HOPE

FORTRAN 90
程序员编程指南

Fortran 77的最新标准版本。具有许多杰出的新特性，可以帮助程序员编写高效、可移植和可维护的程序。

沈 彬 编译

北京希望电脑公司

FORTRAN 90 程序员编程指南

—Fortran77的最新标准版本

沈 彬 编译

北京希望电脑公司

一九九一年十月

前　言

Fortran 90 是 Fortran 语言的最新标准版本，它具有许多杰出的新特性，可以帮助程序员编写高效的、可移植的和可维护的程序。它是由 Fortran 77 发展而来的，并且完全包含了 Fortran 77。本书除包含了 Fortran 程序的所有基本特性（如 Fortran 程序格式，数据类型，表达式和赋值，输入／输出）外，主要集中讲述 Fortran 90 的新特性，如数组处理特性和使用新的全局数据特性（模块）对数据类型进行控制等。利用这些新特性，程序员可获得完成特定程序任务的最佳解决方法。

本章共分十章，第一章讲述了编写 Fortran 程序需要的所有基本特性。第二至第八章含有控制结构、过程、数组、字符串、数据结构、派生类型、模块、递归和指针变量的详细讲述。第九章介绍了输入／输出功能。第十章讲述了编写新程序不应使用的过时特性。附录 A 和 B 给出了语言的完整语法说明和各种内部函数的说明。

本书的问世得到了中国科学院希望高级电脑技术公司的大力支持，在此表示衷心的感谢!

1991 年 8 月

目　录

第一章　Fortran　90 编程简介

学习程序设计语言的最佳方法是立即开始阅读和编写程序．如果有计算机，可编写和运行本章的简单程序例子．你需要简要的指导来了解如何输入和运行程序．

§1.1　计算和打印和程序

由于计算机善于算术运算，而 Fortran 是设计用来进行数值运算的，因此学习 Fortran　90 首先应从算术运算开始．本节说明如何编写用来计算和打印答案的程序．

§1.1.1　简单计算

头一个例子是打印术和结果的程序:

```
PROGRAM CALCULATION_1
   PRINT *, 84 + 13
END PROGRAM CALCULATION_1
```

程序 CALCULATION_1 告诉计算机把数值 84 和 13 加在一起，然后打印和 97．当告知计算机运行 CALCULATION_1 时，执行过程如下：把两个数加在一起并打印其和．执行情况如下所示:

```
RUN CALCULATION_1
 97
```

§1.1.2　缺省打印格式

关键字 PRINT 后面的星号（*）告诉计算机，程序员没有为打印的答案指定严格的格式．这时，Fortran 系统使用缺省格式，也称之为表式格式（§9.8.21），它可满足于大多数情况．Fortran 标准允许缺省格式设计的某些灵活性，因此上例的输出可能会稍有不同．

§1.1.3　打印信息

如果你想让计算机打印自己所指定的电传字符，可用双引号或单引号把它们括起来，如程序 QUOTES 所示．输出并不打印引号．

```
PROGRAM QUOTES
```

```
   PRINT *, "84 + 13"
END PROGRAM QUOTES
```

```
RUN QUOTES
 84 + 13
```

在 Fortran 程序中，用引号括起的电传字符序列称作字符串。字符串可含有字母字符，数字字符，以及诸如标点和算术符号的特殊字符。

打印文字字符和计算的数值会产生如下的易读输出。

```
PROGRAM CALCULATION_1_V2
   PRINT *, "84 + 13 =", 84 + 13
END PROGRAM CALCULATION_1_V2
```

```
RUN CALCULATION_1_V2

 84 + 13 =  97
```

在程序 CALCULATION__1__V2，PRINT 语句列表中有两项，字符常量"84+13＝"按其书写形式打印出来（不带定界的引号），算术表达式先计算后打印。尽管这两个项看起来类似，但对它们的处理却不同。括号中的字符串按原样打印出来，而未带括号的同样表达式却要求值并打印其和。逗号用来隔开 PRINT 语句中的项。

§1.1.4 PROGRAM 语句

每个 Fortran 程序可由 PROGRAM 语句起始。它由关键字 PROGRAM 后跟程序员选择的程序名组成。程序名必须起始于字母，可含多达 31 个的字母、数字和下划线。程序名 CALCULATION__1 只是为了便于阅读。

§1.1.5 END 语句

END 语句起始于关键字 END。它可后跟以关键字 PROGRAM（同样后跟双程序名）。每个 Fortran 程序必须以 END 语句作为最后一条语句。

补充注释：在各个 END 语句上包含关键字 PROGRAM 和程序名。

§1.1.6 练习

1、编写并运行打印你的名字的程序。

2、编写并运行计算整数 1 至 9 总和的程序，前加以简短的信息来说明输出。

3、说明下述程序的输出结果：

```
PROGRAM SIMPLE
   PRINT *, 1, "AND", 1, "EQUALS", 1 + 1
END PROGRAM SIMPLE
```

§1.2 内部数据类型

Fortran 中有五种内部数据类型：整数、实数、复数、逻辑型和字符．每种数据类型有一组该类型所代表的值及对这些值进行的操作．我们已经看到了其中两种数据类型的例子．“84+13”（带引号）是字符串常量，84+13 是值为整数类型的表达式．

下面小节说明每种内部类型及这些类型的常量的 Fortran 书写方法．

§1.2.1 整数类型

整数类型用来表示为全数字的值．在 Fortran 中，整数常量按常见形式书写．整数常量是只含有数字 0 到 9 的串，要后跟以下划线（_）和 1.2.9 中说明的无符号整数类常量．下面是整数常量例子．

```
23   0   1234567   42_1   42_SHORT
```

带符号整数常量是前面可带有+或−号的整数常量．带符号整数常量可看作是受限类表达式，但常常可用在整数常量使用的地方．

各个 Fortran 系统必须至少有一个整数种类，而整数种类号却随着计算机的不同而异．多数 Fortran 系统有几个整数种类，对应于可表达值的不同范围而有不同的种类数．

§1.2.2 实数类型

Fortran 中有两种格式的实数常量．头一种称作位置格式，因为每个数字的位置 1 受限于其相对于小数点的位置．实数常量的位置格式由整数，小数点和小数部分组成，可后跟以下划线和类参数．假定 DOUBLE 和 QUAD 为 Fortran 系统上允许实种类的整数常量名（见§1.2.9 节），下面是以位置格式书写的实数常量．

```
13.5    0.1234567    123.45678    00.30_DOUBLE    3.0
3.      12345.       .0           .1234567_QUAD
```

以位置格式书写的实数常量可能在小数点左边没有数字，或在小数点右边没有数字，但小数点本身并不是合法的实常量．

实数的指数形式由以位置格式书写的整数或实数，后跟字母 E 和带符号整数（无种类参数）及任选的下划线和种类参数组成．字母 E 读作“乘以 10 的幂”，E 后面的整数是乘以 E 前面数值的 10 的幂次．指数形式对书写较大或较小数值很有用．例如，2.3 E5 代表 2.3 乘以 10 的 5 次幂，2.3×10^5 或 2.3×100，000＝230，000．整数幂可在其前面含有

减号或加号，如实数常量 2.3E−5，为 2.5×10^{-5} 或 $2.3\times0.00001=0.000023$。另一个例子是 1E9_DOUBLE，它为带有种类参数 DOUBLE 的 10 亿，1E−3 为千分之一。

各个 Fortran 系统必须具有至少两个实数种类，而这些种类的种类号却随着计算机的不同而异。多数 Fortran 系统有几个实数种类，对应于不同的精度和值的范围而有不同的种类数。

§1.2.3 复数类型

Fortran 的复数类型用来表示数学复数，它由两个实数组成，通常写作 a+bi。头一个实数称作实部，第二个实数称作复数的虚部。在 Fortran 中，复数常量写作两个整数或实数，由逗号隔开并用括号括起。复数常量的例子为:

```
(1, −1)
(3.14_DOUBLE, −7)
(−1.0, 3.1E−27_QUAD)
```

如果两部分为整数类型，则复数常量的种类参数为缺省的复数种类。如果一部分为实数，另一部分为实数，则复数常量的种类参数为实数种类。如果两部分均为实数，则复数常量的种类参数与具有最大精度的部分的种类参数相同。

§1.2.4 算术操作符

可用来组合两个数值的操作符包括+，−，*，/和**。除**外，这些符号具有其通常的意义，用来标明加、减、乘和除。两个星号表明指数，即 2**4 的值为 16。符号+，−可用作单目操作符，来分别表明同一和取反操作。

整除总产生一个整数结果，它由去除算术结果的小数部分而得到。例如，由于 23/2 的运算结果为 11，5，故 Fortran 算术表达式

```
23.0 / 2.0
```

的值为 11.5，而表达式值

```
23 / 2
```

为两个整数常量的商，值为 11。类似地，两个表达式

```
−23 / 2    23 / −2
```

的值为−11。

§1.2.5 关系操作符

数值可用关系操作数来比较．关系操作符的两种格式如表 1–1 所示：

表 1–1 关系操作符

<	.LT.	小于
<=	.LE.	小于等于
==	.EQ.	等于
/=	.NE.	不等于
>=	.GE.	大于等于
>	.GT.	大于

复数值只能用关系操作符==和/=来比较．而由于舍入错误，使用==或/=操作符来比较实数或复数值是不良的编程习惯．

关系操作符的结果为实数类型．

§1.2.6 混合模式表达式

从数学上看，整数为实数的子集而实数为复数的子集．因此可组合两个数值，即便它们的 Fortran 类型不同．数值操作符的两个操作符不必具有同样的数据类型；当它们不同时，则在执行操作之前，将其中之一的类型转换成另一个的类型．如果一个为整数类型，另一个为实数类型，则把整数转换成实数类型；如果一个为整数类型，另一个为复数类型，则把整数转换成复数类型．例如，表达式

```
23.0 / 2
```

的值为 11.5，由于整数 2 被转换成实数值，然后再执行两个实数值的除．

数值操作符的两个操作数也可具有不同的种类参数值．这时，如果两个操作符具有同样的类型，或一个为实数，一个为复数，结果具有带有较大精度的操作数的种类参数．例如，如果种类 5 比种类 2 的精度高，则

```
1__2+3__5
```

的值为 4，种类参数 5．假定种类 4，比种类 2 或种类 3 的精度高，则

```
1.1__4+ (2.2-2, 3.3-3)
```

的值为 3.3+3.3i，种类参数为 4。如果一个操作符为整数类型，另一个为实数类型，而另一个为实数或复数，则结果的种类参数为实数或复数操作数的种类参数。

§1.2.7 逻辑类型

Fortran 的逻辑类型用来表示两个真值"真"和"假"。逻辑常量或为.TRUE.，或为.FALSE.。

可用来组合逻辑值的操作符为.NOT.，.AND.，.OR.，.EQV.，和.NEQV。它们都为二目操作符，只有单目操作符.NOT.例外。每个逻辑操作符的结果值如表 1–2 所示：

表 1–2 逻辑操作符的值

x_1	x_2	.NOT.x_2	x_1.AND.x_2	x_1.OR.x_2	x_1.EQV.x_2	x_1.NEQV.x_2
true	true	false	true	true	true	false
true	false	true	false	true	false	true
false	true	false	false	true	false	true
false	false	true	false	false	true	false

给出一个简单的例子：

.FALSE. .EQV. .FALSE.

的值为真。

§1.2.8 字符类型

字符类型用来表示字符串。字符常量的格式为一系列由边界字符双引号和单引号括起的字符序列。如果定界字符出现在字符串中，则它由两个不带空格的定界符来表示。字符常量可前带以种类参数和下划线。注意，字符常量的种类参数必须在常量之前，而不是在其后，这与其它类型的种类参数情况不同。下面是字符常量的例子。

```
"Joan"
ASCII_'John Q. Public'
"Don't tread on me."
'He said, "Don''t tread on me."'
```

只有一个产生字符结果的操作符，即并置操作符。所用的符号为／／，二元操作符的结果为一字符串，它由第一个串后跟第二个串组成。例如，"John Q."／／"public"的值为串"John Q.public"。注意，句点后无空格。两个操作数的种类参数必须相同。

关系操作符（1.2.5）可用来比较字符值。

§1.2.9 种类参数

种类参数提供了对每种内部数据类型的不同机器表示选择的参数化方式。如果程序员小心的话，这就提供了对可移植数值精度和范围进行选择的机制。对字符数据类型，它允许使用多个字符集，如日文、汉文和化学符号。

每种内部数据类型（整数，实数，复数，逻辑和字符）具有一个称作种类参数的参数。种类参数用来表明特定数据类型的机器表示。例如，一种实现可具有三种实数种类，通常称作单精度，双精度和四倍精度。

种类参数为整数。这些数是处理器有关的。因此种类参 1，2 和 4 可为单精度，双精度和四倍精度，而在不同的系统上，相应种类参数可为 4，8 和 16。唯一的需求是必须至少有两个实数和复数种类，以表示缺省的实数精度和双精度，以及至少一个用于整数，逻辑和字符数据类型的种类。注意，种类参数的值与精度或范围的十进制数字数无关。

你需要查看 Fortran 手册，以确定每种类型的可用种类参数及每种类型的缺省种类参数。种类参数在所有情况下均为任选的，因此总可为你的应用程序使用缺省种类。

内部函数 SELECTED_INT_KIND 和 SELECTED_REAL_KIND 可用来选择变量或命名常量（§1.3.9）的适当种类。这两个函数提供了使程序成为可移植程序的方法，这里需要用某种指定精度来计算的值可使用一台机器上的单精度，而又需要另一台机器上的双精度。详见§1.5.1 节。

对逻辑数据类型，一种实现为具有按每位排列的逻辑值特性表示，但它也由于寻址和指令集类型的原因而按字节排列逻辑值。这些表示可由程序员通过指定种类参数值来选择。

当种类参数为另一个常量的一部分时，它可为整数常量或命名整数常量（参数）。在整数，实数和逻辑常量中，它在后面跟有下划线字符。

```
12345_4
1.345_2
```

在字符常量中，它出现在前面并后跟下划线。

```
ASCII_"abcde"
GREEK_"αβγδε"
```

复数常量的种类由两个实数成份的种类来标明。

§1.2.10 练习

1、将下述实数类型的数值从位置格式转换成指数格式。

48.2613　　.00241__4　　38499.0

2、将下述实数类型的数值从指数格式转换成位置格式.

```
9.503E2        4.1679E+10_DOUBLE        2.881E-5
-4.421E2       -5.81E-2_8               7.000001E0
```

3、编写打印.1+.2__.3+…+.9 和的程序.

4、确定你的计算机系统上比缺省实数种类精度高的一种实数种类号.

5、编写打印复数和的程序

(.1+.1i) + (.2+.2i) + (.3+.3i) +…+ (.9+.9i)

6、编写打印下述每个表达式逻辑值的程序:

```
2 > 3
2 < 3
.1 + .1 == .2
.5 + .5 /= 1.0
```

7、编写计算并打印你的所有名字（即姓和名）并置的程序.

§1.3 变量和输入

编写计算机程序较之用笔或计算器进行求解的优点之一便是当再次出现后类问题时，已编写的程序可重用. 变量的使用为这种重用提供了灵活性. §1.1 节中的程序让计算机对出现在 PRINT 语句之前的数值常量进行所标明的算术操作. 本节的头一个样本程序 ADD__2 用来求两个输入整数的和. 相加的数并不出现在程序中，而用两个变量 X 和 Y 来保持输入的两个值. 由于 Fortran 语句可对变量和常量进行操作，因此可计算和打印其和. 头一个样本运行表明该程序可用来求数 84 和 13 的和，这是由 §1.1 节中的 CALCULATION__1 程序计算的.

```
PROGRAM ADD_2
   INTEGER :: X, Y
   READ *, X
   PRINT *, "Input data  X:", X
   READ *, Y
   PRINT *, "Input data  Y:", Y
   PRINT *, "X + Y =", X + Y
END PROGRAM ADD_2

RUN ADD_2
```

```
Input data  X:  84
Input data  Y:  13
X + Y =  97
```

在说明变量 X 和 Y 将保存整数值后，程序 ADD__2 告诉计算机从输入设备读一个数，称之为 X，然后再读入另一个数，称之为 Y，最后打印 X+Y 的值。两个附加的 PRINT 语句用来回显完成程序 ADD__2 的输入数据值。在该程序执行期间，作为 X 和 Y 值的两个数必须提供给计算机，否则计算机不能运行。

§1.3.1 变量的说明

在 PROGRAM 语句和程序的可执行部分之间出现类型语句。每个说明由指定 Fortran 内部类型的关键字组成，后跟两个冒号或由逗号隔开的变量名表。例如，程序 AD__2 使用类型说明。

```
INTEGER::X, Y
```

对应于在§1.2 节中引入的整数、实数、复数、逻辑和字符常量，有整数、实数、复数、逻辑和字符变量。例如，变量 Q、T、K 为程序中的实数变量，N，B 为整数变量，说明如下:

```
PEAL::Q, T, K
INTEGER::N, B
```

在表示数据类型的关键字之后，变量可说明成具有特定的种类参数，这是由 KIND= 后跟以种类参数值得到的。例如，如果额外精度的种类参数为 2，则变量 DPQ，X 和 LONG 可用下述说明来说明成额外精度实数:

```
REAL (KIND=2) ::DPQ, X, LONG
```

对字符变量，关键字 CHARACTER 应后跟以 LEN= 和一个整数并加上括号来标明字符串中的字符数。如果字符变量具有非缺省的种类参数，可在同样的圆括号内指定。如果变量 NAME 为 20 个字符的串，可说明如下:

```
CHARACTER (LEN=20) ::NAME
```

此外，如果变量名具有种类 KANJI，假定它为整数常量名，则说明可为:

```
CHARACTER (LEN=20, KIND=KANJI) ::NAME
```

补充注释：作为一种良好的编程习惯，Fortran程序中使用的各个变量应列表于类型说明中。

可在说明时为变量提供初值．例如，变量 COUNT 可说明成整数类型，并由语句：

INTEGRE::COUNT=0

置为 0．变量 A，B，C 说明成实数，并置为初值 1.1，2.2 和 3.3，如下语句所示：

REAL::A=1.1，B=2.2，C=3.3

按这种方式初始化的变量值可在程序执行期间改变．

§1.3.2 隐含类型

以前，Fortran 语言没有类型说明．变量是由基于其名字头一个字母的隐含类型来指定类型的．只允许实数和整数变量．名字起始于字母 I–N 的变量为整数类型．所有其它就量为实数类型．这种隐含类型目前仍起作用．但可通过语句：

IMPLICIT NONE

消除隐含类型．

补充注释：每个程序和过程应含有

IMPLICIT NONE

语句来消除隐含类型．

§1.3.3 提供输入数据

变量 X 和 Y 的两个输入值 84 和 13 并不出现在计算机程序中．它们是由用户敲入的．要运行程序，可用命令“RUN”来编译和运行 Fortran 程序，从与该程序同名的文件中读取输入，但附加以字符“–IN”．这样，命令

RUN ADD__2

编译并运行程序 ADD__2 并从文件 ADD__2__IN 中读取输入数据．对用来运行

ADD__2 的 Fortran 系统，命名为 ADD__2__IN 的输入文件必须使用敲入程序的同一编辑器来准备。该文件含有两行:

```
84
13
```

在某些 Fortran 系统上，输入数据在含有程序的文件的末端敲入。在另一些系统上，数据在执行期间由键盘敲入。你得弄清你的 Fortran 系统使用的方法。

§1.3.4 输入数据的回显

在 Fortran 及大多数其它程序设计语言中，对于用户的良好编程习惯是使用 PRINT 语句来提供输入数据的回显，这样输出含有计算所用变量的记录。程序 ADD__2 中每条 READ 语句后跟以刚读入数据的回显。

补充注释: 回显所有输入数据是良好的编程习惯。但有些情况下遵循该规则又是不现实的，如大量输入数据的情况。

§1.3.5 用不同的数据重运行程序

程序 ADD__2 含有回显，其重要性在于演示程序重新运行时的不同输入数据。输入数据的回显有助于弄清程序出现的问题。输入回显的另一种重要用途在后面说明。现在表明 ADD__2 的另一次示例运行，冻改变程序本身，只改变输入数据。这时，数据文件 ADD__2__IN 中有下述两行:

```
4
7
```

```
RUN ADD_2

  Input data  X:  4
  Input data  Y:  7
  X + Y =  11
```

ADD__2 的最后一条 PRINT 语句引用变量 X 和 Y. 如打印结果所示，执行 PRINT 语句时的打印结果为字符串常量: "X+Y = "的值后跟以表达式 X+Y 的值。

程序 ADD__2__REALS 由程序 ADD__2 得到，只需在变量说明中把关键字 INTEGER 说明成 REAL，它为变量 X，Y 为实数类型. 程序 ADD__2__REALS 可用来进行实数求和。程序的执行也表明输入数据值可为负的。该示例执行的输入文件含有两行: