IBM

个人微型計算机
参考資料

Pascal

上海长江电子计算机厂

目　　录

第一章　引言……（　1～5　）
第二章　编译Pascal程序……（　6～20　）
第三章　符号的表示法和专用术语……（21～25）
第四章　编译程序命令(元语言)……（26～35）
第五章　标识符和常量……（36～42）
第六章　数据类型……（43～62）
第七章　变量说明和使用……（63～69）
第八章　表达式……（70～76）
第九章　语句……（77～85）
第十章　过程和函数……（86～97）
第十一章　有效过程与函数……（98～108）
第十二章　文件系统……（109～125）
第十三章　编辑……（126～133）

第一章　引　言

目　录

IBM个人计算机Pascal语言 ……（2）
Pascal语言 ……（2）
IBM Pc Pascal语言扩展 ……（2）
编译程序命令……（2）
设备……（2）
属性……（3）
超数组……（3）
字符串……（4）
常量值……（4）
系统实现……（5）
小结……（5）

IBM个人计算机Pascal语言

本文叙述了IBM个人计算机的Pascal语言（即IBMPC Pascal）。在阅读它之前，读者应从Jensen与Wirth合著的《Pascal用户手册及报告》、Welsh与Elder合著的《Pascal语言导论》或其它出版物中对于Pascal语言有一个一般的了解。

Pascal语言

Pascal最初是由Niklans Wirth为以下两个目的而设计的：作为教授程序编制的一种系统训练的方法以及作为完成编程的一种有效途径。但是Pascal作为一种普通的目标编程语言和系统程序工具语言，带来了更广泛的好处。

• Pascal这种比较先进的语言大大改善了许多早期的语言（如ALGOL），并且成为一些更新的语言的基础。

• 要强调指出的是Pascal是一种较高级的语言，即它对于描述数据的结构与运算提供了有力的支持，在以下几方面表现出其独其的优点：

• 用户可不必关心数据的具体表示（如每个变量所占的字节数、数组的组织、地址范围等等）。

• 程序员可以更精确地指定组成变量的字符，如指定这些字符的取值范围(VARI:O..99）或决定在访问变量构成时是否要以空间换取时间。（PACKED关键字）

IBM还增加了以下一些功能：

1、IBM PC Pascal被设计成一种系统工具语言，特别适应于书写编译程序、解释程序、操作系统等等。

2、产生有效的目的码是最重要的。各种语言的扩展和全程优化措施都使编译程序所需要的时间与存贮容量达到最少。

3、用汇编语言可以容易地做到的操作，在IBM PC Pascal中也应该可以容易地做到。

IBM PC Pascal与ISO规约（ISO/TC 97/SC5N595）相一致。

我们的意图就是要通过IBM PC Pascal编译程序使正确的标准程序可不经修改就得到编译和运行。基于IBM PC Pascal中有一些新的保留字及其它元素，IBM的特征与扩充都列于附录E中。

IBM PC Pascal语言扩展

扩展包括：

- 编译程序命令
- 属性
- 超数组类型
- 使用CONCAT进行的串处理
- 常量值
- 系统实现扩展

编译程序命令

我们提供以下诸方面的功能：产生错误校验码，列表格式控制，形成条件编译，把另外的源文件插入编译。共有30条有效的“元语言”命令（“metalanguage”）。

设备（Unit）

Pascal对于很长的程序（如系统软件应用）来说是一种极好的语言，IBM Pascal通过使用设备这个概念来支持分隔编译的模块。

所谓设备，就是一组相关的数据类型、变量、常数、过程及函数再加上一个初始化过程。它有两个部分：接口和实现。

接口包括一个用以输出的标识符列表和它们的说明（包括过程与函数的首标）。

实现包含任何局部的说明、过程与函数的本体及初始代码。某些设备子程序可用汇编码代替Pascal写成。

一个程序（或实现或其它接口）可以使用只给出接口而不给出实现。这一模式能比外部过程与变量提供更多的的把程序分成模块的结构方法。

属性（Attributes）

变量、过程和函数的属性提供在链接文本级上进行控制的功能。

属性包括：

- 用于全程标识符链接的PUBLIC与EXTERN
- 用于变量的STATIC与READONLY，用于过程和函数的PURE

超数组

IBM PC Pascal提供一种超数组的类型，使数组的长度可变。在超数组中，只规定了下界，而不规定上界。

虽然超数组类型不能直接用于变量(即：VAR-SUPER ARRAY[0..*]OF REAL;)，但它可以用于两个重要的方面：

- 作为一个形式参数的类型
- 作为一个指示字的引用类型。

另外，可以使用一个“标志符”来把超数组的类型送给任何变量。例如：

```
TYPE
  VECT=SUPER ARRAY[0..*]OF REAL;
VAR
  PVEC: ^VECT; V10: VECT(10);
PROCEDURE SORT(VAR V: VECT);
BEGIN
  NEW(PVEC, UPPER(V));
   .
   .
   .
END;
   .
   .
   .
SORT(V10);
```

这里VECT是超数组类型。PVEC是一个指明类型的指示字。

NEW用第二个参数所给出的上界定义一个新的VECT，这里上界就是参数V。

V10是一个变量，是以10为上界的VECT的一种情况。参数V可以取由VECT得来的任何类型的变量，例如V10或PVEC∧。

超数组这个概念可以巧妙而有效地处理动态的或一致的数组。

字符串

标准Pascal中的字符串是固定长度的。然而，BASIC和PL/I的一个主要特点，却是可以定义一个字符串类型的变量，使其长度可变，这也就是一个常用的无名数据类型。

字符串类型是：

SUPER PACKED ARRAY[O..*]OF CHAR

其中零元素包括了长度。我们经常具有一组串过程或串函数。

IBM PC Pascal有一个CONCAT过程，它取出两个字符串参数，并把第二个连到第一个后面去。

字符串处理能很方便地实现，这是因为所有的过程和函数都能用下面的方式写出：

- 指定字符串类型为一个固定长度字符数组。
- 用一个超数组或其它一致（Conformant）数组的扩展来得到一个变量长度字符串类型。

在IBM PC Pascal中字符串的分配、比较和读/写都自动完成。

常量值

IBM PC Pascal的另一特性是有关常量值方面的。许多带常量值的关系式是在编译时进行运算的，所以如果WORDCOUNT和SYMBOLCOUNT是常量的话，就可以定义另一个常量TOTALCOUNT = WORDCOUNT + SYMBOLCOUNT。

数量可以是二进制的、八进制的、十进制的或十六进制的。在READ和WRITE语句中也是这样。

象数组和记录常量一样，对于串常量来说，也有一个连接操作符。

一个程序中可以包含一个以上的VALUE部分，把一个初始的常数值赋给变量。

最后，一个形式参数可以跟在关键字CONST之后，它的作用与VAR相似，唯一的不同是，实际参数可以是一个常量，并不能在例行程序体中进行定义。

系统实现

有几种扩展对系统实现工作来说是特殊的：

- WORD类型
- ADDRESS类型
- 输入/输出能力
- 交互READ
- 随机文件和文件形式
- 内部过程和函数

最重要的是WORD类型，它的范围为O到65535（MAXWORD），就象INTEGER类型的范围为—32767到32767一样。

十六位数可以被作为带符号的或不带符号的来处理，所以数的实际范围为—32768到65535。

有了WORD和INTEGER这两种类型，使Pascal程序可以使用上述的数的范围（除了—32768以外）。

不带符号的字经常在系统中使用，作为地址，最大范围的整形量、一组位或不知语义的存贮值。

试把INTEGER类型用于比较大小这个问题（#FFFF是否大于#FFFE）。

另外，还有一些操作符如AND、OR、XOR也都允许WORD类型。

还增加了地址类型，它类似于一个指示字，但允许分段寻址。这对于访问系统数据还是很有用的。

增加的其它性能还包括各种输入/输出方式(I/O出错陷阱、使用“Lazy evaluation”交互READ、随机文件、文件形式、串READ等等)和附加的内部过程与函数(如ENCODE与DECODE、RETYPE与RESULT）。

小结

IBMPC Pascal可用于系统元件实现。它包括了用结构方法产生和获得Pascal程序的许多有用特性，它们对一般系统编程工作来说也是必需的。

一些特殊的特性包括分列的编译设备，变量长度串、超数组类型及机器的初始定位。

IBM PC Pascal不仅仅是一种工具，而且是工具的制造者，使用它可以写出十分精巧的程序，使计算机更容易使用，也更为人们所接受。

注：在本书的以后部分，IBM Pascal就是指的IBM PC Pascal。

第二章　编译Pascal程序

目　　录

基本的条件……（7）
首次运行……（7）
　复制主盘片……（7）
　建立PAS1盘和PAS2盘 ……（7）
　建立PASCAL.LIB盘……（7）
开始编译……（7）
　启动编译程序：PAS1 ……（8）
　继续编译：PAS2 ……（9）
　连接……（10）
　运行Pascal程序… ……（11）
　任选的PAS1命令行 ……（11）
　任选的PAS2命令行 ……（12）
　任选的连接命令行……（12）
　用批文件进行编译……（13）
　编译大程序……（13）
　编译程序列表……（13）
　连接程序内存分配……（17）

基本的条件

要在IBM个人计算机上成功地编译一个Pascal程序，就需要：

·Pascal程序包：

—三张5吋主盘片，它们分别被为PAS1、PAS2和PASCAL.LIB

·PAS1包括下述文件：

—PAS1.EXE

—PASKEY

—FILKQQ.INC

—FILUQQ.INC

—ENT6XS.ASM

·PAS2包括一个文件：

—PAS2.EXE

·PASCAL.LIB包括两个文件：

—PASCAL.LIB

—PASCAL

·至少为128K字节的机器内存

·两个盘驱动器

·一台打印机

·一台显示器（IBM PC单色显示器、监视器或带有RF调制器的电视机）

·IBM PC磁盘操作系统（DOS）参考手册及磁盘

·一张被称为（Scratch）盘的5 1/4吋盘片

首次运行

首次运行时必需要有：

·三张5 1/4吋盘片来复制主盘片。

复制主盘片

我们建议用户尽快地复制PAS1、PAS2和PASCAL.LIB这三张主盘片以作后援。同时也建议用户在日常操作中使用这三张复制的盘片，而把主盘片放在安全的地方。

建立PAS1盘和PAS2盘

现在你已经把PAS1和PAS2复制好，接下来还需从DOS盘中把COMMAND.COM文件复制到PAS1和PAS2上去。这是因为当PAS1和PAS2被装入时，它们将复盖内存中的COMMAND.COM（COMMAND.COM当系统启动时就被装入内存）。

这样当PAS1或PAS2完成执行以后，COMMAND.COM就自动被再装入。

建立PASCAL.LIB盘

把OOS盘中的链接程序LINK.EXE复制到PASCAL.LIB上去。

现在就可以准备编译一个Pascal程序。

开始编译一

我们推荐把以下的步骤作为一个一般规则：

1、把Scratch盘格式化。从IBM PC DOS参考手册中查看格式化的方法。

2、通过复制程序已在其上的磁盘或者通过行编辑手段（参阅IBM PC DOS手册中

的“EDLIN”），把程序送到Scratch盘上。

3、在程序名后面加上“.PAS”这个扩展名，以表示这是一个Pascal程序。

现在就可以准备编译这个Pascla程序了。

注：你可以用大写体、小写体或大小写混合体来打入编译命令。

启动编译程序：PAS1

PAS1是编译程序的第一次通过。PAS1阅读源文件并检查句法的正确性。它产生两个中间文件存放在Scratch盘上，但不侵占源程序的存贮空间。这两个中间文件是：

PASIBF·SYM—符号表

PASIBF·BIN—中间二进制码

PAS1还产生一个源列表文件。如果系统配有打印机的话，建议打印一份源列表，以在改错时对照。

在编译程序的PAS1部分时应按下列步骤做：

1、把当前驱动器改为B，打入：

B:

2、把写有程序的磁盘放入B驱动器

3、把PAS1盘放入A驱动器

4、打入：

A: PAS1

PAS1就被装入计算机。相隔很短时间后，编译程序就会显示一排字符和提示符：

Source filename[.PAS]: —

Source filename就是程序文件的文件名，例如：

Source filename[.PAS]: myfile

不一定要在打入的文件名后加上.PAS这个后缀，因为编译程序将会自动去寻找.PAS这个扩展名。但如果文件的扩展名不是.PAS，那么就要打入这个后缀。打入源文件名后就可以看到这样的指示符：

Object filename[MYFILE.OBJ]: —

Object filename是你要给予目标（机器可读）文件的文件名。如果你希望的目标文件就是MYFILE.OBJ（即出现在方括号中的名字），只要简单地按下Enter键就行，否则的话也可以给文件另外一个名字，但必须注意其扩展名只能是.OBJ。在我们举的例子中，假定只按了Enter键：

Object filename[MYFILE.OBJ]: —

接下来的指示符就是：

Source listing[NUL.LST]: —

Source listing就是你要给予包含编译源列表的文件的名字。如果不需要列表的话，只要按Enter键。这样给出一个缺省的文件名NUN.LST，它将告诉编译程序不产生源列表文件。

在例子中，假如我们需要列表，打入：

Source listing[NUL.LST]: myfile

（注：编译程序将加上.LST这个扩展名。）

最后的提示符是：—

Object listing[NUL.COD]：—

Object listing就是将包含汇编目标文件列表的文件名。在例子中，假定有如下的响应：

Object listing[NUL.COD]：Myfile

（注：编译程序将加上.COD这个扩展名。）

下面是你使用了例子中的文件名后完整的屏幕显示：

Source filename[.PAS]：myfile

Object filename[MYFILE.OBJ]：

Source listing[NUL.LST]：myfile

Object listing[NUL.COD]：myfile

一经打入上面最后一个名字，编译程序就开始第一次通过你的程序。如果程序中有句法错误，编译程序就在屏幕上和列表文件中显示这些错误（参阅本章最后的“编译列表”）。

完成首次通过后，编译程序显示一个信息，指出它所发现的错误与警告的数目。如果把源列表送入文件，这一信息如下所示：

Pass One No Errors Detected

如果有错误或警告，则它们在屏幕上显示如下的一个信息或两个都显示：

Errors Detected

Pass One Had Warnings

如果编译中确实发现了错误，就必须在源程序中找出这些问题并修改它们，然后在继续用PAS2编译之前再回到PAS1。

如果在PAS1盘上没有复制COMMAND.COM，那么，现在就需要插入DOS盘了。

继续编译：PAS2

PAS2是Pascal编译程序第二部分的名字。在这一部分的编译过程中，编译程序读入由PAS1所产生的.SYM和.BIN文件，并生成.COD和.OBJ这两个目的文件。这也是一个优化的过程。

PAS2在读入并删除PASIBF.SYM和PASIBF.BIN的同时，还产生一个名为PASIBF.TMP的文件（即中间连接文本），并对它进行写、读，最后将其删除。

某些程序也许在PAS1中被正确编译了，但在PAS2中却出现了错误，这些错误可能是：

- 超出了内存容量
- 超出了范围
- 有溢出

在修改了这些错误以后，应该仔细检查一下，看看是否还存在语法上的错误，接下来就可以完整地编译一次程序。

写有PAS1文件的磁盘应放在驱动器B中。

在进行PAS2编译时，可以按以下步骤去做：

1、把PAS1盘从驱动器A中取出。

2、把PAS2盘放入驱动器A。

3、打入：

A：PAS2

PAS2不需要对其进行任何输入。它将产生目的文件和一个表列。当编译完成后，PAS2将给出如下所示的信息：

Code Area Size = #0116 (278)

Cons Area Size = #005E (94)

Data Area Size = #000E (14)

Pass Two No Errors Detected.

这里，Code Area Size是你的程序所占用的总的字节数（例子中是278个字节）。Cons Area Size是你程序中（数组、串、结构及REAL等等。）的常数所占用的总字节数。Data Area Size则与分配数据的STATIC有关，这个区域总是从偏移地址*2开始。这三个长度都是以十六进制和十进制形式给出的。

如果在这第二部分的编译过程中检测到错误，请参看本书的附录A“出错信息”，并改正这些错误，然后再运行PAS1和PAS2。

链接

为对链接的概念有所了解，请阅读《IBM个人计算机磁盘操作系统》（即DOS）这本参考手册。

链接的过程应以下列步骤进行：

1、从驱动器A中取出PAS2

2、把PASCAL.LIB的复制盘（上面就是链接程序）放入驱动器A

3、打入：

A：Link

这样就开始引导链接程序并给出如下提示：

Object Modules：

在这一提示的后面，打入目的文件名（不是目的列表文件）。文件名中的.Obj扩展名部分不用打入。对于我们的例子来说，就应打入：

Object Modules：myfile

接下来的提示符就是：

Run File：

在这一指示之后，打入你为你的可执行代码程序文件所取的名字。这一文件名将被打上扩展名.exe，并存放于驱动器B。例如：

Run File：myfile

下一个提示为：

List File[MYFILE.MAP]：

压下Fnter键就使连接程序选择缺省的文件名即MYFILE.MAP。这个文件包含了连接分配信息，并转向缺省的驱动器即B。

接下来显示：

Libraries[ ]：

当你链接一个Pascal程序时，Pascal库就自动调进，当然你也可以打入；（其实并不需

要这样做）

a：Pascal.lib

接下来的五个指示为：

Publics〔No〕：

Line Numbers〔No〕：

Stack Size〔Object file stack〕：

Load Low〔Yes〕：

DSAllocation〔No〕：

如果你对“Publics”和“Line Numbers”打入y（yes），则这些信息就放在名为MYFILE.MAP的文件中去。

建议用户对上述两条提示作缺省值回答，即只要压下Enter键即可。

对于“Stack Size”和“DSAllocation”来说，编译程序将对除了缺省回答以外的任何选择都不加理采。（它们是由Pascal本身设置的）。

连接程序将对DSAllocation回答“Y”，这也是一个缺省值，所以Enter(即缺省值）就被认为是有效的。

对“Load Low”的回答也必须是“Y”，这也是一个缺省值，所以Enter也为有效。

如果使用我们给出的例子，提示信息的屏幕就如下所示：

```
B a.link
IBM Personal Computer Linker
Version 1.OO(c)Copyright IBM Corp1981
Object Modules myfile
Run File: myfile
List File: 〔MYFILE.MAP〕
Libraries〔 〕
Publics〔No〕:
Line Numbers〔No〕
Stack Size〔Object file Stack〕
Load Low〔Yes〕
DSAllocation〔Noo〕
```

现在就开始对程序进行链接。当链接完成时应该在B盘上有一个运行文件(RunFile)。建议用户把盘目录显示一遍，看一看这个运行文件是否确实存在（它的扩展名为.exe）。使用我们给出的例子，就可在目录列表中看到myfile.exe这个文件名。

运行Pascal程序

要运行程序，只要打入运行文件名（不加.exe扩展名）即可。例如：

```
myfile
```

一经运行结果表明它是正确的，就可以把它复制到另一张盘上去。

任选的PAS1命令行

PAS1也可以用下面的命令行来启动（当然，应该用你的文件名来替换下面的四个文件）：

PAS1源文件，目的文件，源列表，目的列表；

使用这个命令行以后，不再得到PAS1这一节中所述的提示信息。

注意：可以对列表文件加任意的扩展文件名，在我们的例子中用的是.any。

命令行还允许某些其它的变化，如可以只指定：

PAS1源文件；

给出这个命令行后，编译程序将使用其它几个文件的缺省文件名，它们是：

目的文件.OBJ

NUL.LST（即没有源列表）

NUL.COD（即没有目的列表）

另一个变化形式是：

PAS1源.any''）；

这将产生一个名为源.OBJ的目的文件和一个名为源.LST的源列表，但没有目的列表。

注意：插入一个逗号就使.LST和.COD的缺省名作废，而以源文件名加上相应的扩展名来给出列表。

你也可以使用这样的命令行：

PAS1源,,,SCODE.ANY;

这将产生源.OBJ、源.LST和SCODE.ANY这三个文件。

最后，你这样指定：

PAS1源,,,,'

这将产生三个文件：源.OBJ、源.LST和源.COD。如没有给出源文件名，那么PAS1将要求给出一个。如果没有用"；"来结束命令行，则编译程序将要求再输入剩下的文件。

任选的PAS2命令行

在PAS2后面跟一个参数，就可使PAS2在启动以后有一段间歇。例如：

a：PAS2/P

PAS2读入以后，有以下提示：

Hit "enter" to begin Pass two,

现在要等到压下Enter键以后才开始编译。

任选的连接命令行

连接程序可以使用一个"自动生成文件"的任选项，指定这一任选作用的命令行为：

Link ARFILE

如果存放文件的盘名不是缺省盘名的话，就应在命令中再包括一个盘名。

例如：

a：Link a：ARFILE

关于如何自动产生这些连接程序的结果文件的介绍，请参阅《IBM个人计算机磁盘操作系统（DOS）》参考手册。

IBM Pascal提供了一个自动产生的文件以供使用，它在PASCAL.LIB盘上，名为PASCA2（没有扩展名）。你可以使用这个文件，如果需要的话还可以修改。要使用这个文件，打入：—

a：Link a：PASCAL

这个文件将要求得到目的模块名，产生一个名为RESULT.EXE的运行文件，并对其他提示以缺省值作为回答。

用批文件进行编译

自动产生相应文件的功能使编译程序能用批文件来自动运行（参阅《IBM个人计算机磁盘操作系统（DOS）》参考手册中对批文件（.BAT）的解释）。

下面就是一个例子：

```
Pause……Insert PASlin drive A
a: PAS1 myfile,,,,
Pause……Insert PAS2 in drive A
a: PAS2
Pause……Insert Linker*in drive A
a: LINK a: PASCAL
```

如果已经把上面这些内容存放到盘中名为RUN.BAT的文件中，那么只要打入：

```
RUN
```

就自动完成PAS1、PAS2和连接这些工作。

编译大程序

有可能盘上没有足够大的容量来容纳所有由编译程序产生的文件（将会出现一个“Out Of Space”（即“超出容量”）的信息）。

在这种情况下，可以把编译好了的源列表从盘上删除，然后再继续做PAS2。

有时还需把源.LST和目的.COD文件送到显示屏幕（对控制台来说它就是CON）、打印机（LPT1）、RS232接口（AUX或COM1），或者把它们全部删除（NUL），以为编译的最后文件提供磁盘容量。

上面提及的特殊文件名也可用于其它命令中（见十二章“过程ASSIGN”中对它们和其它终端文件名的解释）。

在某些情况下，甚至必须删除源文件本身（如有必要应先把它复制到其它盘上），而在盘上只留下PASIBF.SYM和PASIBF.BIN。这是为了给PAS2提供足够的容量来暂时存放它所产生的PASIBF.TMP文件，并然后再产生和存放.OBJ和.COD文件。

一个很大的程序可被分成较小的单元或模块，由PAS1和PAS2分别进行编译（见十三章“编辑”中对设备和模块是怎样组成与分别编译的解释）。然后再用连接程序把它们连起来产生一个运行文件。

这是通过在连接程序给出的Object Modules提示之后为各个文件指定目的模块而做到的：

```
Object Modules: file1, file2, file3, file4
```

编译程序列表

列表格式举例

```
LISTNG FORMAT EXAMPLE                                    Page1
CHAPTER2                                                 07-27-81
                                                         11:09:55
 JG  IC  Line*  Source Line  IBM personal Computer Pascal Compier
```

```
V1.00
     00   1  { $title: ' LISTING FORMAT EXAMPLE', $subtitle: '
               CHAPTER2' }
            2
     00     3  Program Nonsense;
     10     4  Label 10;
     10     5  Var int,K: integer; finished: boolean;
            6
     20     7  Procedure Print(var stop: boolean);
     20     8  label 12;
     20     9  Var i, j: integer;
     30    10  Function Addit(VarJ: integer): integer;
     30    11  begin(*addit*)
 *   31    12  if j=10 then return; {returnjump}
 %   31    13  int: =j+addit(int)  {combination Of globals}
     20    14  end; (*addit*)
           14  ----/\306Function Assignment Not Found
 Symtab    14  Offset Length Variable-ADDIT
               -  4  10Return Offset, Frame length
               -  8  2(function return): Integer
               -  2  2 J            : Integer Varp
     20    15  begin(*Print*)
     21    16  if K>10 then [Stop: =true; goto 12]; {forward jump}
     21    17  for i: =K to 10
     21    18  begin
       /\  17----------/\Warning 172 Insert DO
  1  22    19    J: =addit(int);   {passing a global}
     22    20  writlen(j)              tifi
           20  -------/\303Unknown Identifier Skip Statement
           20  -187 End Skip/\
     21    21  end;
  =  21    22  12 : K : =K+1; {assign to global}
     21    23  writeln('Exit Procedure Print for the', K: 1', th time'
           23  ---------/\Warning 238 Assumed OUTPUT
     10    24  end; (*Print*)
 Spmtab    24  Offset Length Variable-PRINT
               -  2  12 Return Offset, Frame length
               -  0  2  STOP        : Boolean Varp
               -  6  2  I           : Integer
```

```
                - 8 2 J                    : Integer
LISTING FORMAT EXAMPLE                                    Page 2
CHAPTER 2                                                 07-27-81
                                                          11:10:01

 JG IC Line* Source Lien IBM Personal Computer pascal
Compiler V1.00
        25 { $page+ }
    10  26 begin(*main*)
    11  27 K: = 1;
    11  28 finished: = false,
        28 --------------^Warning 156, Assumed;
    11  29 10: Print(finished);
 -  11  30 if not finished then goto 10; {backwarads jump}
    OO  31 end. {*main*}
 Symtab 31 Offset Length Variable
             0     8  Return Offset, Frame Length
             4     2  K                    : Integer  Static
             2     2  INT                  : Integer  Static
             6     1  FINISHED             : Boolean  Static
           Errors Warns In Pass One
             3     3
```

以下对IBM Pascal编译程序元命令的解释可作为一个引用。参见本书第四章对元命令的完整描述。

Pascal源列表的每一页都有一个页头，在页面左上方的两行的内容是用户对程序标题及子标题的选择，这是用 $TITLE和 $SUBTITLE元命令来设置的。

页面右上方三行为页编号、日期及时间。页编号可用 $PAGE:n来设置，而接下去的页号则用 $PAGE+来设置。编译程序名及版本编号出现在页头的下一行，前面还有一个列标记。

IG列标记

JG列中是由用户信息产生的标志字符。

“J”列中包含以下的转移标志：

J标志	含义
+	正向转移：没有遇GOTO标记或BREAK。
-	反向转移：遇到了GOTO标记或CYCLE。
*	其它转移：RETURN等

“G”列对全程变量给出标志（而不是对局部的现行进程或函数）。这一列可有下面三种标志：

G标志	含义
=	对一个非局部变量赋值。