IBM-PC-07

IBM PC
PASCAL 编译程序

中国科学院科理高技术公司

一九八四年十二月

目　　录

第1章 引　言

1.1 IBM个人计算机Pascal

本手册叙述IBM个人计算机Pascal。要求读者具有 Pascal 语言的一般知识，为此请参阅Jensen和Wirth的“Pascal用户手册和报告”及Welsh和Elder的“Pascal导言”等书籍。

1.1.1 Pascal语言

Niklaus Wirth最初设计Pascal时，有两个原始目的：一是将程序设计的讲授作为一种系统训练；二是用一种既可靠又有效的方法实现程序，但作为一种通用编程语言和一种系统程序实现语言广为流传的理由是：

·Pascal是一种更为近代的语言，它既具有许多较早语言（诸如ALGOL）的优点，又构成几种新语言的基础。

·主要应强调Pascal是一种更高级的语言；也就是说，它是一种有用的、能说明数据结构和算法而和其后的实现无关的抽象工具。

·用户不需要涉及数据的表示（每个变量的字节数，数组的组织、地址大小等等）。

·程序员能较精确地说明变量的特性，诸如，变量取值范围（VAR:I:0..99)或决定是否要权衡存取变量分量（PACKED保留字）的时间和空间。

IBM为此还增加几条：

1，IBM个人计算机Pascal被设计成一种系统实现语言，特别适用于写编译程序，解释程序，操作系统等。

2，产生有效代码是主要的。各种语言扩充和全局优化程序阶段（通过第二遍编译）的

全部工作都力争尽量减少被编译程序所需的时间和空间。

3．凡用汇编语言能容易做到的操作，用IBM个人计算机Pascal也应容易做到。

IBM个人计算机Pascal普遍遵照ISO草案（ISO/TC97/SC5N595）。

IBM的目的是使正确的标准程序能在IBM个人计算机上正确地编译和运行。Pascal编译程序不加以改动、而IBM特征引进了新的保留字和其他成份。故有关IBM特征和扩充总结在附录E。

1.2 IBM个人计算机Pascal扩充

Pascal扩充包括：

- 编译指引
- 属性
- 超数组类型
- 用CONCAT的字符串处理
- 常量值
- 系统实现扩充

1.2.1 编译指引

本手册给出对错误检验代码产生的控制、清单格式控制、条件编译结构和编译中其他源文件的插入机理、这类可用的“元语言”命令有30条。

1.2.2 单元

Pascal对非常大的、又必须确实可靠的大程序（如系统软件应用）是一种优秀的语言。IBM个人计算机Pascal支持用单元概念分别编译的模块。

一个单元是一组相互有关的数据类型、变量、常量、过程及函数，再加上一初始化过程，单元分成两部分：接口和实现。

接口包含输出用的标识符表及其说明。（包括过程及函数标题在内）。

实现包含那些局部说明、过程及函数体，和初始化代码。某些单元例行程序可以用汇编代码，而不用Pascal实现。

程序（或实现或其他接口）能用一单元作接口，但不能作实现。这种方法提供了更为结构化的方式把程序拆成模块，而不是外部过程和变量。

1.2.3 属性

变量、过程和函数的属性给出对连接程序文本级的控制。

属性包括：

- PUBLIC和EXTERN连接全局标识符。
- STATIC和READONLY对变量，而PURE对过程和函数。

1.2.4 超数组

IBM个人计算机Pascal提供超数组类型，即数组长度是可变的。所谓超数组类型，其下界一定，但上界不定。

虽然超数组类型不能直接用于变量（即，VAR V=SUPER ARRAY [0..*] Of REAL），但有两种重要的使用方法：

·作为形式引用参数类型

·作为指针涉及对象的类型

此外，任何变量均可用一"命名符"，定为超数组类型的导出类型，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);
```

其中VECT是超数组型，PVEC是超数组型指针。

NEW规定了一个新的VECT，括号内的第二个参数给定上界。在上述情况中，上界和参数V的上界相同。

V10是一变量，是上界为10的VECT。参数V可以是任一VECT导出类型的变量。例如V10或PVEC^。

超数组概念以一种清晰、有效方式一并处理动态数组和形态数组。

1.2.5 字符串

在标准Pascal中，字符串长度是固定的。然而，说明长度可变的字符串型变量的能力是BASIC和PL/I以及一种常用抽象数据型的主要特点。字符串型为：

SUPER PACKED ARRAY [0..*] OF CHAR

其中零数组元包含在长度中。还有一组字符串过程和函数。

IBM个人计算机Pascal有CONCAT过程，此过程取两字符串参数，并把第二个和第一个连接起来。

这就使字符串处理具有可移动性，所有过程和函数可用下列任一种Pascal写成：

·说明字符串类型是一固定长度字符数组。

·采用可变长度字符串类型，超数组或其他形态数组扩充。

IBM个人计算机Pascal自动进行字符串赋值，比较和READ/WRITE。

1.2.6 常量值

IBM个人计算机Pascal的另一组特征用于常量值。具有常量值的大部分表达式在编译时估算。这样，如果WORDCOUNT和SYMBOLCOUNT是常量，就能定义另一常量TOTALCOUNT=WORDCOUNT+SYMBOLCOUNT。

数可按二进制、八进制、十进制或十六进制。READ和WRITE都支持。

如同数组常量和记录常量一样，有一字符串常量连接算符。

一程序能包含一个或多个VALUE段，在此段中给变量一初始常量值。

最后，形参前面可有关键字CONST，就和变量前用VAR一样。期望实参只能是一常量，且在例行程序体内不能再修正。

1.2.7 系统实现

对系统实现工作，有几种具体扩充：

- WORD型
- ADDRESS型
- 输入/输出功能
- 交互式READ
- 随机文件和文件模式
- 内部过程和函数

最重要的是WORD型。事实上正好是0到65535（MAXWORD）的子域；而INTEGER型的子域正好是－32767到32767（MAXINT）。

可以把第十六位看成符号位，以区别带符号或不带符号的量。这样，数的量程事实上是从－32768到65535。

WORD型和INTEGER型虽不是一个类型，但都允许把这个量程（除－32768以外）用在Pascal程序中。

在系统实现工作中，常用不带符号的字，诸如地址，或具有一很大的最大值的整数，或一组二进制位，或正好是带有未知语义的存贮器值。

为了这些目地，试图把INTEGER型用到比较问题中（#FFFF 比#FFFE来得大）。

其他算符，诸如AND，OR，XOR也允许是WORD型的。

还加上地址型，类似一指针，但允许分段寻址，这对访问系统数据区很有用。

已增加的其他特征包括各种输入/输出功能（I/O错误陷阱，采用“懒散估值”的交互式READ，随机文件，文件模式，字符串READ等）和附加的内部过程和函数（诸如ENCODE和DECODE，RETYPE和RESULT）。

1.3. 总结

IBM个人计算机Pascal能用于系统软件实现，它包括很多特征，用于按结构方式产生和维护Pascal程序，和/或为公用系统程序设计任务所必需。

特别值得注意的特征是包含了分离的编译单元，可变长度字符串，超数组类型和面向机

器结构。

IBM个人计算机不只是一个工具，而是一个工具制造者。设计成能写更巧妙的程序，使计算机易于使用和更能被人们所接受。

第2章 Pascal 程序的编译

2.1、准备

为在IBM个人计算机上顺利编译Pascal程序，必须准备好：

· Pascal软件包。有三片5¼吋主盘片，分别标为PAS1，PAS2和PASCAL.LIB，其中

① PAS1包含文件：PAS1.EXE，PASKEY，FILKQQ.INC，FILUQQ.INC和ENT6XS.ASM。

② PAS2包含文件：PAS2.EXE。

③ PASCAL.LIB包含文件：PASCAL.LIB和PASCAL。

· 本手册：IBM个人计算机Pascal参考手册。

· 至少有128K字节的驻机存贮器。

· 两台软盘驱动器。

· 壹台打印机。

· 壹台显示器（IBM个人计算机单色显示器，监视器或带有射频调制器的电视机）。

· IBM个人计算机磁盘操作系统（DOS）参考手册和盘片。

• 壹片5¼时空白盘片，以后称作暂存盘片。

2.2. 盘片安排

2.2.1 复制主盘片备份

用三个5¼时空白盘片分别复制Pascal主盘片，得到PAS1，PAS2和PASCAL.LIB的备份。以后用到的盘片都是指备份盘片。请将主盘片妥善保存。

2.2.2 PAS1和PAS2盘片安排

将DOS盘片上的COMMAND.COM复制到PAS1和PAS2盘片上，因为装入PAS1和PAS2时，它们要把COMMAND.COM写入内存（系统启动时，装入COMMAND.COM）。

于是，无论是PAS1还是PAS2，完成执行，且仍处于软盘驱动器内时，会自动重装入COMMAND.COM。

2.2.3 PASCAL.LIB盘片安排

将DOS盘片上的连接程序LINK.EXE复制到PASCAL.LIB盘片上。

2.3. 启动编译

启动编译前的准备步骤为：

1. 暂存盘片格式化。有关格式化的资料请参阅IBM个人计算机DOS参考手册。

2. 将待编译的Pascal程序放到草稿盘片上。有两种做法。一是从原在盘片复制，二是用行编辑程序（参阅IBM个人计算机DOS参考手册）直接在暂存盘片上产生。

3. 将待编译的Pascal程序添上文件名后缀".PAS"。表示用Pascal语言写的程序。

这样，为Pascal程序的编译作好了准备。

注意：输入编译命令时，可全用大写字母，或全用小写字母或两者混用。

2.3.1 用PAS1开始编译

PAS1是编译程序的第一遍编译。PAS1读源文件，并检验其语法的正确性。它产生两个中间文件，存在暂存盘片上（原文件未占用）的空白空间。这些文件是：

PAS1BF.SYM—符号表

PAS1BF.BIN—中间二进制代码

PAS1还产生源清单文件。如果系统有打印机，建议打印一份源清单，便于查错。

PAS1部分的编译步骤为：

1. 将空缺驱动器改为B，输入：

B:

2. 把包含源程序的暂存盘片放入B驱动器。

3. 把PAS1盘片放入A驱动器。

4. 输入：

A: PAS1

PAS1被装入计算机。待一会儿，编译程序显示标题和下列提示：

source filename [.PAS]: —

source filename是指存放待编译程序的文件的名。例如：

source filename [.PAS]: myfile

不必输入文件名后缀.PAS。因为编译程序会自动添上的。如果给文件名加上别的后缀，则就用那个后缀。输入源文件名后，将看到下列提示：

Object filename [MYFILE.OBJ]: —

Object filename 是给编译后将产生的目标文件（机器可读）取的名。如果同意取为MYFILE.OBJ（即方括号内所表示的任何名），则只需要简单地按下 "Enter" 键；否则，重新取名。别忘了加上文件名后缀.OBJ。本例中，假设只按下 "Enter" 键：

Object filename [MYFILE.OBJ]:

接着提示：

Source listing [NUL.LST]: —

source Listing是给包含编译源 清单的文 件取的名。要是不想要清单，则按 "Enter" 键。空缺文件名为NUL.LST。它告诉编译程序，不产生源清单文件。

本例中，假设要一份清单，则输入：

source listing [NUL.LST]: myfile

注意：编译程序自动添上文件名后缀.LST。

最后提示：

Object listing [NUL.COD]: —

Object listing是给包含反汇编目标文件清单的文件取的名。本例中，假设应答：

Object listing [NUL.COD]: myfile

注意：编译程序将添上文件名后缀.COD。

本例的整个屏幕显示内容为：

Source filename [.PAS]: myfile

Object filename [MYFILE.OBJ]:

Source listing [NUL.LST]: myfile

Object listing [NUL.COD]: myfile

一旦输入上列最后一个文件名后，编译程序立即将源程序通过第一遍编译。源程序中若有语法错误，编译程序就会把错误显示在屏幕上，以及清单文件中（参阅本章末的 "编译清单"）。

第一遍完成后，编译程序显示它所找到的错误次数和警告次数。例如，送出的源清单文件中有如下信息：

Pass One No Errors Detected

出错时，根据不同情况，显示下列的任一行或二行：

Errors Detected

Pass One Had Warnings

如果编译程序确实发现错误，则一定要在源程序中找到并解决这些问题，重新运行PAS1，然后再用PAS2继续编译。

如果PAS1盘片上没有复制COMMAND.COM，则计算机会请求插入DOS盘片。

2.3.2 用PAS2继续编译

PAS2是Pascal编译程序的第二遍编译。在此期间，编译程序读PAS1形成的.SYM和.BIN文件，并产生两个目标文件.COD和.OBJ。第二遍是优化。

PAS2产生，写，读和删去一个所谓PASIBF.TMP（中间连接文本）的文件，同样还读和删去PASIBF.SYM和PASIBF.BIN文件。

有些程序在PAS1期间能正确编译，但在PAS2期间，可能产生错误。这些错误包括：

- 超出存贮器
- 超出量程
- 溢出

完成修改（如有必要）后，重新运行PAS1，确保不再存在句法错误，也就为完成编译作好了准备。

具有PAS1文件的暂存盘片应放在B驱动器内。

PAS2部分的编译步骤如下：

1. PAS1盘片取出A驱动器。
2. PAS2盘片放入A驱动器。
3. 输入：

```
A: PAS2
```

PAS2不要求输入别的，它将产生目标文件和清单。编译完成时，给出下列信息：

```
Code Area Size = #0116 (278)
Cons Area Size = #005E (94)
Data Area Size = #000E (14)
Pass Two No Errors Detected。
```

Code Area Size 是程序占据的总字节数（本例中为 278字节）。Cons Area Size是程序中常量（数组，字符串，结构，REAL 等）占据的字节数。Data Area Size 只标STATIC分配的数据。该区总是以偏移量#2开始。这三个区的大小同时用十进制和十六进制表示。

如果在第二遍期间，检测出错误（参见本册附录A“信息”），则加以改正，然后，如有必要，再重新运行PAS1和PAS2。

2.3.3 连接

有关连接的解释，请参阅IBM个人计算机DOS参考手册。

连接步骤：

1. 从A驱动器取出PAS2盘片。
2. PASCAL.LIB盘片（上面已复制好连接程序）放入A驱动器。
3. 输入：

```
A: Link
```

引导连接程序，给出提示：

Object Modules:

输入目标文件（不是目标清单文件）名，在此，不需要后缀.obj。如用举例中的文件名，则输入：

Object Modules: myfile

接着提示：

Run File:

输入给包含程序可执行代码的文件取的名。将给定此文件名的后缀为.exe，并将它放入空缺驱动器（B）。例如：

Run File: myfile

下一提示为：

List File [MYFILE.MAP]:

若按下 "Enter" 键，则连接程序选用空缺名MYFILE.MAP。此文件包含连接程序映象。将被放入空缺驱动器（B）。

下一提示为：

Libraries [ ]: —

所谓库文件是指按Pascal运行程序时所必须的运行时例行程序。全部例行程序都包括在PASCAL.LIB内。为应答此提示，可按 "Enter" 键：

Libraries [ ]:

当连接Pascal程序时，自动带上Pascal库文件。虽然并不必要，但也可以输入：

a: Pascal.lib

接下来五个提示是：

Publics [No]:

Line Numbers [No]:

Stack size [Object file stack]:

Load Low [Yes]:

DSAllocation [No]:

如果对 "Publics" 和 "Line Numbers" 输入 "Y"（即Yes），则这些清单将在MYFILE.MAP文件中出现。

若想对 "Publics" 和 "Line Numbers" 选空缺值，则只需按 "Enter" 键来应答两个中的每个提示。

除对 "Stack size" 和 "DSAllocation"（是用Pascal设置的）的空缺应答外，编译程序将忽略任何选择。

连接程序将对DSAllocation设置自动应答 "Y"。因此，允许Enter（空缺应答）作为一有效回答。

对 "Load Low" 的回答一定是 "Y"，这是空缺应答。所以，在此，Enter回答也是有效的。

本例的全部屏幕指示为：

B>a:link