计算机基础教育丛书

修订版（下册）

COBOL语言

谭浩强 编著

清华大学出版社

计算机基础教育丛书

COBOL 语 言（下册）

修订版

谭浩强 编著

清华大学出版社

（京）新登字 158 号

内 容 提 要

本书是作者在其编著的《COBOL 语言》一书的基础上修订补充而成的。作者针对初学者在学习 COBOL 语言时所遇到的问题，对各部分内容作了合理的安排，全书分上、下两册，本书为下册，内容紧接上册，包括数据部、子程序、表的建立和查找、磁带文件和磁盘文件、排序与合并、报表编制、程序的编译和运行等。

附录中列出了最新版本 COBOL-85 的语法及其说明。

本书可作为高等学校和计算机学习班的教材，也可供程序设计人员和企事业单位管理人员及其他初学者自学参考。

图书在版编目(CIP)数据

COBOL 语言　下册/谭浩强编著. —修订版. —北京：清华大学出版社，1995
（计算机基础教育）
ISBN 7-302--01783-2

Ⅰ.C…　Ⅱ.谭…　Ⅲ.COBOL 语言　Ⅳ.TP312CO

中国版本图书馆 CIP 数据核字（95）第 01932 号

出版者：清华大学出版社（北京清华大学校内，邮编 100084）
印刷者：通县宏飞印刷厂
发行者：新华书店 总店北京科技发行所
开　本：787×1092　1/16　印张：16　字数：395 千字
版　次：1995 年 7 月第 2 版　　1995 年 7 月第 1 次印刷
书　号：ISBN 7-302-01783-2/TP・789
印　数：0001—6000
定　价：12.00 元

计算机基础教育丛书

出　版　说　明

近年来，我国的计算机应用事业迅速发展，大批科技人员、大中学生、管理人员以及各行各业的在职人员都迫切要求学习计算机知识，他们已经认识到，计算机知识是当代知识分子的知识结构中不可缺少的重要部分。

计算机应用人才的队伍由两部分人组成：一部分是从计算机专业毕业的计算机专门人才，他们是计算机应用人才队伍中的骨干力量；另一部分是各行各业中从事计算机应用的人才，他们既熟悉本专业的业务，又掌握计算机应用的技术，人数众多，是计算机应用人才队伍的基本力量。他们掌握计算机知识情况和应用计算机的能力在相当大程度上决定了我国计算机应用的水平。因此，在搞好计算机专业教育的同时，在广大非计算机专业中开展计算机基础教育是十分必要的。

非计算机专业中的计算机教学，无论就目的、内容、教学体系、教材、教学方法等各方面都与计算机专业有很大的不同，它以应用为目的，以应用为出发点，如果不注意这个特点，将会事倍功半。广大非计算机专业的师生、在职干部迫切希望有一套适合他们的教材，以便循序渐进地迈入计算机应用领域，并且不断地提高自己的水平。我们在前几年陆续编写了一些适合初学者使用的教材，受到广大群众的欢迎。许多读者勉励我们在此基础上进一步摸索和总结规律，为我国的广大非计算机专业人员编写一整套合适的教材。

近年来，全国许多专家、学者在这个领域作了有益的探索，写出了一批受到群众欢迎的计算机基础教育的教材。特别是全国高等学校计算机基础教育研究会作了大量的工作，在集思广益的基础上，提出了在高等学校的非计算机专业中进行计算机教育的四个层次的设想，受到广泛的注意和支持。我们认为：计算机的应用是分层次的，同样，计算机人才的培养也是分层次的；非计算机专业中各个领域的情况不同，也不能一律要求，在进行计算机教育时也应当有不同的层次。对于每一个学习计算机知识的人，还有一个由浅入深，逐步提高的过程。

我们认为，编辑出版一套全面而有层次的计算机基础教育的教材，目前不仅是十分必要的，而且是完全有条件的。在全国高等学校计算机基础教育研究会和许多同志的积极推动和清华大学出版社的大力支持下，我们决定编辑《计算机基础教育丛书》。它的对象是：高等学校非计算机专业的学生、计算机继续教育或培训班的学员、广大在职自学人员。

本丛书包括计算机科学技术的一些最基本的内容，例如计算机各种常用的高级语言、微机系统应用基础、计算机软件技术基础、计算机硬件技术基础、微型计算机的原理与应用、算法与数据结构、数据库基础、计算机辅助设计基础、微机网络与应用、系统分析与设计等，形成多层次的结构，读者可以根据需要与可能选学。

本丛书的宗旨是针对广大非计算机专业的需要和特点来组织教材，敢于破除框框，从

实际出发，用读者容易理解的体系和叙述方法，深入浅出、循序渐进地帮助读者更好地掌握课程的基本内容。希望我们的丛书能在这方面闯出自己的风格，在实践中接受检验。

本丛书的作者大多数是高等学校中有较丰富教学经验的教师。但是，由于计算机科学技术的飞速发展以及我们的水平有限，丛书肯定会存在许多不足，丛书的书目和内容也应当不断发展和更新。我们热情地希望得到社会各界和广大读者的批评指正。

主编　谭浩强　林定基　刘瑞挺

关于下册修订的说明

考虑到COBOL语言的发展和应用需要，在这次修订中对《COBOL语言》（下册）一书作了以下较大的修订：

1. 在第十章中，将原书有关磁盘直接文件的内容删去。因为ANSI COBOL-74中不再包含这方面的功能，而且在实际使用中比较复杂，已很少被使用。在程序中已用相对文件来取代直接文件。

2. 增加了第十二章“COBOL的报表功能”。COBOL语言的一个重要特点是具有较强的报表功能，这是它适合于数据处理领域的重要原因之一。在实际应用中常用到这一部分的功能。在这章中介绍了有关的知识并详举了两个实例以帮助读者掌握。

3. 原书第十二章改为本书第十三章，内容也作了一定的修改。

4. COBOL的新标准ANSI COBOL-85已经公布，尽管目前在许多计算机上还不能运行COBOL-85，但随着时间的推移，COBOL-85必然会在我国推广开来，许多读者也希望介绍有关的知识，我们将在本书的附录中列出COBOL-85的语法表，并说明COBOL-85对COBOL-74的发展，以帮助读者了解COBOL-85的特点。有了COBOL-74的基础，在今后需要时学习和使用COBOL-85将是不会很困难的。

高志强同志帮助进行了本书的修订并执笔编写了第十二章的部分内容。在本书修订过程中得到许多专家和读者的支持与鼓励，谨表深切的谢意。

谭浩强

1994.5 北京

目　录

第七章　数据部之二

——数据部的较高技巧

§7.1　数据在计算机内的表示形式

至今为止，我们介绍的是数据在计算机内部存放的最简单的情况，即：一个字符在计算机内存中占一个字节，无论是数字（0～9）或字母（A～Z）或其它字符（如：＋，$，＊…）都各占一个字节。在本章我们将介绍其它的形式。

7.1.1　计算机内存的组织形式

在计算机中是以二进制形式来表示数据的。内存单元是由一系列的二进制位（它的值是 0 或 1）组成的，因此它的最小单位是二进制的“位”（bit）。

若干位组成一个字节（byte），在大多数机器中一个字节为 8 个二进位。在计算机高级语言中一般是以字节来作最小处理单位的，即存取数据以字节为单位。譬如说，取数据时一次至少要取出一个字节 8 个二进位的内容，如 00100100 等。

字节又可组成半字、字、双字。各种不同计算机对一个字包含几个字节有不同的规定。一般计算机以 4 个字节（32 位）作为一个“字”（word），以 2 个字节（16 位）作为“半字”，以 8 个字节（64 位）作为一个“双字”。

数据按指定形式存放在计算机的内存中。如一个字符可以用 ASCII 代码或 EBCDIC 代码表示（见附录Ⅵ），把它放到计算机内存中，需要占一个字节（8 位二进制）。从内存中取数据时，则是取出这个字节内容。

7.1.2　字符数据在内存中的存储形式

字符型、字母型和数值编辑型、字符编辑型数据项中的数据，每一个字符都在内存中占一个字节。这种形式称为标准数据形式。

由于内存中数据都是以二进制数来表示的，因此要规定每一个字符用怎样的一组二进制数来表示。每类计算机系统分别选择其所用的代码形式。例如，在 ASCII 代码中，以 8 位二进制数 01000001 代表“A”，以 00110001 代表“1”。而在 EBCDIC 代码中，以 11000001 代表“A”，以 11110001 代表“1”。总之，一个字符以 8 位二进制数代表，占一个字节。字符转换成二进制代码不是由用户自己转换的，而是由计算机系统自动转换的。例如在键盘上按下 A，B，C 三个字符，就把相应的二进制代码输入到计算机内存中去。反之，从计算机中输出一个数据，先将这些字节中存放的二进制数（如 ASCII 码的 01000001，01000010，01000011）取出，再把它们转换成字符（如 A，B，C）打印出来。

如果采用字符型数据形式，不论是字母或数字，都按一个字节存放一个字符。如有两个数据项 A 和 B，它们的描述为：

77 A PIC X (3) VALUE'ABC'.

77 B PIC X (3) VALUE'123'.

在计算机内存中的实际存储情况如下表所示：数据项 A 占三个字节，如果计算机系统用的是 ASCII 码，则第一个字节中的数据是 01000001，如果采用 EBCDIC 码，则第一个字节中为 11000001，其余类推。

	A			
实际的二进码	01000001 (11000001)	01000010 (11000010)	01000011 (11000011)	ASCII 码 EBCDIC 码
代表字符	'A'	'B'	'C'	

	B			
实际的二进码	00110001 (11110001)	00110010 (11110010)	00110011 (11110011)	ASCII 码 EBCDIC 码
代表字符	'1'	'2'	'3'	

特别需要指出的是，如果数据描述形式为字符型（例如本例中描述为 X (3)），则非数值常量中的数字亦按一般字符处理，如上述“1 2 3”，仅意味三个字符 1，2，3，而不代表一百二十三。

7.1.3 数值型数据在内存中的存储形式

数值型数据在内存中存放的形式有以下几种：

（一）外部十进制（或称扩张十进制）形式

按数值在机器外部的表现形式，一个数字在内存中占一个字节。如：

77 C PIC 9 (3) VALUE 486.

在内存中占三个字节。每一个数字与二进制代码的关系同上述。在内存中实际内容如下（假设机器内用 EBCDIC 码）。

C		
11110100	11111000	11110110
'4'	'8'	'6'

为表示方便，有时用十六进制数来表示一个数。一个十六进制数包括四个二进位数，用十六进制的“0”代表二进制数 0000，“1”代表 0001，“2”代表 0010，“3”代表 0011，“4”代表 0100，“5”代表 0101，“6”代表 0110，“7”代表 0111，“8”代表 1000，“9”代表 1001，“A”代表 1010（十进制的 10），“B”代表 1011（十进制的 11），“C”代表 1100，“D”代表 1101，“E”代表 1110，“F”代表 1111（F 就是十进制中的 15）。

因此 C 在内存中的情况可以表示为：

F4	F8	F6
'4'	'8'	'6'

如果数据项D的描述如下（D的值为负）：

77 D PIC S9(3) VALUE-486.

此时，负号不占一个字节，而在最后一个字节中放入某个信息，一般是将此字节的前四位1111改为1101，后四位仍为0110。即

D		
11110100	11111000	11010110

计算机检查最后一个字节的前四位，如为1101，则按负数处理，如为1100，按正数处理。或者说，用十六进制中“C”（1100）代表“正”，“D”（1101）代表“负”，“F”（1111）代表无符号，即绝对值（也有些计算机系统不用“C”，“D”，而用其它数代表“正”，“负”）。

（二）外部浮点数形式

某些数据，它的值很大或很小（如$+1.23876\times10^{59}$或-1.38457×10^{-69}），用以前讲的外部十进制形式存储是有困难的。COBOL允许用指数形式来表示一个数，例如上面两个数，用指数形式可以写为：

+1.23876E+59

-1.38457E-69

其中“E”表示“以10为底的指数”。“E”前面的部分称为“数值部分”或“尾数部分”，“E”后面的是“指数部分”或“阶码部分”。数值部分和指数部分各有一个符号以表示正或负。其一般形式为：

数符 数值部分 E 阶码符 阶码

为了表示这种指数形式的数据（外部浮点形式），在PIC子句中可以这样写：

77 A PIC +9.99999E+99.

或 77 B PIC +9V99999E-99.

它表示在内存中按以上形式存放数据。其中每一个“9”表示此位置可放入一个0～9间的数字，一个“9”占一字节。E前面放数值部分，E后面放指数部分。小数点“.”表示此位置有一小数点，“V”表示此位置有一隐含的小数点，它不占内存字节。正号“+”的意思是，如果数值为正，此处为正号，数值为负时，此处为负号。负号“-”的意思是：数值为正时此处空白，数值为负时，此处为负号（它们的用法和数值编辑项中的“+”、“-”编辑符相同）。“E”也占一个字节。因此，A在内存占12个字节，B占11个字节。如果A的值为1.23876×10^{59}，B的值为-1.38457E69，则A和B在内存中的情况为

A：+1.23876E+59（12字节）　　B：-1V38457E69（11字节）

在多数COBOL版本中浮点数可以表示的数的范围是5.4×10^{-79}到0.72×10^{76}。用外

部浮点形式，PIC 字符串中数值部分最多可以出现 16 个“9”，指数部分除“E”外，应有一个“+”或“−”，以及两个“9”（不应有三个或更多个“9”，因为不可能出现 10^{100} 以上的数值）。

对外部浮点项不能用 VALUE 子句赋初值，如下面写法是不对的：

```
02  N  PIC  +99.99E+99  VALUE  +12.45E+45.
```

只能从外部文件上读入以指数形式表示的数据（例如可以从卡片或磁盘上相应的位置开始，按列依次输入“+12.45E+45”），也可以用 MOVE 语句给 N 传送数值。

下面是一些例子：

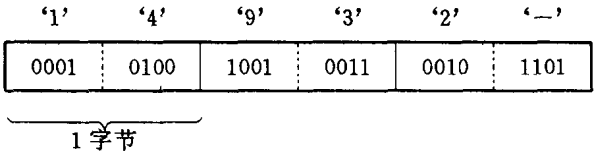
PIC 字符串	外部数据格式	表示的值
+99V9E−99	␣183E−13	$+18.3\times10^{-13}$
−9V99E+99	␣687E+65	$+6.87\times10^{65}$
+9 (3) .99E+99	+875.46E−12	$+875.46\times10^{-12}$
−V9 (6) E+99	␣678123E+37	$+0.678123\times10^{37}$
+.99E−99	+.87E␣62	$+0.87\times10^{62}$

（三）内部十进制（又称缩合十进制）形式

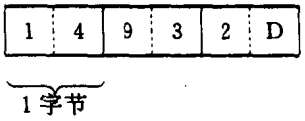
外部十进制形式在内存中一个字节放一个字符。数值型数据只用到 0～9 十个数字，如 129，23868 等。我们从下表中可以看到，0～9 的代码前四位是相同的。

十进制数字	EBCDIC 码	ASCII 码
0	1111 0000	0011 0000
1	1111 0001	0011 0001
2	1111 0010	0011 0010
3	1111 0011	0011 0011
4	1111 0100	0011 0100
5	1111 0101	0011 0101
6	1111 0110	0011 0110
7	1111 0111	0011 0111
8	1111 1000	0011 1000
9	1111 1001	0011 1001

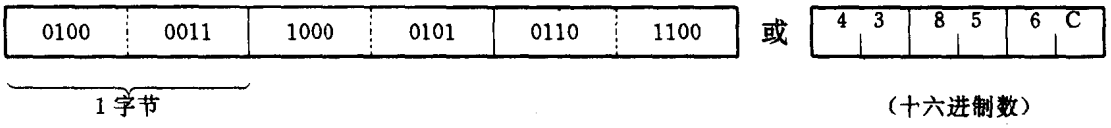
因此在数值型数据的前提下，为表示 0～9，前四位并不起辨别作用，只是根据后四位的不同来判别是 0～9 中的哪个数字。因此，为节省内存，可以只用四位二进制数字来代表一个十进制数。在一个字节中放两个十进制数字。每个数字占四个二进位，即半个字节。符号也占半个字节。如，−14932 在内存中为：



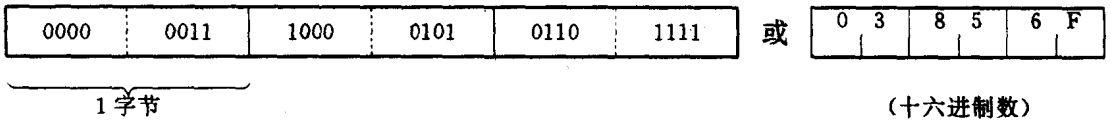
或用十六进制表示（十六进数的一位代表四个二进位）：



这种存放形式称内部十进制，或“缩合十进制”，+43856 在内存中为：



无符号的数 3856，在内存中为：



F 表示数值无符号，即取绝对值。由于存取的最小单位是字节，因此 3856 虽然只需两个半字节，但也占三个字节，最前面半个字节补零。

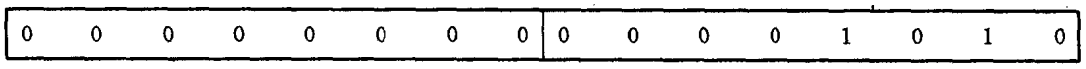
（四）定点二进制形式

不是以一个数字对应一个字节或半个字节，而是先把十进制数化成定点二进制数形式，然后存放在内存中。如：10 先化成 $(1010)_2$，括弧外下角 2 表示是二进制的数。然后把 1010 放到内存单元中。

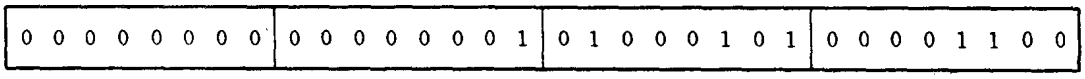
如果在数据部中以 PIC 9（4）描述一个数据项，并指定为定点二进制形式，数据长度为 4 位数字，则需要在内存中开辟二个字节。COBOL 规定在内存中根据数据项的长度分别用二字节、四字节或八字节来存放一个以定点二进制形式存放的数。

在 PIC 子句中描述字符 ‘9’ 的个数	占 内 存 字 节
1～4	2
5～9	4
10～18	8

例如，十进制数 10，如果以 PIC 9（2）描述，在内存中以定点二进制形式存放，则占两个字节：



十进制数 $(83212)_{10}$用二进制形式表示为 10100010100001100，由于用 PIC 9（5）在内存中占四个字节，存放形式为：

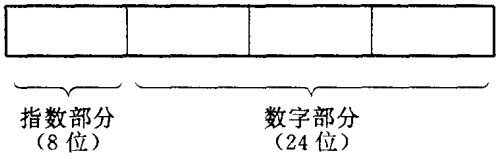


因为 2 个字节放不下十进制中所有的 5 位整数（2 个字节存放数的范围为－32767～

32768)，所以系统规定5位以上整数用4个字节存放。同样，4个字节放不下所有的10位整数，故规定10位以上的整数一律以8个字节存放。

（五）内部浮点形式

它以内部的指数形式（二进制的指数形式）来表示一个数，以固定长度的内存单元来存放一个数。如以4个字节（32位）表示一个数，其中8位表示指数部分，24位表示数字部分。这称为短浮点形式。



也可以用8个字节（64位）表示一个数，指数部分仍为8位，数字部分为56位，称长浮点形式。长浮点形式比短浮点形式有更高的精确度（内部浮点形式的数值在内存中所占字节长度只有4个字节和8个字节两种）。

无论短浮点形式或长浮点形式，所表示的数值范围均为：

$$5.4\times10^{-79}\sim0.72\times10^{76}$$

7.1.4 数据描述与存储形式的关系

（一）字母型、字符型、编辑型、外部十进制数据和以外部浮点形式表示的数据用标准数据形式来存放，即一个字符占一个字节。

如：77 A PIC X (6).
77 B PIC A (6).
77 C PIC 999.99.

均占6个字节。

（二）数值型数据可以由程序员任意选定存放形式（外部十进制、外部浮点形式、内部十进制、定点二进制、内部浮点形式）。用外部十进制形式最简单、最好理解，但一般说，它占内存多。如135879326九位数，用外部十进制需9个字节，用内部十进制需5个字节，用定点二进制需4个字节，短浮点形式需4个字节，长浮点形式需8个字节。

（三）数据在计算机内进行运算，都是化成二进制数以后再进行的，因此，外部十进制形式是不能直接进行运算的，需要由计算机把它们先化为定点二进制或内部浮点形式，然后再进行运算。假定A和B已定义为外部十进制形式，其值分别为11和12，它在内存中的存储形式为F1|F1和F1|F2（假设以EBCDIC代码存放），A和B各占两个字节。如果执行ADD A TO B，即A+B⇒B，并不是将A和B中各自第一个字节“F1”相加，第二字节“F1”和“F2”相加。而是先将A和B化成$(1011)_2$和$(1100)_2$，然后进行二进位的相加，相加后得$(10111)_2$，把它转换成23的EBCDIC码，即“F2”和“F3”，再存到B中。计算前后各要转换一次。显然，花时间多，速度慢。

从运算速度上看，定点形式最快，内部浮点形式次之，外部浮点形式和外部十进制、内部十进制最慢。因此，用户应根据需要选择用哪一种数据形式。这就要用下一节介绍的USAGE子句。

§7.2 用法子句（USAGE 子句）

使用用法（USAGE）子句可以使程序设计者自由选择数据在内存中的存放形式。譬如，有数据项 A 和 B，需要多次对它们进行运算，如果用外部十进制形式则来回转换会大大降低运算速度，这时，可以选择 A 和 B 为定点二进制形式或内部浮点形式。如果数据项 C 和 D 参加运算次数少，而且需要多次打印出 C 和 D 的结果，这时用外部十进制比较适合，因为它最适合打印的要求，不必再进行转换。

USAGE 子句的一般格式为：

```
[USAGE IS]  { DISPLAY
            { COMPUTATIONAL
            { COMP
```

其中“USAGE IS DISPLAY”的意思是“显示型的用法”，即：此数据项适于显示、打印，它采用标准数据形式（一个字节放一个字符）。

COMPUTATIONAL 和 COMP 是同一意思（COMP 是 COMPUTATIONAL 的缩写）。“USAGE IS COMP”的意思是“计算型的用法”（它只能用于数值型数据项）。表示此数据形式适于计算，它采用适于计算用的定点二进制形式或内部浮点形式。

标准 COBOL 只列出 DISPLAY 和 COMPUTATIONAL（COMP）两种用法的格式。各种计算机还规定了它可采用的有哪几种数据存放形式以及用什么来代表它们。例如，在 M 系列，IBM 系列和许多计算机系统中提供的功能包含：

DISPLAY	（标准数据形式，一个字节放一字符）
COMPUTATIONAL COMP	（定点二进制形式）
COMPUTATIONAL－1 COMP－1	（内部短浮点形式）
COMPUTATIONAL－2 COMP－2	（内部长浮点形式）
COMPUTATIONAL－3 COMP－3	（内部十进制形式）

在使用前，应查阅所用计算机的 COBOL 说明书。

说明：

（一）USAGE 子句是用来指定数据项在内存中的存储形式的。如：

```
02  A  PIC  9 (4)  USAGE  IS  COMP.
```

表示数据项 A 用定点二进制形式（假设该计算机系统所用的 COBOL 按我们介绍的上述规定）。

USAGE IS 这两个英文字可以省写。上面的数据项 A 描述体可简写成：

```
02  A  PIC  9 (4)  COMP.
```

（二）如省略 USAGE 子句，则隐含表示为用 DISPLAY 形式。

如

```
02  B  PIC  9 (6)  USAGE IS DISPLAY.
```

```
02  B  PIC  9 (6)   DISPLAY.
02  B  PIC  9 (6).
```

以上三种写法等价。字符型、字母型、编辑型、外部十进制、外部浮点形式的数据项必须用 USAGE DISPLAY（可以采用显式或隐含形式表示）。

（三）如果对组合项描述为某一种存储形式，则表示这个组合项的下属各初等项都是这种形式。如：

```
01  T.
    02  T1  USAGE  COMP.
        03  X  PIC  S9(3).
        03  Y  PIC  S9(3).
```

在 T1 的描述中用了 COMP，表示定点二进制，它下属的 X 和 Y 也都是定点二进制形式。它相当于：

```
01  T.
    02  T1.
        03  X  PIC  S9(3)  COMP.
        03  Y  PIC  S9(3)  COMP.
```

组合项的描述中如果用了用法子句，其下属的初等项可以再用用法子句，但二者的说明不应矛盾。例如将上面的 02 层 T1 描述体改写为：

```
02  T1  USAGE  COMP.
```

是可以的，因它与 03 层中用的一致。但如果改成：

```
02  T1  DISPLAY.
```

则 02 层与 03 层描述发生矛盾。不允许。

（四）USAGE 子句指定的数据存储形式不应与 PIC 子句指定的数据类型矛盾。

如：

```
    04  A  PIC  A(4)      USAGE  COMP.
或  04  B  PIC  $99.99    COMP-1.
```

都是错误的。前者 A 是字母型数据，不能采用数值型的存储形式。后者 B 是编辑数据型数据，当然也不能采用短浮点形式。

（五）长、短浮点形式已确定了内存的长度，不应再用 PIC 子句。如：

```
04  A1   COMP-2.
```

没用 PIC 子句，但已确定为长浮点形式，如果某一具体计算机规定长浮点形式占 8 个字节，则 A1 不论是什么值，在内存中都占 8 个字节。

（六）在传送或运算时不同存储形式的数值型数据间可互相转换。

```
如:77  A  PIC  9(3)  COMP.
   77  B  PIC  999V99  DISPLAY.
```

二者的存储形式不同。如果有语句：

```
MOVE  A  TO  B.
```

则先将 A 转换成 B 的形式再存入 B。又如：

```
ADD  A  TO  B
```

也由编译系统进行自动转换，使 A 和 B 具有同一形式，相加后按 B 的形式存入 B。