兼容Fortran语言

伦敦大学协会计算机顾问
阿·科林戴 著
魏绍贤 译

航空工业部六三一研究所资料室

一九八二年十一月

目录

前　言

一个标准化语言主要优点之一是它的兼容性，即与它所运行的计算机无关。当在编写可移植 Fortran 程序有了一些经验之后，笔者很快认识到：仅遵照 Fortran 标准是不够的，构成兼容 Fortran 的子集与美国国家标准协会（ANSI）的标准 Fortran 有很大区别，需要予以单独描述。

通过在一个大学（它拥有几个不同类型的计算机，在那儿对其中一个计算机研制的程序可送入另一台用于运算过程的计算机）的工作，使笔者确信：有必要对这个兼容 Fortran 进行描述。当局部计算机须用不同厂商制造的设备来代替时，这种必要性变得更为迫切。

写本书的目的是试图在多个机器上运行程序方面所获的经验有益于读者。由于笔者的经验有限，所以在本书的观点难免有许多不足之处。然而，从经验看：在这一领域尽管这是些有局限性的看法，但总比没有要好得多。

阿·科林戴

1977·10

1. 绪　言

没有一种计称机语言能象Fortran 语言那样具有可兼容性。尽管如此，仅有很少程序设计者能利用这种可兼容性。编译程序的设计者一直在编制具有新特点的编译程序，制造厂家则很少给予指明，使用这些特点能够防止程序在一个竞争机器上运行。

本书叙述一个Fortran 方案（"兼容Fortran"或缩写"CF"），它可以接受大多数编译程序，用它编写的程序可在许多不同的机器上运行。

为对某些术语不产生误解，需在一开始就给以澄清。能为多种计称机所接受的语言叫做兼容语言。用这种语言编写的程序被叫做可移植程序（或可移动程序）。

假定读者口对一个机器来讲已具备丰富的Fortran 知识。又假定他不是一个关于Fortran 标准化或Fortran 方案的内行。

可移植性的优点

人们编写程序常常只使其在一个机器上好用，而不致虑移植它们。然而，好程序的消息传播很快。对某一种机器的程序，保守秘密，将是最笨的一着，因为这种机器终将淘汰，为新的与它不同的机器所代替。

当今是计称机网络时代，你可发现，程序在一个机器上进行调试，然后再沿线将程序送到大的处理主机上去运行，这一个优点，很诱惑，只有可移植程序才可能做到这点。

可移植性的主要优点是使你（或其他人——或两者）能将一

个程序移植到另一个机器上，而避免修改程序的问题。学习一个新的Fortran方案并遵守它的规则，对你来说似乎是件麻烦事，然而要知道修改程序可能是更加麻烦的事。要使程序在每一个新的计算机上运行，则程序须修改。如非兼容特点已深嵌入编码之中时，它可能是重构基本逻辑所需要的。无数程序技术工作者已经发现：没有可移植程序，象Alice，为了原地不动，将要求你跑得的确很快。

Fortran标准和Fortran 77

"为促进在各种自动数字处理系统上使用的…。程序具有高度的互换性之目的"（ANSI 1966a P7），已经起草了Fortran语言的标准。为什么Fortran具有如此的潜在兼容性，其主要的原因是有一个定义完整总义明确的标准存在。遗憾地是由于许多Fortran程序设计者并不了解这个标准规定了什么，所以本书大部分将用来描述这个标准的规定。

要了解Fortran标准化的历史，请看Muxworthy的文章（1972）。美国国家标准协会（ANSI）对用他们的文件X3.9—1966（ANSI 1966a）定义的最著名Fortran标准，美国标准Fortran负责。这个语言就是大家都熟悉的ANSI Fortran，或（按不严格的讲法），称之为Fortran IV。ANSI在标准化它们自己名称时不幸地迁到了麻烦。在一个时间它们是美国国家标准协会（ASA），后来为美利坚合众国标准化协会（USASI），因此，人们会发现参照"ASA Fortran"或"USASI Fortran"，其总义与参照"ANSI Fortran"即参照X3.9—1966所规定的语言是一样的。

在公布X3.9的同时，ANSI还颁布了另一标准。这一语言

被称做美国标准基本 Fortran，它由 ANSI 的 $X_{3.10}$—1966（ANSI 1966）所规定。这一语言为 $X_{3.9}$ 的一个子集。在权限方面相当于旧的 Fortran II。

ANSI 发表了两篇文章，针对已产生的各种解释的某些观点进行澄清（ANSI 1969、1971）。

欧洲计算机制造商协会（ECMA）已制定了一个 ECMA Fortran 标准，这个标准界于美国的两个标准之间（ECMA 1965）。国际标准化组织（ISO）制定了一个编入这三级 Fortran 的建议（ISO 1972）。

这两个 ANSI 标准，由于它们5年没有修订或重申继续延用，于1971年，从学术上作废。一个 ANSI 子委员会，X_{3J3} 已起草了一个称之为 Fortran 77 的新的 ANSI 标准。这个标准现在已被采用，它分为两级，较高一级几乎包括3套个 $X_{3.9}$，做为它的一个子集；然而它还包含大量的扩展（见下部分）。

针对兼容性而言，这些标准是有益的，但还不够。其中某些部分尚有待于解释。而许多编译程序在接受 ANSI Fortran 的同时，实际上还有更多的限制。如果要维护兼容性，那么就应避免以后的扩展。无论由一个编译程序允许扩展还是由一个如 Fortran 77 的新标准进行扩展。

即使这些标准（按它的大多数文字解释是否定兼容性的，然而仍有许多方面可能实现兼容性。一个这样的方面是字符操作（14节）。在这样的情况下兼容 Fortran 限制比这些标准少。

最接近兼容 Fortran 的标准是（并将在某一些时候内继续是）美国国家标准 Fortran。无论"Fortran"或"标准"本书都将给以叙述。参改这一标准本书中使用的术语尽可能是

这个标准中使用的术语。并将描述兼容 Fortran 和标准不同的那些地方。

在兼容性这个范围中，许多问题可在 Fortran 的初级教课书中找到根源，或是教师不了解这一标准，或是出于要介绍"有用"的扩展之好心。应极力劝告按程序设计教程教授标准 Fortran。（例如，见 Day 的文章（1972a）和磁带录象教程的附注释。）

因为兼容 Fortran 与标准 Fortran 靠得那样近，所以本书的大部分将用来教授这标准规定了什么。为此理由，最方便的方法是采用标准使用的术语表，即使它与通常使用的术语常有各种差别。不可能当每个术语出现时再告诉读者。也不可能到那时再做解释。但是通过索引查找这些术语的解释，经常的查阅查找直到读者熟悉标准术语。虽然本书企图阐明通用（非兼容的）Fortran 用法、Fortran 标准和兼容 Fortran 之间的不同，但是，本书不重复 Fortran 的全部基本规则。因为每一本初级教课书都讲述到它们，每一个编译程序都检验它们。

Fortran 77

一部分人认为，Fortran 的新标准，在想像中按书的要求应大刀阔斧删减。少数人认为，不应如此。既然已批准了这个新的标准，某些编译程序将为之编写，但并非个个编译程序都如此改变。兼容子集将仍是一个最低"公分母"，因为 Fortran 77 将是一个大"环"，可通过较小"环"的程序也应能通过这个大"环"。

然而问题出在 Fortran 77 与 X3.9 不能获得充足的兼容

性。兼容 FORTRAN 已被注意的地方是比 1966 年的标准有更多的限制。

环境

必须强调：一个程序的可移植性不在于遵守某些语法规则。一个程序可以对其操作的环境提出某些要求，达到这些要求的环境仅有很少机器可满足，而这可能严重地限制了它们的可移植性。

要求大量外部设备或要求大量专用设备的程序，很明显不可能是可移植的。然而一个程序需要保持实数具有最大的精度，它对它的环境的要求同样也是没有多少机器可以满足。同理，一个很大的程序不管它编写得如何好，也不可能是移植为。

FORTRAN 方言

不同的编译程序所认的 FORTRAN 稍有不同。因此不同的"方言"就出现了。方言的出现在两个方面引起麻烦。一方面，它比标准限制要少，以鼓励程序设计者去使用这些不超过一个编译程序具有的特点。而另一方面，它可比标准有更多的限制，以使有关编译程序将不接受可移植程序。所以兼容 FORTRAN 将需要的语言不同于标准中定义的语言。

在此要提一下，另一个目标是提供一个兼容 FORTRAN 方言。这就是在 Ryder (1974) 的文章中给予描述的 PFORT。一般说来，CF 比 PFORT 有更多的限制，所以它对编译程序所兼容的范围很广。

专门的编译程序在某些特殊地方将产生麻烦，本书对它不可能做详细地描述。这样做有很多理由。为了命名故障而改编编译程序，你可找到大量的运动目标。通过参数特殊型式的计

标机来解决问题是无效的，这是因为在特殊型式的计标机上使用的编译程序与一般机器的编译程序有差别。不命名特殊的编译程序的主要理由毫无疑问是我自己的知识短缺。由于我的知识缺陷，将一定表现为兼容 FORTRAN 不能具有它应具有的兼容性。然而，这样的尝试，尽管不完善，也是很有意义的。

对各种不同 FORTRAN 编译程序已做了一些比较。有兴趣可参阅 Stuart 的文章（1969），和 Muxworthy 与 Shearing 的文章（1970）。

虽然一般地命名编译程序是不可能的，但是已试图给出为什么增加限制和提出警告的理由。它们都以机器体系结构的术语或某些无命名的编译程序的习惯术语来表示。

一些人可能用下面观点来反对我，说 CF 本身就又是一个方言，不要再增加非标准 FORTRAN 的数目了。也有人可能用另外的观点来反对我，说方能钥匙就是另一种钥匙。虽然如此，但它可能大大地减少人们必须带着到处走的另外一种钥匙的数目。

2. 构造块

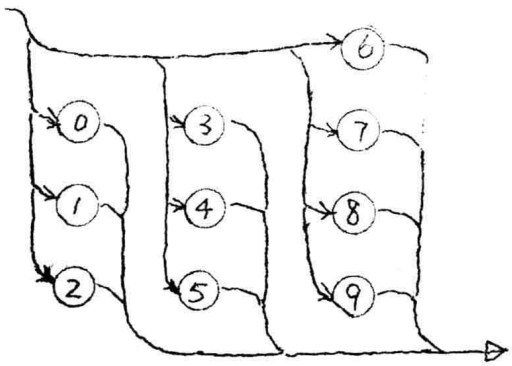
本章叙述一些兼容 FORTRAN 的基本部件，即字集符，符号名，常数和数组元素（包括下标的表示方法）。另一些表示法在第十一章叙述。

字符集

字符集由数字、字母和专用字符组成。在此有必要介绍一下用来指明允许格式的专用流程图。第一个流程图表示一个数

字是十个字符 0，1，2，3，4，5，6，7，8，或 9 的任何一个。

数字（digit）

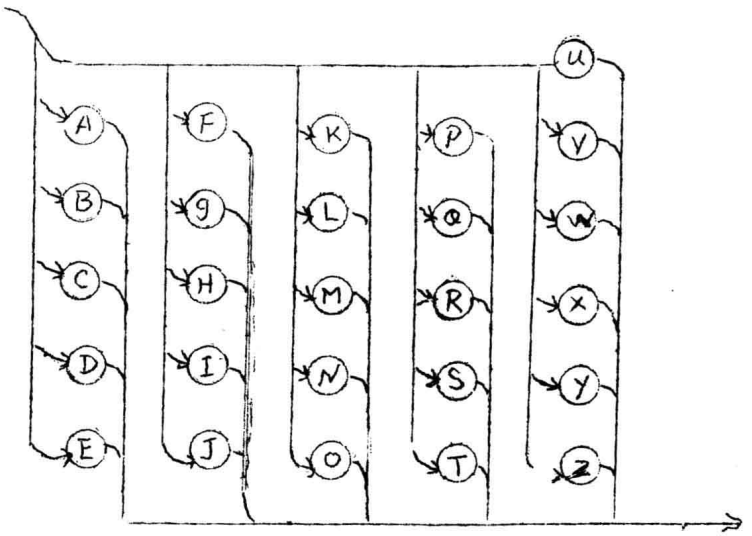


这种流程图象是一个“迷宫图”。从左上角进入，由右下角出去。而这些逻辑框像一个个“旋转栅门”，只能沿一个方向通过（按箭头指明的方向），并且仅通过那些要求给出的项目。圆形的逻辑框需要文字符号说明。矩形逻辑框需要一个在另一地方已定义的实体。路径可视为铁路线。要调转车头而反向通过尖形拐角是不允许的。有时需要用史下面的语言限制来修改这个图。笔者认为：Bell 实验室的 M·D·McIlroy 首先将这种图用到 Fortran 语言上，见 McIlroy 的文章(1974)

入口上的名字指明该流程图所规定的实体。通过上述数字流程图，仅能通过 10 个数字中要求给出的一个数字。

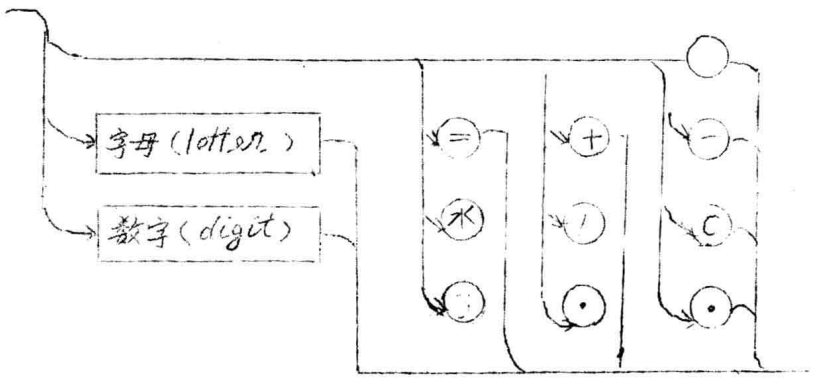
字母流程图可以用类似的方法给以定义。

字母（letter）



由于我们已定义了数字和字母，将它们和10个专用字符加在一起可定义全字符集。

字符（character）



要是穿过不含字符的逻辑框，就给出一个空白。

标准Fortran还含有常用符号（$），但这个标准没有指明如何在语言中进行使用。某些编译程序将它解释为一个额外的字母符。另一些编译程序用它来分隔同一行的两个语句。这个字符很难表示成穿孔卡的编码。由于有这些理由，所以应该避免使用它。

标准在下面三个地方允许使用非Fortran字符：

（1）在注释行当中，允许使用非Fortran字符。这些字符将引起有关转换成其他卡的编码问题。在注释中只准兼容Fortran字符中的那些字符。

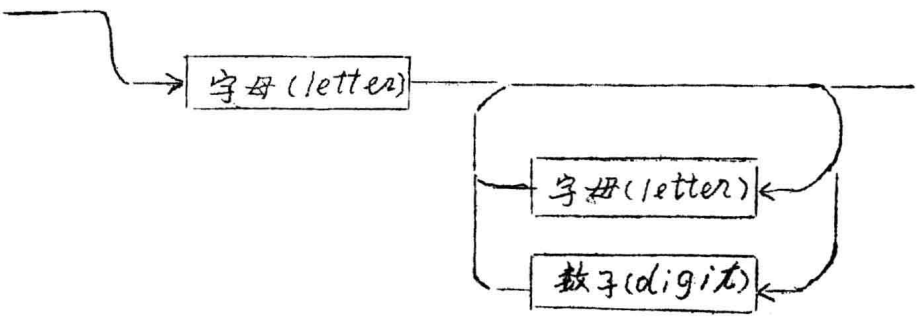
（2）在一个Hollerith数据（即在一个DATA语句或一个Fortran语句中）中允许使用非Fortran字符。卡片编码转换问题会再一次出现。与兼容Fortran字符集中的字符不同的字符，此处也应避免使用。

（3）在数据卡上，允许使用非Fortran字符。如果数据和程序一道进行转换，应该避免使用非Fortran字符。如果将数据卡用于在一个机器上的运行程序，而不用于其他机器；在数据中由于存在有非Fortran字符并不影响结果。

符号名

一个符号名是一个变量名，数组名，公用块名，语句函数名，函数名，子例行程序名或外部程序名。这样的一个符号名结构表示于下。

符号名（symbolic name）



限制：在符号名中不多于6个字符。

例如：A，I，X_2，Y58902，K2504

注意：符号名不能长于6个字符；即使某些编译程序允许8个字符，甚或31个字符。

标准允许把一个字符名在一定限度内用于同一程序段，可有一个以上的用途。例如：一个变量可以和一个COMMON（公用块）用同一个名字。依据标准，变量也可以和该程序段不涉及的内部函数用相同名字。而某些编译程序禁止这样做。

标准允许使用等同于FORTRAN的关键字（例如END或REWIND）的符号名。某些编译程序不允许这样做。尤其是已经知道编译程序处理的所有语句开头是字符FORMAT（像一个FORMAT语句），因此要禁止如下语句序列：

```
DIMENSION FORMAT(10)

FORMAT(1) = 0.0
```

为了更安全一些，对符号名规定如下：

（1）一个COMMON块在同一程序段中禁止用作任何别的用途。

（2）任何内部函数名或任何基本的外部函数名在任一程序

段禁止用作别的用途。

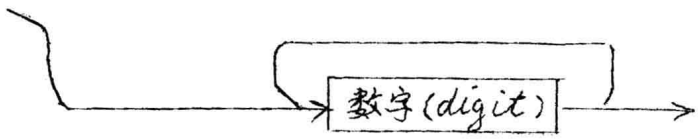
（3）一个符号名与一个 FORTRAN 关键字禁止相同。

常 数

这节只讨论常数的形式。关于常数值大小的限制见 13 章

整常数定义如下：

整常数（Integer constant）

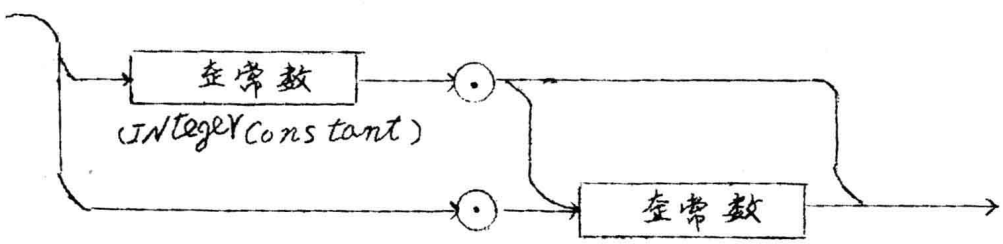


举例：1， 0， 17， 68458 。

注意：整常数这个术语（像标准所使用的）涉及一个无符号常数。这一点对实常数和双精度常数也一样。整常数允许有符号，在后面流程图中将专门标明。

实常数和双精度常数以标准中称为基本实常数的一个中间结构术语来定义。

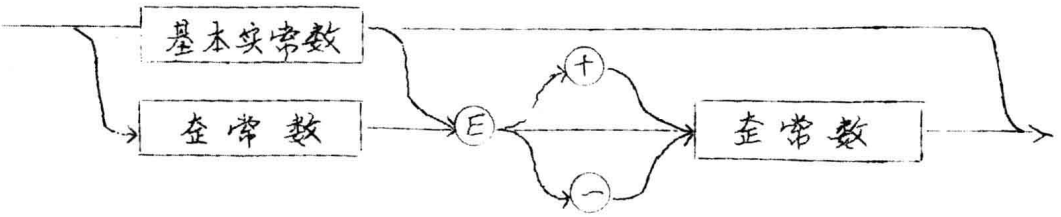
基本实常数（Basic Real Constant）



例如： .002487， 5.， .0， 72， 865.489

用这一术语定义一个实常数如下：

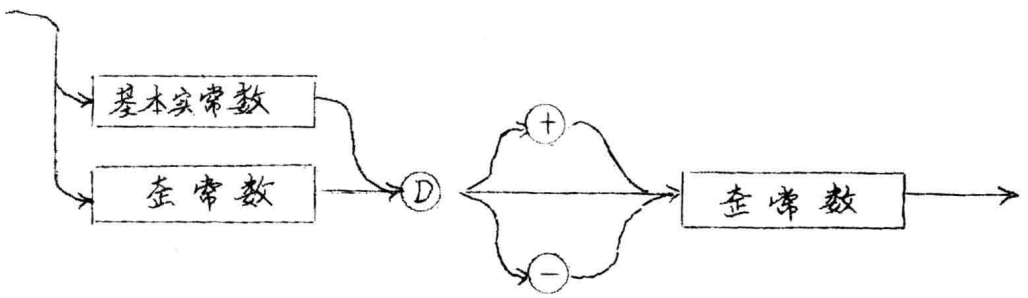
实常数



举例：5.E-32，2E5，0.，7.2，7.2E+8 。

可类似地给出一个双精度常数定义：

双精度常数（Double precision Constant）

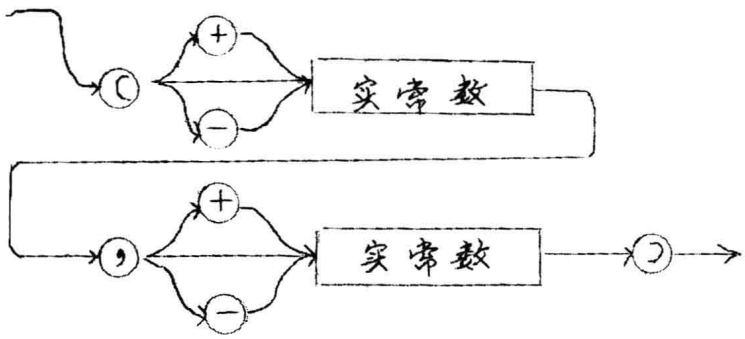


举例：5.D-32，2D5，7.2D+8，0D0 。

注意：一个双精度常数必须有一个D指数。某些编译程序也将具有多于某一定数目的数字的一个基本实常数当作双精度常数，但这既不是兼容Fortran也不是标准Fortran 。

复常数用实常数来定义：

复常数（Complex Constant）

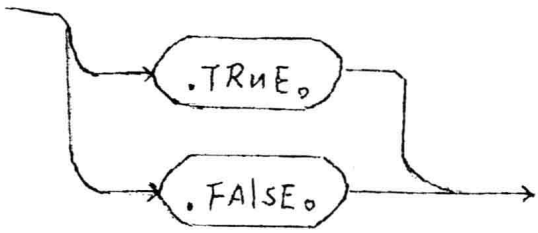


举例：（0.，0.），（-5.6E3，.25），（-1.0，-2.0E-1）。

双精度复常数不是标准常数。

只有两个逻辑常数。

逻辑常数（Logical Constant）



逻辑常数不准写做 .T. 和 .F.，而必须写全字。

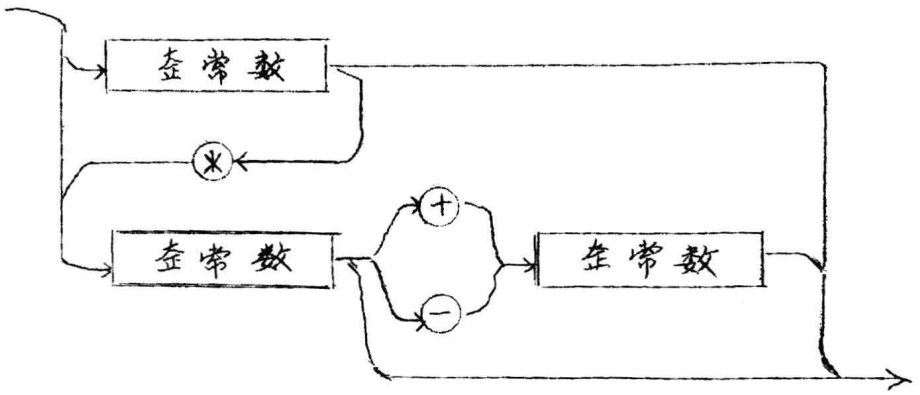
最后，有 Hollerith 常数。这些常数由一个整常数 n，后面跟一个 H，再紧跟 n 个字符（连字符中间的空白是有效的）组成。尽管许多编译程序允许使用这种 Hollerith 常数，但是在它的字符串中不许有逗号（或引号、或星号）。一个 Hollerith 常数不允许分成两行书写（见第三章）。

八进制常数或十六进制常数（开始是 O，或 Z，或 B）是被禁止使用的。

数组元素

在标准中，变量项只适用于由符号名直接表示的无下标的标量（如I或XYZVAL）。一个有下标的量（如A(I)）称为数组元素。数组元素的每一个下标必须遵守标准对下标表达式的规定。

下标表达式（Subscript expression）



举例：　2，　I，　2*K2S04，　84*IJK-57。

许多编译程序允许更多通用表达式作下标。有一些编译程序允许左变量，左常数和标符进行任意组合。而另一个编译程序允许任何含有数组元素和函数引用的左数表达式。正因为有了这些扩展，能检测非标准（而且非兼容的）下标表达式中的错误的编译程序很少。标准下标表达式在实际上只有7种可能形式。用C和K表示左常数，用V表示左变量，7种允许形式是：

C

V

V + K