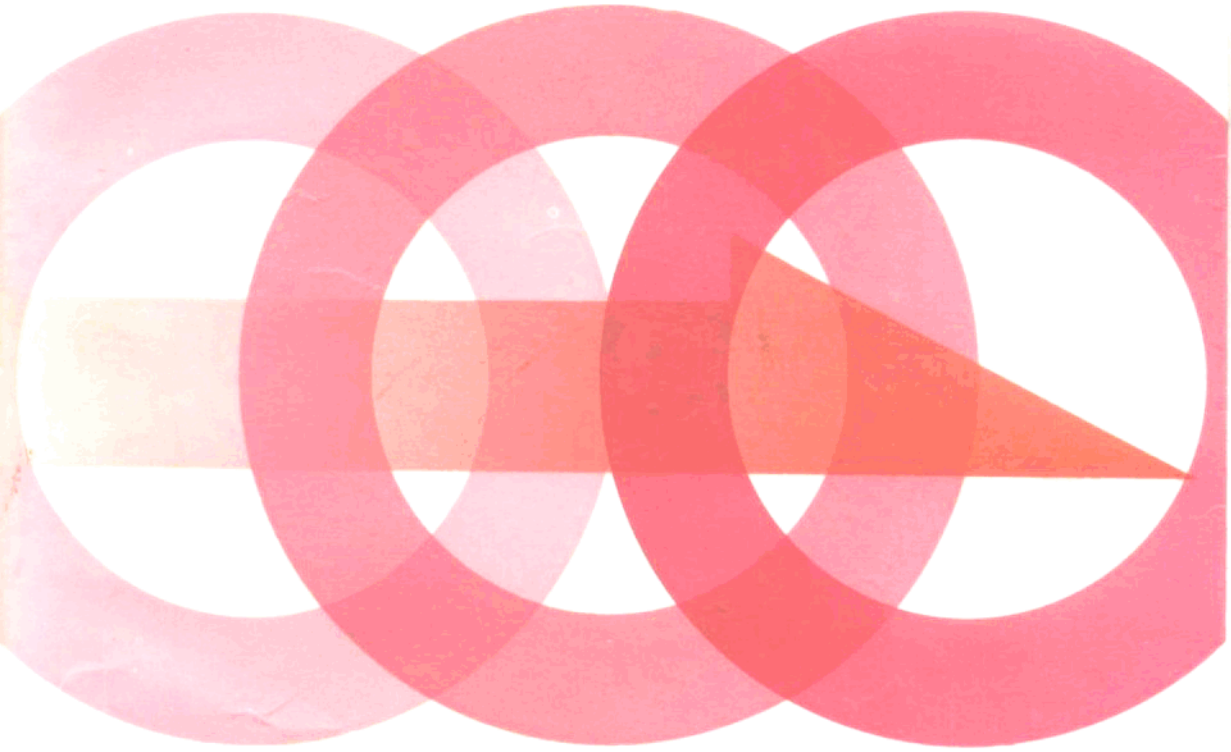
FORTRAN 90学习指南

王文才　郭淑芬　编译　王懋江　校订



GUOFANG GONGYE CHUBANSHE

国防工业出版社

前　言

国际标准化组织已于1991年正式公布了Fortran语言的新标准——Fortran 90。为了向我国广大Fortran语言用户及时介绍Fortran 90的基本内容，我们编译了《Fartran 90学习指南》。

Fortran 90除了把目前广泛应用的Fortran 77作为其中的一个子集之外，还加强了数组处理功能以及用户定义数据类型、民族字符集、模块和过程定义等方面的内容，从而大大加强了数值计算能力。此外，也指出了Fortran 77中有争议和过时的内容，并提出了解决问题的方法。例如：用逻辑IF和块IF取代算术IF语句；用END DO或CONTINUE语句作为循环结束语句；用模块取代COMMON、BLOCK及语句函数等；用ALLOCATE动态分配数组取代EQUIVALENCE语句；用SELECT CASE构造取代计算GO TO语句等。Fortran 90对自由格式书写语句行、源程序语句、续行标识、注释字符、语句之间分隔、字符集扩展和标识符等都作了具体和明确的规定。Fortran 90的编译系统遵循向上兼容的原则，即在旧标准下编写的程序，一般可以在新标准的编译系统中运行，这十分有利于Fortran 90的应用和推广。

本书内容新颖，注重实用并介绍了目前国内外关于Fortran程序设计语言的最新思想和研究成果。对于初学者来说，读了之后就能容易地编写出结构化、模块化且可读性好、易修改、易调试的高质量程序；对已学过和使用Fortran语言的老用户，更能大大提高程序设计水平。

全书共有十一章，并有附录A、B、C、D和部分习题的解答。全书由王文才和郭淑芬同志编译，由王懋江同志校订。

在编译“Fortran 90学习指南”过程中，得到了唐荣锡教授、李其汉教授的指导，王延荣、华国红、马益民、张翅、王雪梅同志给予了热情帮助，在此一并表示感谢。

由于我们的水平有限，书中难免有缺点和错误，恳请广大读者批评指正。

编译者

目　录

第一章　Fortran 发展概况 …… 1
1.1　Fortran 的历史 …… 1
1.2　Fortran 的新标准 …… 2
1.3　Fortran 的标准化工作 …… 3
1.4　Fortran 语言的演变 …… 4
1.5　Fortran 的一致性 …… 5
第二章　语言元素 …… 7
2.1　引言 …… 7
2.2　字符集 …… 7
2.3　记号 …… 8
2.4　源程序的形式 …… 8
2.5　类型的概念 …… 10
2.6　固有类型的文字常数 …… 10
2.6.1　整型文字常数 …… 10
2.6.2　实型文字常数 …… 12
2.6.3　复型文字常数 …… 13
2.6.4　字符型文字常数 …… 13
2.6.5　逻辑型文字常数 …… 15
2.7　名字 …… 15
2.8　固有类型的标量变量 …… 16
2.9　派生数据类型 …… 16
2.10　固有类型的数组 …… 18
2.11　字符子串 …… 19
2.12　对象和子对象 …… 20
2.13　指示字 …… 21
2.14　小结 …… 21
习题 …… 22
第三章　表达式和赋值 …… 24
3.1　引言 …… 24
3.2　标量数值表达式 …… 25
3.3　有定义的和无定义的变量 …… 26
3.4　标量数值赋值 …… 27
3.5　标量关系运算符 …… 28
3.6　标量逻辑表达式和赋值 …… 28
3.7　标量字符表达式和赋值 …… 30
3.8　结构构造符和标量定义的运算符 …… 31
3.9　标量派生赋值 …… 33
3.10　数组表达式 …… 34
3.11　数组赋值 …… 35
3.12　表达式和赋值的指示字 …… 35
3.13　小结 …… 36
习题 …… 36
第四章　控制语句 …… 39
4.1　引言 …… 39
4.2　GO TO 语句 …… 39
4.3　IF 语句和构造 …… 40
4.3.1　IF 语句 …… 40
4.3.2　IF 构造 …… 40
4.4　SELECT CASE 构造 …… 43
4.5　DO 构造 …… 45
4.6　小结 …… 49
习题 …… 51
第五章　程序单元和过程 …… 53
5.1　引言 …… 53
5.2　主程序 …… 54
5.3　STOP语句 …… 54
5.4　外部辅程序 …… 55
5.5　模块 …… 55
5.6　内部辅程序 …… 57
5.7　过程的变元 …… 57
5.7.1　指示字虚变元 …… 59
5.7.2　实变元的限定 …… 59
5.8　RETURN 语句 …… 59
5.9　变元的意图 …… 60
5.10　函数 …… 60
5.10.1　被禁止的副作用 …… 61
5.11　显式与隐式接口 …… 61
5.12　过程作为变元 …… 62
5.13　关键字变元和可选变元 …… 64
5.14　标号的作用域 …… 65
5.15　名字的作用域 …… 65
5.16　递归过程 …… 66
5.17　RESULT 子句 …… 69
5.18　复载 …… 70
5.19　假定字符长度 …… 71
5.20　SUBROUTINE 和 FUNCTION 语句 …… 72
5.21　语句的顺序 …… 72

5.22 小结 ……73
习题 ……74
第六章 数组部分 ……75
6.1 引言 ……75
6.2 零长度数组 ……75
6.3 假定形数组 ……76
6.4 自动对象 ……76
6.5 基本运算和赋值 ……77
6.6 数组值函数 ……78
6.7 堆阵存储 ……78
6.7.1 可分配的数组 ……78
6.7.2 ALLOCATE 语句 ……79
6.7.3 DEALLOCATE 语句 ……80
6.7.4 NULLIFY 语句 ……81
6.8 WHERE 语句和构造 ……81
6.9 数组元素 ……82
6.10 数组片段 ……83
6.11 数组作为结构分量 ……84
6.12 作为别名的指示字 ……85
6.13 数组构造符 ……85
6.14 屏蔽数组 ……86
6.15 小结 ……88
习题 ……89
第七章 说明语句 ……91
7.1 引言 ……91
7.2 隐含类型 ……91
7.3 说明不同形的实体 ……93
7.4 有名常数和常数表达式 ……94
7.5 变量的初值 ……96
7.6 PUBLIC(公用) 和 PRIVATE (专用) 属性 ……98
7.7 POINTER、TARGET 和 ALLOCATABLE 语句 ……99
7.8 INTENT 和 OPTIONAL 语句 ……99
7.9 SAVE 属性 ……100
7.10 USE 语句 ……101
7.11 派生类型定义 ……102
7.12 类型说明语句 ……104
7.13 类型和类型参数说明 ……105
7.14 说明表达式 ……105
7.15 NAMELIST 语句 ……106
7.16 小结 ……106
习题 ……107
第八章 内部过程 ……109
8.1 引言 ……109
8.1.1 关键字调用 ……109
8.1.2 内部过程的种类 ……109
8.1.3 INTRINSIC 语句 ……109
8.2 任意类型询问函数 ……110
8.3 基本数值函数 ……110
8.3.1 可转换的基本函数 ……110
8.3.2 不转换的基本函数 ……111
8.4 基本数学函数 ……111
8.5 基本字符函数和逻辑函数 ……112
8.5.1 字符型-整型转换 ……112
8.5.2 词汇比较函数 ……112
8.5.3 字符串处理基本函数 ……113
8.5.4 逻辑转换 ……113
8.6 非基本字符串处理函数 ……113
8.6.1 询问串处理函数 ……113
8.6.2 串处理转换函数 ……113
8.7 数值询问和操作函数 ……114
8.7.1 整型数据和实型数据的模型 ……114
8.7.2 数值询问函数 ……114
8.7.3 操作实型的基本函数 ……115
8.7.4 类别值的转换函数 ……115
8.8 位操作过程 ……116
8.8.1 询问函数 ……116
8.8.2 基本函数 ……116
8.8.3 基本子程序 ……117
8.9 转换函数 ……117
8.10 向量和矩阵乘积的函数 ……117
8.11 化简数组的变换函数 ……118
8.11.1 单变元情况 ……118
8.11.2 可选变元 DIM ……118
8.11.3 可选变元 MASK ……118
8.12 数组询问函数 ……119
8.13 数组构造和操作函数 ……119
8.13.1 合并基本函数 ……119
8.13.2 压缩数组与非压缩数组 ……119
8.13.3 数组改形 ……120
8.13.4 用于复制的变换函数 ……120
8.13.5 数组移位函数 ……120
8.13.6 矩阵转置 ……120
8.14 几何位置的变换函数 ……121
8.15 非基本内部子程序 ……121
8.15.1 实时时钟 ……121
8.15.2 随机数 ……121

8.16 小结 ……122
习题 ……122
第九章 数据传输 ……123
9.1 引言 ……123
9.2 格式I/O ……123
9.2.1 数的转换 ……123
9.2.2 I/O表 ……124
9.2.3 格式定义 ……125
9.2.4 部件定义 ……126
9.2.5 格式READ ……128
9.2.6 表控输入 ……129
9.2.7 NAMELIST语句 ……131
9.2.8 格式输出 ……132
9.2.9 走纸控制 ……132
9.2.10 非前进I/O ……133
9.3 编辑描述符 ……134
9.3.1 重复数 ……134
9.3.2 数据编辑描述符 ……135
9.3.3 字符串编辑描述符 ……137
9.3.4 控制编辑描述符 ……138
9.4 无格式I/O ……140
9.5 直接存取文件 ……141
9.6 数据传输语句的执行 ……142
9.7 小结 ……142
习题 ……142
第十章 外部文件操作 ……144
10.1 引言 ……144
10.2 文件定位语句 ……144
10.2.1 BACKSPACE语句 ……144
10.2.2 REWIND语句 ……145
10.2.3 ENDFILE语句 ……145
10.2.4 数据传输语句 ……145
10.3 OPEN语句 ……146
10.4 CLOSE语句 ……149
10.5 INQUIRE语句 ……150
10.6 小结 ……153
习题 ……153
第十一章 不赞成的部分 ……154
11.1 引言 ……154
11.2 存储结合 ……154
11.2.1 EQUIVALENCE语句 ……154
11.2.2 COMMON语句 ……156
11.2.3 BLOCK DATA ……157
11.2.4 变元的形不一致 ……158
11.2.5 ENTRY语句 ……159
11.3 新的多余部分 ……160
11.3.1 INCLUDE行 ……160
11.3.2 DO WHILE循环控制 ……161
11.4 旧的多余部分 ……161
11.4.1 固定的源程序形式 ……161
11.4.2 双精度 ……161
11.4.3 计算GO TO ……162
11.4.4 字符长度说明*len ……162
11.4.5 DATA语句的位置 ……162
11.4.6 语句函数 ……162
11.4.7 内部过程的特定名 ……163
附录A 内部过程 ……166
附录B Fortran 90语句表 ……169
附录C 过时的部分 ……171
C.1 算术IF ……171
C.2 DO构造的变形 ……172
C.3 赋值GO TO和赋值格式 ……173
C.4 转移到END IF语句 ……174
C.5 交替RETURN ……174
C.6 PAUSE语句 ……175
附录D Fortran用语 ……175
习题解答 ……182
参考文献 ……194

第一章　Fortran 发展概况

国际标准化组织已于 1991 年正式公布 Fortran 语言的新标准 Fortran 90。我们编译这本书的目的，就是让读者自己学习和理解新标准。为了便于读者阅读和理解，本书以教科书的形式编写。读者在读了第二、三、四章关于语言的组成、表达式以及赋值和控制语句之后，就可以编写很多简单的程序。若读者将每一章中的内容都理解了，则可以编出比较复杂的程序来。本书第二章到第十一章说明语言的各种成分。其中第五章描述了模块新概念并对过程提出了许多扩充。第六章透彻地说明了许多有用的数组细节。第七章说明了数据对象和新派生类型的具体内容。第八章详细论述了已大大扩充了的内部过程。第九章和第十章概括了输入和输出的全部细节。第十一章描述了语言中多余的部分。本书第二到第十一章的每一章结论部分，都将 Fortran 77 与 Fortran 90 的区别作了归纳和总结。

1.1　Fortran 的历史

早期的计算机程序是极为繁琐的。程序员需要详细了解为之编写代码的计算机的指令、寄存器和中央处理器（CPU）的其他有关方面。源程序本身是用数学符号编写的，叫做八进制码。后来采用了助记符，就是所谓的机器码或汇编码的形式。这些编码通过汇编程序转换为指令字。在 50 年代，书写和调试一个程序要用很长时间，很明显，使用这种程序形式是很不方便的，尽管它确实能使 CPU 非常高效地工作。

这些问题促使由 John Backus 率领的 IBM 公司的一个小组研究开发最早的高级语言——Fortran。他们的目标是开发这样一种语言：容易理解，几乎能与汇编语言一样高效地运行。在这方面，他们获得了异常的成功。这种语言确实简单易学，因为几乎可能像抄写数学教科书里的公式一样书写数学公式（实际上，Fortran 这一名字是 FORmula TRANslation 的缩写，意为“公式翻译”）。这样使编写可运行程序比以前快，但由于把许多精力用在编译程序的构造上，所以在效率上也有些小的损失。因此我们可以看出，用 Fortran 作为第一种高级语言是一种创新。但是，Fortran 不仅是一次革新，而且是一次革命。程序员可以摆脱使用汇编语言的冗长乏味的负担，从而使他们能用更多的精力去解决其他问题。更重要的或许是这样一个事实：对于任何一名科学家或工程师，只要他愿意稍加努力去学习和使用 Fortran，他就能使用计算机，从而结束了只有计算机专家才能编写实用程序的历史。

因为 Fortran 满足了现实的需要，所以它传播得很快。在传播和使用过程中不可避免地产生了 Fortran 语言的方言，这给在计算机语言之间交换程序带来了一定的困难。于是经过 4 年的工作，当时的美国标准化协会 ASA〔ASA（American Standard Association）后来改名为美国国家标准协会简称 ANSI（American National Standards Institute）〕，于 1966 年提出了程序语言的第一套标准，这就是现在所熟知的 Fortran 66。实际上，Fortran 66 是各种方言的公共子集，因此每种方言都可以看成这一标准的扩充。用户希望编写可移植的程序就必须注意避免采用这些扩充。

除了简单易学和运行有效外，Fortran 还有其他一些优点。例如，这种语言是和硬件密切联系的易开发的语言，而不是抽象的概念。通过 COMMON 语句和 EQUIVALENCE 语句，为程序员提供了用简单的方法控制存储器分配的可能性，这些对于早期小容量存储器是十分必要的。最近源程序在语法上不依赖任何空格字符，使程序员不必在严格规定的列上书写代码并允许语句主体部分以任意所希望的方式来安排。不言而喻，在现代的情况下，并非所有这些优点都已全部满足了人们的要求。

在 1966 年的标准颁布以后，方言的增加仍然是一个问题。最初的困难是有许多编译程序甚至不向这一标准靠拢。其次是在编译程序中，广泛实现一些对大型程序很有用的功能，但是却忽略了标准。其例子是直接存取文件处理，不同的编译程序采用了不同的方法实现这样一些功能。

这种现状加上语言中存在的一些明显缺陷，例如缺少结构程序设计构造，这样就导致了大量所谓的“预处理程序”的产生。这些程序能够读入一些用定义良好的 Fortran 扩充方言写的源程序，生成标准 Fortran 的二次文本，然后按通常的方法将这些文本提供给 Fortran 编译程序。这就为扩充 Fortran 提供了手段，同时保留了在计算机之间交换经过转换的源程序的能力。同时，大量的这类预处理程序意味着有很多不同的高级方言在使用。虽然用预处理程序写出来的程序可以在 Fortran 源程序级上交换移植，但这种自动生成的 Fortran 程序通常难以读懂，所以人们一般不采用。

1978 年颁布的新标准 Fortran 77，部分地解决了这个问题。这种新标准包括一些新的功能，新功能是建立在预处理程序的基础上。这种新标准不是现有方言的公共子集而是自成一体的新语言。由于新的编译程序未能及时提供使用，Fortran 66 和Fortran 77 之间的过渡期比人们预计的要长得多，两种标准不得不长时间共存。不管怎样，到 80 年代中期，向 Fortran 77 的转变形成潮流，Fortran 66 很快被取代了。这时制造商们也开始停止提供旧的编译程序，从而增强了转变的压力。用新标准编写新的程序是相当简单的，改写旧标准的程序通常也很简单，因为两个标准之间有大量兼容的地方。另一方面，用扩充的语言编写的程序通常较难转换，因为新的编译程序一般不包括Fortran 77 的扩充。实际上，有些编译程序非常严格，仅实现了没有任何扩充的 Fortran 77。

1.2 Fortran 的新标准

30 年过去了，在许多计算机上，Fortran 已不是唯一适用的程序设计语言，一些新的语言发展起来了。事实证明，在某些特殊的应用领域里，一些新语言更为合适，但是 Fortran 的优势仍然是在数值、科学和工程技术应用领域，而且在这些领域里没有更强的竞争对手。Fortran 用户已在 Fortran 代码上进行了巨大的投资，有许多程序（有些长达 10 万条，有些甚至更长）在频繁地使用。然而这不是说人们完全满足于这种语言。为此，ANSI 委派的技术委员会 X3J3 已不止一次地推出新的标准草案。第一个草案是在 1987 年推出的，第二个草案在 1989 年推出，随后在一段时间内征求了意见和进行了修改，国际标准化组织（ISO）终于在 1991 年夏季公布了 Fortran 90 新标准。

为什么要继续修订 Fortran 语言呢？这是因为不但要把厂家的一些语言扩充部分标准化，而且也是为了 Fortran 现代化，使之能赶上语言结构的发展。在这方面，其他语言如 APL、Algol68、Pascal 和Ada 已使用了一些新的概念。在这里。X3J3 可以利用像

数据隐蔽等一些概念的优点。出于同样的考虑，需要开始为危险的存储结合提供代替办法，废弃过时的严格的源程序书写格式，并继续改善语言的正规性，同时提高用语言编程序的安全性，以及使一致性要求更为严格。为保护对 Fortran 77 程序的巨大投资，整个 Fortran 77 被作为 Fortran 90 的子集。

由此可以看出，新的标准不像以前的仅仅是现有实践用法的标准化，而更多的是这种语言的新发展，引入了一些对 Fortran 来说是全新的细节，这些细节吸取了其他语言的经验。最重要的部分是用简单有效的记法处理数组的能力，以及定义和处理用户定义数据类型的能力。前者使许多数学问题编码简化，并使 Fortran 在新一代的超级计算机上更为有效，因为这类数组特点与这些硬件配合得很好。后者使程序员能够用满足其要求的数据类型来表达问题。

这些新增加的部分可以概括如下：

（1）通过将某些部分标明为“过时的”而演变语言的方法；

（2）数组运算；

（3）指示字；

（4）改进数值计算的设施，其中包括一组数值询问函数；

（5）固有类型的参数化，允许处理程序支持短整数、非常大的字符集、双精度以上精度的实型和复型，以及压缩逻辑；

（6）由任意数据类型组成的用户定义的派生数据类型以及对这些结构的运算；

（7）定义称为模块的程序段的设施，这对于全局数据定义和过程库是有用的，它提供了一种封装派生数据类型的安全方法；

（8）要求编译程序能够检测出使用了不符合语言语法规则或过时的构造；

（9）一种更加适用于终端的新源程序书写形式；

（10）新的控制构造，例如 SELECT CASE 构造和 DO 语句的一种新形式；

（11）具有写内部过程和递归过程的能力，以及用任选变元和关键字变元调用过程的能力；

（12）动态存储分配；

（13）输入/输出设施的改善，包括处理部分记录和标准化的 NAMELIST 的设施；

（14）许多新的内部过程。

总之，Fortran 90 中新的部分将保证人们在今后长时间内继续成功地使用 Fortran 语言。Fortran 90 把整个 Fortran 77 作为一个子集的事实，意味着由 Fortran 77 转换到 Fortran 90 是非常简单的。

1.3 Fortran 的标准化工作

1966 年和 1978 年，美国国家标准学会（ANSI）和国际标准化组织两次对 Fortran 语言进行了标准化。负责这项工作的委员会是 X3J3。X3J3 本身由 45 名计算机硬件和软件制造商、用户和学术界代表组成，它是由最终颁布美国标准的 ANSI 任命的，直接向它的上级委员会 X3 报告。X3 负责决定是否采用它们提出的标准草案。在决定过程中，它要尽量保证提案真正体现出有关各方的意见。相应的国际组织是 ISO/IEC JTC1/SC22/WG5。这一组织由国际上的专家组成，负责使一个标准草案在成为一个国家标

准的同时成为国际标准。在我们国内，全国计算机与信息处理标准化技术委员会下属的Fortran 语言标准化工作组是其成员。因此，在 Fortran 90 成为国际标准的过程中，采纳了我国的不少建议。例如在语言中增加民族字符集处理能力是由我国和日本分别提出的，经过各成员国代表投票表决，承认了我国建议的科学性，拒绝了日本的建议，从而采用了中国的建议。

作为漫长的标准化过程的一部分，在 X3 考虑研究标准草案之前，需要 X3J3 的至少 2 / 3 的成员同意后才能向 X3 提交建议草案。一旦 X3 同意接受，首先就得有一段时间征求公众的意见。这时不但要求 X3J3 注意收集公众的反应，而且要对 X3 提出草案修改的意见。这要经过几个反复。正常情况下，一套新标准在得到国际标准化组织承认之前需要经历 18 个阶段。

1986 年 4 月，X3J3 内部对当时的草案进行了一次投票，结果是 16 票赞成 19 票反对。这样，就不得不降低所提草案的水平，以求在下述三派之间取得折衷：要求加入大量新功能的激进派，目标适中的保守派，以及强调长期兼容性的兼容派。到 1989 年 5 月，第二个草案建议稿提交给X3，1990 年 6 月，X3J3 公布了 Fortran 90 标准草案。委员会内部的意见不一致是正常的，之所以经历数年的反复讨论，主要是在下列问题上存在着不同看法，如：

（1）Fortran 90 应当是革新的，还是仅为现有经验的标准化？

（2）这种语言应当是短小而简单的，还是庞大而功能强大的？现在的方案是不是太大了，以致使它不能适合 90 年代的小型机，不能由小型的软件商所实现，而且不能被非专业的程序员所理解（Fortran 的用户大部分是这些人）？

（3）经过 10～20 年的酝酿，用户是否准备放弃现有的部分或者将现存的程序一直运行下去？

（4）这种语言的子集（作为过渡措施）是有效的还是将会妨碍可移植性？

（5）所提出的语言的体系结构是否可行？

（6）用户需要的是一种安全可靠的语言，还是一种允许他们编写技巧性强，往往与汇编语言联系在一起的程序语言？

（7）建议的语言是否实现起来有困难或不能高效率地实现？若因此用户有一个好的新开端，这是否还成为一个问题？

（8）新部分的出现是否会使现有的部分实现起来困难？

到 1991 年夏季，在这些问题上终于取得了一致意见，标准化工作胜利完成，国际标准化组织正式公布了新的 Fortran 标准文本，即 Fortran 90。我们现在期望，在不久的将来能有符合新标准的 Fortran 90 编译系统提供给我国的广大 Fortran 用户使用。

1.4 Fortran 语言的演变

从语言中删去任何现有的部分之前，X3J3 的工作程序要求有一段发出通知的时间。这就意味着，实际上对 Fortran 来说，最短的修订周期也要 10 年左右。删去一些部分的必要性是明显的：如果委员会的工作只是增加新的功能，这种语言将会庞大得出奇，其中包含着许多重复多余的东西。X3J3 最后采用的解决办法是在标准草案后面公布两张表作为附录，这两张表列出那些已经删去或者最后准备要删去的内容。

第一张表是已删除的部分。由于 Fortran 90 包含所有 Fortran 77 的内容，这张表是空白的。

第二张表包含了过时的部分，就是那些被认为已是多余的或用途不大的内容，以及建议下次修订时删除的内容（尽管这对下一届委员会并没有约束作用）。这些过时的部分是：

（1）算术 IF 语句；

（2）从 IF 块之外转移到 END IF 语句；

（3）实型和双精度型的 DO 变量和控制表达式；

（4）采用 DO 循环的公用终端语句以及 CONTINUE 或 END DO 以外的语句作为 DO 循环结束语句；

（5）ASSIGN 和赋值 GO TO 语句；

（6）交替 RETURN 语句；

（7）PAUSE 语句；

（8）赋值 FORMAT 说明符。

如果Fortran 90 标准是成功的，它肯定会继续修订，例如，增加异常处理和并行处理的部分。使这种语言马上就能普及的一个方法是编写包含某些部分并被 X3J3 认为是辅助标准的模块。这些模块的例子可以是矩阵运算和快速傅立叶变换。写这些模块时必须不使用过时的部分。这就提供了一种保证手段，使 Fortran 新标准在今后 10 年，甚至更长的时间里，成为在数值计算和科学应用方面的一种精致而有效的工具。

在后面的章节中将开始对 Fortran 语言的描述，而把过时部分放在附录C中。

1.5 Fortran 的一致性

Fortran 77 标准的缺陷是没有一种语句要求处理系统提供一种手段，以便查出程序对所允许的语法任何一种偏离，而且这种偏离并不与标准定义的语法规则冲突。所以此标准几乎只与程序规则有关而与处理系统无关。人们要求处理系统接受标准一致的程序，而且按照标准理解它，同时受到一些限制，这些限制是处理程序对程序复杂性和规模所加的。倘若深一层的语法及标准中未指明的关系不与标准冲突的话，则允许处理系统接受这种语法及解释这些关系。某些语法的解释是与处理系统有关，即随处理系统而异的。例如，在字符上下文中所允许的字符集是与处理系统有关的。当使用一个与处理系统有关的部分时要小心，以防它导致程序在所希望的处理系统上不可移植。

新标准以不同于老标准的风格写成。语法规则以带有限制的 BNF 形式表达，而语义通过文字加以说明。这种不完全正规的风格不在本书中使用，这里举出一个例子或许是有益的：

R 606	*substring*	is *parent-string* (*substring-range*)
R 607	*parent-string*	is *scalar-variable-name*
		or *array-element*
		or *scalar-structure-component*
		or *scalar-constant*
R 608	*substring-range*	is [*scalar-int-expr*]:[*scalar-int-expr*]

限制：*parent-string* 必须是字符型的。

在 *substring-range* 内，第一个 *scalar-int-expr* 称为起点，而第二个 *scalar-int-expr* 称为终点。

这个标准以如下方式书写，即在编译时间，处理系统可以检查程序是否满足所有的限制。特别是处理系统必须提供查出和报告使用了外加的语法和过时部分的能力。处理系统也必须能查出和报告它不支持的类型参数值的说明（2.5 节）。这些能力对于产生正确和可移植的程序将有很大价值，同时是超出 Fortran 77 一致性要求的。

第二章 语言元素

2.1 引 言

用自然语言书写的文章，诸如一本英语教科书，首先要包含着字母这些基本元素。这些元素组成较大的实体——词汇，用以表达对象、行为和限制条件等基本概念。语言词汇根据某些规则进一步组成较大的单位，即短语和句子。一组规则定义了语法，这就告诉人们词的某种组合是否正确在于它是否符合语言的语法，也就是那些被认为正确表达人们想要表达的意思的公认型式。句子依次连在一起组成为段落（通常包含了句子的综合意思），而每一段都能表达更多的内容。在小说中，若干段组成一章，而若干章组成一本书，一本书一般来说是完整的，与其他的书是相对独立的。

2.2 字 符 集

在Fortran 90中，基本元素或称字符集是由字母、数字、下横线和专用字符组成的。字母是英文字母表中26个大小写字母；数字是10个阿拉伯数字：0 1 2 3 4 5 6 7 8 9；下横线为_；下横线用作名字之间的分隔字符；21个专用字符由表1列出。在Fortran 90的语法中，大小写字母是等价的；只有用其作为字符串的一部分时才能区别开来。在本书中，语法上有效的字符一般总是写成大写字母。通常把字母、数字和下横线看成字母数字字符。处理系统也必须提供给用户一种能力，使其能指明另外一些字符，如法语中的重读字母。有了指明这样字符的能力将会使非英语国家编写和维护程序更方便些。

表1 Fortran 90语言的专用字符

字 符	名 称	字 符	名 称
	空格	:	冒号
=	等号	!	叹号
+	加号	"	引号
-	减号	%	百分号
*	星号	&	与号
/	斜杠	;	分号
(	左括号	<	小于号
)	右括号	>	大于号
,	逗号	?	问号
.	小数点或句号	$	美元符号
'	撇号		

在本节和下面的章节中，我们将进一步引出与自然语言的类比。Fortran 90 的信息单位是词汇记号，它相当于一个单词或一个标点符号。相邻的记号通常是由一个空格或由行结束分开，但一个专用字符或一对专用字符的作用可以看作是一个定界符，类似文章中的标点符号。记号序列形成语句，相当于句子。像句子一样的语句，可以连接形成像段落一样的比较大的单位。在 Fortran 90 中，这些是众所周知的程序单元，可以由此构造一个程序。程序形成了一套完整的计算机指令，用来完成一系列特定的操作。一个最简单的程序可以只有几条语句，但是，现在 10 万多条语句的程序已相当普遍了。

2.3 记　号

在 Fortran 90 的上下文中，字母数字字符（字母、下横线和数字）组合成为具有一个或多个含义的序列。例如，序列 999 的一种含义是数学上的一个常数。下面还将遇到这种类型序列的其他解释。同样地作为一种可能的解释，序列 DATA 可以表示为一个变量，可以把日历上的日期赋给它。

专用字符用来分隔这些序列，并具有各种含义。我们将会看到在 X * Y 中“*”号表示乘法操作，同时它还有多种不同的解释。

字母数字或专用字符的基本有效的序列称为记号，它包括标号、关键字、名字、常数、运算符和定界符。例如，表达式 X * Y 包含有三个记号 X、*、Y。除了在字符串中或记号内，空格可以任意使用，以便改进程序的外形。变量 DATA 不能写成 D A T A，而 X * Y 在语法上相当于 X * Y。在这样的上下文中，多个空格在语法上相当于一个空格。

一个名字、常数、或记号必须与相邻关键字、名字、常数或标号、同一个或多个空格或行结束分开。例如：

```
   REAL X
   READ 10
30 DO K = 1,3
```

上例中，在 REAL、READ、30 和 DO 之后要求有空格。然而一些成对的关键字，如 ELSE IF，不要求用空格分开。类似的情况，一些关键字可以用空格分开；例如，ELSEWHERE 可以写成为 ELSE WHERE。在正文中我们不用这些可能的形式，但在附录 B 的语句总结中给出了确切的规则。

2.4 源程序的形式

Fortran 90 开始使用新的源程序书写形式，它适宜在终端上使用。组成源程序的语句按行书写。每一行最多可包含 132 个字符，而通常一行只有一条语句。因为前导空格是无意义的，所以可以在第一列，或者在用户选定的任意位置上开始书写语句。因此，一条语句可以写成下列形式：

```
X=(-Y+ROOT_OF_DISCRIMINANT)/(2.0*A)
```

为了在程序中增加一些有关的适当的说明，Fortran 90 允许在任何行上写注释，为此只需写在感叹号“!”之后，例如，

```
X=Y/A-B          ! Solve the linear equation
```

任何注释总是延伸到源程序行的结束，同时可以包含处理系统有关的字符（不限于Fortran 字符集）。其第一个非空格字符是感叹号或只包含空格的任何行都纯属是注释行，注释行不参加程序的编译。这样的注释行可出现在程序单元内的任何地方，包括第一条语句之前（但不能出现在最后一个程序单元之后）。允许字符上下文中含有"!"，所以在此情况下，"!" 不能开始一个注释行。在其它情况下是可以的。

这是因为，比较长的语句可能在一行中的 132 列中写不完，所以允许有另外 39 个续行。这种续行标记是 "&" 字符，它写在每一行的行尾，表示其后面有续行。因此，本节的第一条语句（相当于间隔）可以这样写：

```
X =                                  &
  (-Y + ROOT_OF_DISCRIMINANT)  &
  /(2.0*A)
```

在非注释行上，如果 & 是某一行的最后一个非空格字符或在注释符号 ! 前的最后一个非空格字符，这条语句从 & 之前的字符开始向下继续。在正常情况下，语句继续到下一非注释行的第一个字符，但是，若下一非注释行的第一个非空格字符是 &，则继续到 & 之后的字符。例如，上述语句可以写成下列形式：

```
X =                                  &
 &(-Y + ROOT_OF_DISCRIMINANT)/(2.0*A)
```

特别是，若一个标记符不被包含在一行的结尾，则在下一非注释行的第一个非空格字符必须是 &，之后紧跟有标记符的下余部分。

允许注释包含任何字符（包括 & 在内），它们不是连续的，因为一个尾随的 & 应该被看成是注释的一部分。因此注释行可自由地插在续行之间而且不受 39 行的限制。

在一个字符上下文中（参见 2.6.4 节），续行必须是从没有尾随注释的行到带有前导 & 符号的行，这是因为 ! 和 & 两个符号都允许出现在字符上下文中和注释中。在本书中，通常将 & 符号对齐，以便提高程序的可读性。

当连续写短语句时，在一行上写几个语句是很方便的。在这种情况下，分号 ";" 用来作为语句的分隔符，例如：

```
A = 0; B = 0; C = 0
```

由于注释总是要延伸到行尾，所以不能在同一行的不同语句之间插入注释。从理论上讲，可以将很长的语句一条接一条地写，每行长 132 个字符，用分号来适当分开单个的语句。实际上，这样的程序可读性差。在不重要的情况下，才可以使用多个语句，就像上面的例子表示的那样。

任何 Fortran 90 语句都可以带标号，目的是为了能标识它。对某些语句其标号是必须的。语句标号必须放在语句之前，并且被看作为记号。标号必须是 0 ～ 5 位的数字，其中至少有一位数不为 0 。有标号语句的例子是

```
100 CONTINUE
```

前导空格在区别标号时没有意义。例如，10 和 010 是等价的。

2.5 类型的概念

在 Fortran 90 中，可以定义和使用各种不同类型的数据。例如，在程序中可以有数值 10，把它赋给 *I* 表示的整型标量。10 和 *I* 都是整型；10 是固定的或称常数值，而 *I* 则是可被赋予其它值的整型变量。例如，整型表达式 *I* +10 也是有效的。

数据类型由数据值的集合、表示那些值的方法，以及所允许对它们施加的运算的集合组成。对于整型数据类型，值为…，－3，－2，－1，0，1，2，3，…，其上下界取决于所使用的计算机系统和整数的种类。像这样一些记号是文字常数，而每一数据类型都有自己的表达方式。可以建立有名的标量如同 *I*。在程序的运行过程中，*I* 的值可以变成任何确定的值或变成无定义的，也就是没有预测值。对整型的运算就是通常的算术运算，可以写成为 *I* +10 或 *I* － 3 并且得到预测的结果。也可以建立一个有常数，在程序给定的执行期间，它们具有不变的值。

上面提到的那些特点与 Fortran 90 的所有数据类型都有关，且将在这一章和下一章中详细说明。Fortran 语言本身包含了五种数据类型，认为它们是存在的，而且都属于固有数据类型，其文字型常数是下一节要讨论的内容。在每一固有类型中，有缺省类别和其他类别的依赖处理系统的数。每一类别都与一个整数发生联系，此整数则称为类型类别参数。这是标识和区别可用的各种类别的一种方法。此外，可以在固有类型数据集合的基础上定义其它的数据类型，这些统称为派生数据类型。能够根据程序员的兴趣定义数据类型（例如矩阵、几何形、表、区间数），这是语言的强大功能，它允许数据高度抽象，即可以定义并使用数据对象而不需要考虑它们在计算机内部的实际表示。

2.6 固有类型的文字常数

固有的数据类型分为两类：第一类包含三种数值类型，它们主要用于数值计算——整型、实型和复型；第二类包含两个非数值类型，它们用于文字处理和控制——字符型和逻辑型。数值类型同普通的算术运算符号（如＋或－）连接在一起使用，这些将在第三章介绍。非数值型同各自的一套专用运算符一起使用，例如，字符数据可以并置连接。这些也将在第三章中叙述。现在仅对五种固有数据类型的文字常数分别加以说明。

2.6.1 整型文字常数

文字常数的第一类是整型文字常数，简单地说，缺省类别是带有符号或无符号的整数值，例如：

```
1
0
473
+56
-101
21_SHORT
1976354279568241_8
```

其中，SHORT 是一个标量正整型有名常数，其值必须是整型的有效类别类型参数。

缺省整数的范围不是在语言中确定的，但在 *n* 位（二进制位）组成一个字的计算机

上，通常是在-2^{n-1}～$+2^{n-1}-1$的范围内。因此，在16位的计算机上，整数范围通常是－32768～＋32767。

为了确保计算机内数的范围够用，就要通过给类别参数值来说明整数的类别。这最好由命名整常数来实现。如要求范围从－999999到＋999999，通过语句把K6作为有适当值的常数而建立起来，后面将详细阐明。

```
INTEGER, PARAMETER :: K6=SELECTED_INT_KIND(6)
```

并因此以常数形式被利用

```
-1234567_K6
+1_K6
2_K6
```

其中，SELECTED_INT_KIND(6)是内部查询函数的调用，它返回产生范围在－999999到＋999999的最小类别参数值。

在一个指定的处理系统上，可认为所需的类别值是3。在此种情况下，第一个常数可以写成－1234567_3。

但这种形式缺少可移植性。如果我们将程序移植到另一个处理系统上，则有可能不支持这种特殊的值，或者相应于不同的范围。

我们希望许多实现都使用类别，这种值指出由一个值占据的存储字节的数目，但Fortran 90标准有更大的灵活性。例如，一个处理系统可能只有4个字节整数的硬件，然而用这种硬件支持类别值1，2和4（使硬件具有1，2，4字节整数的处理系统放宽可移植性）。新标准只是指明对于处理系统上的任意两个K和L值，如果K＜L，则K的范围就小于等于L的范围。类别值永远不是负的。

在一个给定的处理系统上，一给定数据类型的类型类别参数值可由KIND内部函数得到（参见8.2节）。

```
KIND(1)      为缺省值
KIND(2_K6)   为例子
```

一个给定实体的实际范围可以从另一个函数里得到，例如，

```
RANGE (2_K6)
```

在这种情况下，该函数只少返回一个值6。

除了十进制系统的通常的整数外，对某些应用，能以二进制、八进制、十六进制的形式表示正整数是很方便的。在Fortran 90中，这种形式的无符号常数存在，而要表示它们就要象下面例中所说明的那样：

```
binary (base 2):          B'01100110'
octal (base 8):           O'076543'
hexadecimal (base 16):    Z'10FA'
```

在16进制形式中，字母A到F表示超过9的值；也可以使用小写形式。定界符可以是括号或撇号，例如下列成对的定界符是合法的。例如，

```
(…)
/…/
(/…/)
```

使用常数的这些形式，仅限于在DATA语句中它们作为隐式整数出现时。二进制、八进制、十六进制常数作为数字串，不带前导字母和定界符，可以出现在内部文件或外部文件中。

2.6.2 实型文字常数

文字常数的第二个类型是实型文字常数。缺省类别是浮点形式，它由部分或全部的浮点形式构成：带符号或不带符号的整数部分、十进制小数点、小数部分和带有符号或不带符号的指数部分。整数部分和小数部分二者至少有一个存在，指数部分可以不存在或由字母E的后面再跟一个带符号或不带符号的整数组成。实型数中至少应有十进制小数点和指数部分的两者之一。例如：

$-1012.6E-11$ 指的是 -1012.6×10^{-11}

下列形式也是合法的

```
1.
-0.12
23E - 1
3.14159
5678.00
```

缺省的实型文字常数表示数学上实数的子集，而标准既不规定指数的允许范围也不规定由处理系统所能表示的有效位数。通常实数值的范围大约是 $10^{-38}\sim10^{+38}$ 的7位有效数字。

为了能得到理想的范围和有效数，要求说明类别参数值。例如：

```
INTEGER, PARAMETER ::                    &
        LONG=SELECTED_REAL_KIND(9,99)
```

能保证常数

```
1.7_LONG
12.3456789E30_LONG
```

只少有9位有效小数的精度和 10^{-99} 到 10^{+99} 的指数范围。

就整数而言，我们希望许多实现使用类别值，以指出由一个值所占存储的字节数，但标准允许有更大的灵活性。它只指明，对一个处理系统上的两个值K和L，如果K＜L，则对K类别的精度就小于或等于L类别的精度。类别值不能是负数。如果已知所期望的类别值，则可以直接使用。例如，

```
1.7_4
```

但其产生的代码就难于移植。

处理系统只少提供一个具有比缺省类别更精确的实型数的表示，而这第二种表示也可以被指明为DOUBLE PRECISION。我们暂缓叙述这另外一种，把过时的语法放在第11.4.2节内介绍。

对实数值来说类别函数KIND是有效的。此外，有两种可利用的询问函数，它们能分别返回一个给定实数实体的实际精度和范围（见8.7.2），因此函数

```
PRECISION(1.7_LONG)
```

的值至少是9，而