上我将提供完整的勘误表和错误改正说明，网址是 www.mhhe.com/chapman3e，请为获取修订信息和最新更新，查看本网站。

致谢

对于以下编辑的无价帮助，我在此深表感谢：

Marvin Bishop, Manhattan College（曼哈顿大学）

Terry Bridgman, Colorado School of Mines（科罗多矿业大学）

Kyle V. Camarda, University of Kansas（堪萨斯大学）

Charlotte Coker, Mississippi State University（密西西地州立大学）

Keith Hohn, Kansas State University（堪萨斯州立大学

Mark S. Hutchenreuther, California Polytechnic State University（加州理工州立大学）

Larry E. Johnson, Colorado School of Mines（科罗多矿业大学）

Joseph M. Londino, Jr., Christian Brothers University（克里斯蒂恩兄弟大学）

Joseph J. Wong, Worcester Polytechnic Institute（伍斯特理工学院）

最后，我要感谢我的爱人 Rosa 和我们的孩子 Avi，David，Rachel，Asron，Sarahm Naomi，Ahira 和 Devorah，他们总是激励我做好编写工作！

作者

Stephen J. Chapman

Melbourne, Victoria, Australia

2006 年 8 月 29 日

目　　录

译者的话
前　言
第 1 章　计算机和 Fortran 语言简介 …… 1
1.1　计算机 …… 2
1.2　计算机中数据的表示 …… 3
1.3　计算机语言 …… 10
1.4　Fortran 语言发展历史 …… 10
1.5　Fortran 的演进 …… 12
1.6　小结 …… 15
第 2 章　Fortran 基础知识 …… 18
2.1　介绍 …… 18
2.2　Fortran 字符集 …… 18
2.3　Fortran 语句结构 …… 19
2.4　Fortran 程序结构 …… 20
2.5　常数与变量 …… 22
2.6　赋值语句和算术运算 …… 29
2.7　内置函数 …… 37
2.8　表控输入和输出语句 …… 39
2.9　变量初始化 …… 43
2.10　IMPLICIT NONE 语句 …… 45
2.11　程序举例 …… 46
2.12　调试 Fortran 程序 …… 52
2.13　小结 …… 53
第 3 章　程序设计与分支结构 …… 64
3.1　自顶向下设计技术入门 …… 64
3.2　伪代码和流程图的使用 …… 68
3.3　逻辑常数、变量和运算符 …… 70
3.4　控制结构：分支 …… 75
3.5　有关调试 Fortran 程序的问题 …… 93
3.6　小结 …… 94
第 4 章　循环和字符操作 …… 100
4.1　控制结构：循环 …… 100
4.2　字符赋值和字符操作 …… 123

4.3 Fortran 循环的调试 ……135
4.4 小结 ……136
第 5 章 基本的 I/O 概念 ……146
5.1 格式和格式化 WRITE 语句 ……146
5.2 输出设备 ……147
5.3 格式描述符 ……150
5.4 格式化 READ 语句 ……167
5.5 文件及文件处理初步 ……173
5.6 小结 ……190
第 6 章 数组 ……201
6.1 声明数组 ……202
6.2 在 Fortran 语句中使用数组元素 ……203
6.3 在 Fortran 语句中使用整个数组和部分数组 ……214
6.4 输入和输出 ……217
6.5 程序举例 ……222
6.6 什么时候该用数组？ ……236
6.7 小结 ……237
第 7 章 过程 ……244
7.1 子程序 ……246
7.2 用模块共享数据 ……264
7.3 模块过程 ……271
7.4 Fortran 函数 ……274
7.5 过程作为参数传递给其他过程 ……280
7.6 小结 ……285
第 8 章 数组的高级特性 ……298
8.1 二维数组 ……298
8.2 多维数组 ……308
8.3 对数组使用 Fortran 内置函数 ……310
8.4 加掩码的数组赋值：WHERE 结构 ……313
8.5 FORALL 结构 ……315
8.6 可分配数组 ……317
8.7 小结 ……325
第 9 章 过程的附加特性 ……335
9.1 给子程序和函数传递多维数组 ……335
9.2 SAVE 属性和语句 ……346
9.3 过程中的可分配数组 ……350
9.4 过程中的自动数组 ……350

9.5 在 Fortran 2003 过程中的可分配数组 ····· 357
9.6 纯过程和逐元过程 ····· 361
9.7 内部过程 ····· 362
9.8 小结 ····· 364
第 10 章 字符变量的更多特性 ····· 372
10.1 字符比较操作 ····· 373
10.2 内置字符函数 ····· 377
10.3 把字符变量传入子程序或函数 ····· 379
10.4 可变长字符函数 ····· 384
10.5 内部文件 ····· 387
10.6 例题 ····· 387
10.7 小结 ····· 392
第 11 章 附加的内置数据类型 ····· 397
11.1 REAL 数据类型的可选择类别 ····· 397
11.2 INTEGER 数据类型的可选长度 ····· 417
11.3 CHARACTER 数据类型的可选类别 ····· 418
11.4 COMPLEX 数据类型 ····· 419
11.5 小结 ····· 426
第 12 章 派生数据类型 ····· 431
12.1 派生数据类型简介 ····· 431
12.2 派生数据类型的使用 ····· 433
12.3 派生数据类型的输入与输出 ····· 433
12.4 在模块中声明派生数据类型 ····· 434
12.5 从函数返回派生类型 ····· 442
12.6 派生数据类型的动态内存分配 ····· 446
12.7 参数化派生类型 ····· 447
12.8 类型扩展（Fortran 2003 新特性） ····· 448
12.9 类型绑定过程 ····· 449
12.10 ASSOCIATE 结构 ····· 453
12.11 小结 ····· 454
第 13 章 过程和模块的高级特性 ····· 460
13.1 作用范围和作用域 ····· 460
13.2 递归过程 ····· 465
13.3 关键字参数和可选参数 ····· 467
13.4 过程接口和接口块 ····· 471
13.5 通用过程 ····· 475
13.6 用用户自定义操作符和赋值符扩展 Fortran ····· 487
13.7 绑定赋值符和操作符 ····· 498

13.8 限制对模块内容的访问……498
13.9 USE 语句的高级选项……501
13.10 内置模块……504
13.11 访问命令行参数和环境变量……505
13.12 VOLATILE 属性和语句……507
13.13 小结……508
第 14 章 高级 I/O 概念……519
14.1 更多格式描述符……519
14.2 表控输入的默认值……527
14.3 Fortran I/O 语句描述符详述……528
14.4 有名 I/O 列表……547
14.5 未格式化文件……549
14.6 直接访问文件……551
14.7 流访问模式……555
14.8 派生数据类型的非默认 I/O……556
14.9 异步 I/O……563
14.10 访问特定处理机相关的 I/O 系统信息……565
14.11 小结……565
第 15 章 指针和动态数据结构……573
15.1 指针和目标变量……574
15.2 在赋值语句中使用指针……579
15.3 使用数组指针……581
15.4 使用指针的动态内存分配……582
15.5 指针当作派生数据类型的元素……585
15.6 指针数组……595
15.7 在过程中使用指针……598
15.8 过程指针……602
15.9 二叉树结构……603
15.10 小结……620
15.10.3 习题……623
第 16 章 Fortran 面向对象程序设计……627
16.1 面向对象程序设计介绍……628
16.2 Fortran 类的结构……631
16.3 CLASS 保留字……633
16.4 在 Fortran 中实现类和对象……634
16.5 第一个例子：timer 类……637
16.6 方法的分类……642
16.7 对类成员的访问控制……649

16.8 析构函数 …… 649
16.9 继承性和多态性 …… 653
16.10 避免在子类中重载方法 …… 666
16.11 抽象类 …… 667
16.12 小结 …… 685
第 17 章 冗余、废弃以及已被删除的 Fortran 特性 …… 689
17.1 Fortran90 前的字符限制 …… 689
17.2 已被废除的源码格式（不再使用） …… 690
17.3 冗余数据类型 …… 691
17.4 过时、废弃以及不适应需求的说明语句 …… 691
17.5 共享内存空间：COMMON 和 EQUIVALENCE …… 694
17.6 不必要的子程序特性 …… 700
17.7 其他执行控制特征 …… 706
17.8 被废除的分支和循环结构 …… 708
17.9 I/O 语句的冗余特性（不再使用） …… 712
17.10 小结 …… 712
附录 A ASCII 和 EBCDIC 编码系统 …… 718
附录 B Fortran 95/2003 内置过程 …… 723
附录 C Fortran 95/2003 程序中语句工作状态一览表 …… 758
附录 D 术语表 …… 760
附录 E 各章测验的答案 …… 779

第1章

计算机和 Fortran 语言简介

本章学习目标：

- 了解计算机的基本组成
- 理解二进制、八进制和十六进制数据
- 了解 Fortran 语言发展历史

计算机可以说是 20 世纪最重要的发明，它以多种方式深刻地影响着我们的生活。当我们去食品店购物，收银台的扫描设备通过计算机识别我们购买的食物。客户在银行的账户是用计算机来管理，这使得我们无论白天还是晚上什么时候都可以办理业务，因为有很多的计算机一直在服务着。计算机也控制着我们的电话和电力系统、微波炉和其他设备的运行，甚至它还控制着汽车的引擎。今天，如果突然使他们的计算机失灵，那么发达国家几乎一夜之间就会崩溃。考虑到计算机对人们生活的重要性，很难想象它是在大约 65 年前才发明出来的东西。

这是个怎样的设备呢？为什么对我们的生活有如此大的冲击能力？计算机（computer）是一种特殊的机器，它可以存储信息，并能对信息以令人难于置信的极高速度进行算术计算。存储在计算机中的程序（program）能告诉计算机需要按怎样的顺序执行计算任务，从而保证对信息的计算顺利完成。大多数计算机都非常灵活，例如，如果在它的上面执行相应的程序，我用来编写本书的计算机也可以进行银行账户管理。

计算机能够存储巨大容量的信息和相应的处理程序，当需要的时候，也可以使信息被立即有效利用。例如，银行的计算机能存储每个客户所有账户上每一笔资金进出的明细表。更大规模的信息存储例子是，在美国信用卡公司使用计算机存储每个公民的信用历史——照字面的意思来解释，这有数十亿条信息记录。当需要的时候，公司可以搜索这十几亿条的记录信息，从而确认每个人的信用记录，并在几秒钟的时间内向用户出示他的信用状况。

认识到计算机的能力不会超出人们的想象力这一点非常重要。计算机仅仅是按照它的程序步骤在工作着，当计算机展示它做事情的聪明才智时，是因为聪明的人已经编写好了它正执行的程序，即那个舞台人类已经到达，是人类的集体智慧使得计算机创造了奇迹。

本书将教会读者如何编写自己需要的程序，以便计算机完成你所需要它完成的工作。

1.1 计算机

图 1-1 所示框图是典型的计算机结构图，计算机上重要的组成部件（component）是中央处理单元（central processing unit，CPU）、主存（main memory）、辅助存储器（secondary memory）和输入/输出设备（input and output device）。这些组成部件用框图描述如下：

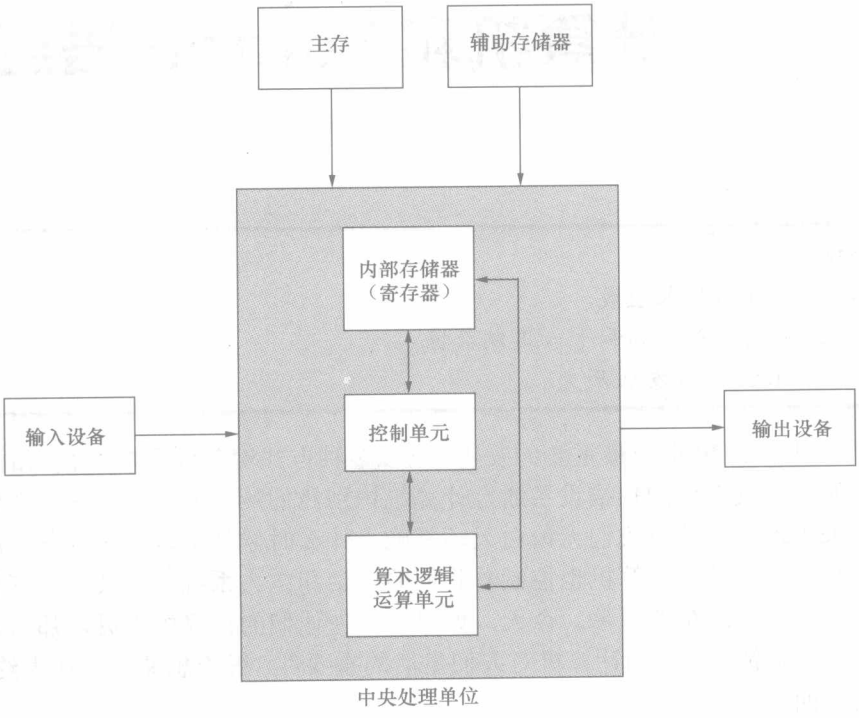

图 1-1 典型的计算机结构框图

1.1.1 CPU

中央处理单元是任何计算机的核心，它由一个控制单元、一个算术逻辑运算单元（ALU）和内部存储器组成。当 ALU 在完成实际算术运算时，CPU 中的控制单元控制着计算机中所有其他部件的操作。CPU 中的内部存储器由一系列寄存器组成，用于临时存储运算时产生的中间结果。

CPU 中的控制单元不仅解释计算机程序指令，也负责从输入设备或主存获取数据值，存储到寄存器中，还负责把数据值从寄存器传送到输出设备或主存。例如，如果程序要求实现两个数的相乘，并保存结果，控制单元将从主存获取两个数据，把它们存储到寄存器中，之后，把数据送入 ALU，同时控制 ALU 完成两个数据相乘，再将结果存储到另一个寄存器中。最后，在 ALU 完成数据相乘之后，控制单元取出目标寄存器中的结果，将它

存储到主存中。

现代 CPU 用并行技术控制 ALU 运算单元，使得完成数据相乘的速度极其快，这使得在给定的时间内可以完成更多的操作。

1.1.2 主存和辅助存储器

计算机存储器主要划分为两大类：主存或内存，以及辅助存储器。主存通常由半导体材料做成，它的速度非常快，但相对价格很高。在现代计算机上，存储在主存中的数据可以用 50mμs 或更少（有时候能这样）的时间取出。因为它是如此之快，以至于现在主存常被用于临时存储正被计算机执行的程序，以及程序需要处理的数据。

主存不用于存储永久保留的程序或数据。大多数主存是不稳定的（volatile），这意味着一旦计算机电源关闭，它的内容就会丢失。此外，主存价格很高，于是我们仅购买足于容纳下任何时候运行的最大程序的主存。

辅助存储器是由比主存速度慢和廉价的设备组成。购买它们的价钱可以比主存低很多，但是它们却能存储下比主存多很多的信息。另外，大多数主存设备是稳定的（nonvolatile），这意味着无论计算机电源是否关闭，它都能保存程序和数据，且不丢失。典型的辅助存储器有硬盘（hard disk）、软盘（floppy disk）、USB 存储棒、CD 和磁带等。辅助存储器通常被用于存储不需要立即使用、但将来还是要用到的程序和数据。

1.1.3 输入和输出设备

数据通过输入设备输入到计算机，通过输出设备输出计算机。现代计算机上最常见的输入设备有键盘和鼠标。人们可以用键盘敲入计算机程序或数据到计算机中。在某些计算机上也可以发现其他类型的输入设备，如扫描仪、麦克风和摄像机。

输出设备允许人们使用存储在计算机中的数据。今天计算机上最常见的输出设备是显示器和打印机，其他类型的输出设备有绘图仪和扬声器。

1.2 计算机中数据的表示

计算机存储器是由单个数十亿的开关组成的，它们每一个都可以开（ON）或关（OFF），且可以在两种状态下随意切换。每个开关表示一位二进制（binary digit）（也称为位，bit）数据，ON 表示二进制的 1，OFF 表示二进制的 0。获取开关状态，单个开关就可以唯一表示数字 0 和 1。显然，由于人们日常需要使用的数据不仅仅是 0 和 1，所以在计算机中需要把多个数字位组合在一起来表示数据。当几个位被组合在一起的时候，可以用来表示二进制（基数为 2）系统中的数据。

最通用的一组位被称为字节（byte）。一个字节是 8 位二进制，它用来表示二进制数据。字节是用于度量计算机内存容量的基本单元。例如，我用来编写本书的计算机的主存大小为 1024MB（1 024 000 000 字节），辅助存储器（磁盘驱动器）容量为 200MB（200 000 000 000 字节）。

在计算机中比字节更大的位组称为字（word）。一个字由 2，4 或更多连续的字节组成，每个字表示存储器中的单个数据。字的大小随计算机类型的不同而不同，因此，字不是可以用来度量计算机存储器大小的单位。现代 CPU 的趋势是每个字的大小为 32 位或 64 位二进制。

1.2.1 二进制数系统

在人们熟悉的基数为 10 的计数制系统中，数据的最小位（最右边的）是个位数（10^0），下一位是十位数（10^1），再下一位是百位数（10^2）等。因此，数据 122_{10} 实际上是（1×10^2）+（2×10^1）+（2×10^0）。在基数为 10 的系统中，每个数字比它右边的数字的幂大 10，如图 1-2（a）所示。

类似的，在二进制系统中，最小位（最右边的）是 1 的位置（2^0），下一位是 2 的位置（2^1），再下一位是 4 的位置（2^2）等。在基数为 2 的系统中，每一位比它右边的数字的幂大 2。例如，二进制数据 101_2 实际上是（1×2^2）+（0×2^1）+（1×2^0）=5，而二进制数 111_2=7，如图 1-2（b）所示。

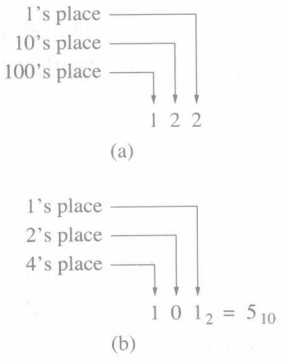

图 1-2 （a）基数为 10 的数据 122 实际上是（1×10^2）+（2×10^1）+（2×10^0）
（b）基数为 2 的数据 101_2 实际上是（1×2^2）+（0×2^1）+（1×2^0）

值得注意的是，三位二进制数字可以被用来表示 8 种可能的取值：0（=000_2）到 7（=111_2）。通常，如果 n 位组合在一起构成一个二进制数，那么它们可以表示 2^n 种可能的取值。因此，一组 8 位（1 个字节）二进制数可以表示 256 种可能取值，一组 16 位（2 个字节）二进制数可以表示 65 535 种可能取值，32 位（4 个字节）二进制数可以表示 4 294 967 296 种可能取值。

在典型的实现系统中，所有可能的取值一半被留存于表示负数，一半用于表示 0 和正数，因此，一组 8 位（1 字节）二进制数常用于表示-128 到+127 的数据，包括 0。一组 16 位（2 字节）二进制数常用于表示-32 768 到+32 767 的数据，包括 0。❶

注意事项：

二进制补码运算

二进制系统中最常用的表示负数的方法称为二的补码表示法。什么是二的补码？它特殊在哪？请看下文。

负数的二进制补码表示

在二进制补码表示法中，数字最右边的位是符号位。如果位为 0，则数据为正数。如果是 1，则数据为负数。为了在二进制补码系统中把正数改成为相应的负数，需要完成以下两步操作：

❶ 在计算机存储器中有几种不同的方式表示负数，在任何一本计算机工程方面的教材上都介绍了这些知识。其中最常用的表示方式被称为二进制补码，在本章的注意事项中对此有详细介绍。