APPLE II 资料(八)

APPLE PASCAL 语言参考手册

APPLEUII
APPLEPASCAL

语言参考手册

目　录

第一章 引 言

§1.1 了解启动过程

假如你还不知道怎样启动 Apple Pascal 操作系统来使用 Apple Pascal 语言，那么，如果你具有一个盒式磁盘(diskette)驱动器的话，请阅读附录 D。如果你有二个以上的盒式磁盘驱动器，就请阅读附录 E。每个附录都是一个指导性部分，包括系统启动，盒式磁盘初始化，盒式盘的复制，以及 Apple Pascal 程序设计的例子。

§1.2 本文的范围

本文阐述了与 Jensen 和 Wirth 在 Pascal User Manual and Report (Springer Verlag New York, 1978) 中所定义的“标准 Pascal”有所不同的 Apple Pascal 程序设计语言的物征，包含了在 UCSD Pascal 中所介绍的差别，以及在 Apple 计算机上的 UCSD Pascal 的专用扩充体。

Apple Pascal 系统设施：如编辑程序，连接程序等，在 Apple Pascal 操作系统参考手册中说明，这些设施除了对 Apple Pascal 程序设计外，还适用於许多其他用途。在这里，仅当它们明确地涉及到 Apple Pascal 程序时才予以讨论。

§1.3 怎样使用这本书

为了使用这本书，你必须具有完整的标准 Pascal 或 UCSD Pascal 的知识，或者借助于某些完整地描述了标准 Pascak 或 UCSD Pascal 的书或手册。本书是参考手册，目的只是告诉你一些事，而不是着重教你 pascal。

你还应该有一本 Apple Pascal 操作系统参考手册。这本书给出了各种系统设施一个完整的报告，而这些设施是 Apple Pascal 程序建立和开发的一个支撑。

本手册描述了 Apple Pascal 操作系统的一个方面：就是当你要用 Apple Pascal 程序工作时，启动系统的过程，附录 D 和 E 描述了这些过程。

§1.3.1 结构安排

第二、三章阐述了在 Apple Pascal 中主要差别，这些差别对程序设计的效果有着直接的影响。既在予定义型、过程以及函数，特别是输入输出过程等上的差别。

第四章阐述了十分有用的，并且是很重要的编译程序操作和编译程序任选项。在《Apple Pascal 操作系统参考手册》中对编译程序操作进行了更详细的描述。

第五章阐述了将一个程序折成一些分离块的技术，这些块是可以连接在一起的。这些技术是另一个重要的差别，但小程序不需要这些技术。

第六章给出了语言中其余的差别，这些差别对绝大部分程序影响很小。

第七章包含了功能很强的 Apple Pascal 的库任选项，包括龟（Turtle）图形色。

附录 A 提供了一个具有完全注解的图形程序，也描述了用 Apple Pascal 书写的示范程序。

附录 B 包含与 Apple Pascal 语言和系统有关的各种表格。

附录 C 给出了有关正文文件输出/输入操作的技术细节。

附录 D 和 E 涉及系统的启动和使用 Apple Pascal 语言的基本操作过程。

附录 F 是一组完整的 Apple Pascal 语言的语法图。

§1.3.2 本手册中使用的符号

在语法描述中，有以下约定：

——方括号[ ]被用来括住那些可合法地从语法中省去的部分。

§1.4 APPLE Pascal 和标准 Pascal 之间的差别

主要差别概括如下，其它不重要的差别见第 6 章。

§1.4.1 予定义变量类型

——STRING，由一组新的内部构造过程和函数提供的一种新变量类型。见第 2、3 章

——INTERACTIVE，由扩充文件 I/O 过程提供的一种新文件类型。见第 2、3 章。

——对 SET 类型稍有限制。

——对 PACKED 变量处理的一定差别。自动的 PACK 和 UNPACK 操作，以及取消标准 Pascal 的 PACK 和 UNPACK 过程，见第 2 章。

——INTEGER 类型的扩充称为 LONG INTEGER。LONG INTEGER 是由长度可达 36 位的二进制编码的十进制数（BCD）所表示的值，见第 2 章。

§1.4.2 内部构造过程和函数

和在系统库配所提供的专用函数不同，这些过程和函数是 Apple Pascal 语言本身的一部分。内部构造过程和函数简称“内部构造”（Built—ins）。

——提供 STRING 变量的新的内部构造，见第 2、3 章。

——文件 I/O 内部构造的扩充定义，提供了 INTERACTIVE 文件，见第 2 章和第 3 章。

——一组新的，面向字节的内部构造，见第 3 章

——替代标准 Pascal 中 DISPOSE 过程的被称为 MARK 和 RELEASE 的新的内部构造参见第 3 章。

——其它新的内部构造和标准 Pascal 内部构造的重定义，参见第 3 章。

——超越函数 SIN，COS，EXP，ATAN，LN，LOG 和 SQRT 不是 Apple Pascal 中的内部构造，它们被做为库函数提供，参见第 7 章。

§1.4.3 程序的分段

——SEGMENT 过程和函数，仅当它工作时才进入内存，参见第 5 章。

——UNITS 是一组被各别编译的过程，这些过程能通过一个库设施汇集进某一个宿主程序，参见第 5 章。

——EXTERNAL 过程和函数，它在一个 Apfle Pascal 程序中说明，但以汇编语言来实现，然后通某个库设施汇集进一个宿主程序，参见第 5 章。

§1.4.4 Apple 的特殊单元（UNITs）

——这些是 Apple 的主要设施，它们被做为系统库的单元（UNITs）。来实现，包括高清晰度的 Apple 彩色显示器的龟图形包，参见第七章。

第二章　予定义类型

除标准 Pascal 的予定义类型（REAL，INTEGER，CHAR，ARRAY 等）以外，Apple Pasacl 还有 STRING 类型、INTERACTIVE 文件类型和 LONG INTEGER 类型，还有不同标准 Pascal 的某些其它予定义类型的细节。

§2.1　STRING 类型

Apple Pascal 具有一种新的予定义类型 STRING。STRING 变量的值是一个字符的序列。STRING 类型的变量实质上是其元素(字符)个数动态变化的 PACKED ARRAYs OF CHAR。然而，STRING 变量的值不能赋给 PACKED ARRAY OF CHAR，PACKED ARRAY OF CHAR 的值也不能赋给 STRING 变量，串由一组内部构造过程和函数来提供，参见第 3 章。

任一时刻在一个串中字符的个数就是串的长度。在说明缺省的情况下，STRING 变量的最大长度被自动处理为 80 个字符。但这可在 STRING 变量的说明中作出改动（绝对限制为 255 个字符）。为此，将所需的最大长度放在类型标识符 STRING 后的方括号中。以下是 STRING 变量说明的例子：

TITLE：STRING（*最大长度的补缺为 80 字符*）。

NAME：STRING[30]；（*允许串的最大长度 30 字符*）。

STRING 变量的值可以被一个串常量或另一个 STRING 变量通过使用一个赋值语言来更换：

TITLE：='THIS IS A TITLE' 或

NAME：=TITLE

或用在下章中讲到的 READ 过程来更换：

READLN（TITLE）

或用也将在下章中讲到的 STRING 内部构造来更换：

NAME：=COPY（TITLE，1，30）

注意串常量不能包含一个行结束符，并且必须出现在程序中的一个单独的行上。

串中的每一个字符被从 1 到串的 LENGTH 加以索引。LENGTH 是一个将在第 3 章中描述的内部构造函数。例如：假如 TITLE 是一个串的名字，则

TITLE[1]

指出 TITLE 的首字符

TITLE[LENGTH（TITLE）]

指出 TITLE 的最后一个字符。

一个 STRING 类型的变量可不考虑当前的动态长度，而与其它 STRING 类型的变量

比较，或与一个串常量比较。这种比较是按字典顺序的，即如果一个串在一个按字母表顺序排列的串表中先出现，则这个串"小于"另外的串。ASCII 字符集的字符次序（见附录 B）决定这个字母表顺序，以下程序是一个包含对 STRING 类型变量进行合法比较的例子。

```
PROGRAM COMPARESTRINGS;
VAR S: STRING;
    T: STRING[40];
BEGIN
S:='SOMETHING';
T:='SOMETHING BIGGER';
IF S=T THEN
WRITELN('Strings do not work very well')
ELSE
IF S> T THEN
WRITELN(S, 'is greater than', T)
ELSE
IF S< T THEN
WRITELN(S, 'is less than', T);
IF S='SOMETHING'THEN
WRITELN(S, 'equals', S);
IF S>'SAMETHING'THEN
WRITELN(S, 'is greater than SAMETHING');
IF S='SOMETHING    'THEN
WRITELN('BLANKS DON' 'T COUNT')
ELSE
WRITELN('BLANKS APPEAR TO MAKE A DIFFERENCE');
S:='XXX';
T:='ABCDEF';
IF S > T THEN
WRITELN(S, 'is greater than', T)
ELSE
WRITELN(S, 'is less than, 'T)
END
```

以上程序产生以下输出：

```
SOMETHING is less than SOMETHING BIGGER
SOMETHING equals SOMETHING
SOMETHING is greater than SAMETHING
BLANKS APPEAR TO MAKE A DIFFÈRNCE
XXX is greater than ABCDEF
```

串也能被说明为常量，例如：

```
PROGRAM BAZ;
CONST SAMMY='Hi there, I' 'm Sammy the String! ';
BEGIN
WRITELN(SAMMY)
END.
```

STRING 变量的运用将在下一章关于 Apple Pascal 的内部构造过程和函数部分中讨论。

对 STRING 类型变量所加的索引不能超过它的当前动态长度。以下序列将导致一个无效索引运行时刻错。

```
TITLE:='1 2 3 4
TITLE[5]:='5'
```

对长度为零的串要谨慎处理。如果对它们进行索引，将会导致意想不到的结果或引起运行时刻错。如果一个程序要索引一个长度可能为零的串，它应该首先用 LENGTH 函数检查一下是否其长度为零。假如长度是零，则此程序不应处理索引该串的语句。有关 LENGTH 函数的细节参见第 3 章。

〈注意〉

仅包含一个字符的字符串的值和 CHAR 类型的值不是一回事。串和 CHAR 是不同的数据类型，仅包含一个字符的串常量确实具有和 CHAR 类型常量相同的形式，这是一个例外这样的一个常量既可作 CHAR 类型的值，又可当作串的值来使用。

你不能自己定义一个 STRING 类型的函数，在下一章中将描述 STRING 类型的内部构造函数。

§2.2 文 件 类 型

§2.2.1 术语的说明

对于每一个在 Pascal 程序中说明的，取名为 F 的文件，都有一个被自动说明的名为 F^ 的变量。这是文件的“缓冲变量”。有些 Pascal 手册也用不很确切的术语“窗口”来描述可把不同的文件记录装入缓冲区变量的方法。而在本手册中，用与每个打开文件相关的一个“文件指针”来代替。文件指针指向文件中的一个记录。该记录称为“当前记录”。应该知道文件指针并不是一个 Pascal 的 POINTER 变量，而只是一个讨论文件记录的简便方法。

以下部分描述 Apple Palcal 的特有的文件特征，INTERACTIVE 文件类型，无类型文件，予定义文件，以及字符文件的一种特殊形式。

§2.2.2 INTERACTIVE 文件

象 TEXT 文件一样，INTERACTIVE 文件也是一个字符文件，差别在于 RESET，READ 和 READLN 过程对 INTERACTIVE 和 TEXT 文件的处理方式不同。

当一个 Pascal 程序从一个 TEXT 文件读（READ）字符时，首先必须用 RESET 打开此文件，RESET 自动地完成一个 GET 操作，即它把文件的首字符送入文件缓冲区变量，然后将文件指针推进到指向下一个字符，接下去带有 CHAR 类型变量的 READ 或 RE-

ADLN 开关作用时，首先取出已在缓冲区变量中的字符，然后执行 GET。

假如文件是 INTERACTIVE 类型而不是 TEXT 类型，则打开文件用的 RESET 不执行 CET 操作的缓冲区变量未被定义，文件指针指向文件的首字符，而不是第 2 个，因此，后面的 READ 或 READLN 开始作用时，必须首先执行 GET，然后取出由 GET 放入缓冲区变量的字符，这与用于 TEXT 文件的 READ 执行顺序正好相反。

使用 INTERACTIVE 类型的根本原因是假如一个文件不是盒式磁盘文件，而是代表了一个象键盘那样的设备，则只有到一个字符被打入后才有可能对它进行 GET 操作，假如 RESET 试图执行 GET，则直到一个字符被打入程序才继续向后执行，而对于 INTERACTIVE 类型，直到程序执行 READ 或 READLN 时才执行 GET，标准予说明文件 INPUT 和 OUTPUT 是与控制台键盘和荧光屏相关的 INTERACTIVE 文件，另一个称为 KEYBOARD 的予定义文件也相关于键盘的（见后面予定义文件部分）。

§2.2.3 无类型文件（UNTYPED FILES）

除标准文件类型和 INTLRACTIVE 类型外，Apple Pascal 允许“无类型”文件——这种文件只要用字 FILE 来说明即可，而不需任何其它说明。

例如：

VAR F: FILE;

无类型文件仅能用于为实现高速数据传送的内部构造函数 BLOCKREAD 和 BLOCKWRITE。

一个无类型文件可被看成是一个无缓冲区变量 F^ 的文件，这种文件的所有 I/O 必须由 BLOCKREAD 和 BLOCKWRITF 完成，这些函数将在下一章中描述。

§2.2.4 予定文件（PRDEFINED FILES）

标准予定义文件 INPUT 和 OUTPUT 分别涉及键盘和荧光屏。除此以外，Apple Pascal 还提供一个称为 KEYBOARD 的予定义文件，INPUT 和 KEYBOARD 之间的差别在于，当使用 INPUT 而涉键盘时，被打入的字符自动显示在荧光屏上，而使用 KEYBOARD 时，不自动显示字符，这就使得 Pascal 程序可以完整的控制对用户打入字符的响应。

这三个予定文件都具有 INTERACTIVE 类型，并且当 Pascal 程序开始执行时，它们都是通过 RESET 自动地打开。

§2.2.5 正文文件（TEXTFILES）

Apple Pascel 系统提供了一种 TEXT 类型或 INTERACTIVE 类型盒式盘文件，这种文件是通过将“.TEXT”做为其标题的最后一部分而形成的，并且具有一种特殊的内容格式，这种文件在本手册中被称为“正文文件”，不要将正文文件和具有 TEXT 类型或 INTERACTIVE 类型，但标题后没有.TEXT 的文件搞混淆。

Pascal 系统所有涉及字符文件的部分（例如编辑程序）都是使用专门的正文文件格式假如一个 Pascal 程序访问一个正文文件，则此 Pascal 程序也应使用这种专用格式，因此无论何时，你要建立一个 Pascal 的 TEXT 类型或 INTERACTIVE 类型的盒式盘文件，通常的过程是使用一个以“.TEXT”结尾的标题。正文文件的格式如下：

文件的一开始是一个 1024 字节的标题页（header page），它包含正文编辑程序使用的信息，这块空间与系统的各个部分都是有联系。当用户 Pascal 程序建突一个正文文件时，（通过 REWRITE），系统将自动地建立标题（header），当用户 Pascal 程序访问一个现

存的正文文件时（通过 RESET），系统跳过标题，也就是说，对于使用 REWRITE 和 RESET 的用户 Pascal 程序来说，标题是看不见的。

<注意>

当一个程序使用 BLOCKREAD 和BLOCKWRITE 来访问文件时，它就不按专门的正文文文件结构进行。

仅当从盒式盘转输时，系统才将标题一同传送。如果是传送到一个串行设备，标题将被省去不传（因此从盒式盘打印机或控制台的传送省去标题）。

在标题页的后面，是正文本身的内容，以1024个字节为一页，且由若干页组成，每个正文页是一个行的序列，最后一行后面填上足够的空字符，（ASCII00）以填满 1024 个字节，立的定义如下：

[DLE indent][正文]CR

这里方括号的意思是 DLE 和 indent 可被省略，正文本身也可被省略，CR是“脱架返回（Carriage Return）”控制字符，如果是在文件的最后一行的末尾，则此符号可省去，DLE 的“数据传送换码（Data Link Eascape）”控制字符（ASCII 16），假如出现的 DLE 后面跟一个指出行的空格的代码，这个代码是 32＋空出的空格数目，因此行的任何开头单无被 DLE 和 indent 码替换。

DLE 和 indent 代码以及在正文页末端的空字符象标题一样，对 Pascal 程序来说是不可见的，DLE 和 indent 被自动地转换成行开关的空格，反之亦然。

文件的结尾用 ETX 控制字符（ASCII 3 ）标明。

§2.3 集 合 类 型

Apple Pascal 提供了标准 Pascal 的所有集合结构，但对集合增加了两点限制：

——一个集合的元素个数不能多于 512 。

——一个集合所具有的整数（INTEGERs）不能小于 0 或大于 511。

一个 512 元素的集合占用内存 32 个字。

对于集合的比较和运算仅允许在元素类型相同的集合之间进行。假如，在以下这个样板程序中，集合 S 的基类型（base typd）是子域类型 0..49，而集合R 的基类型是子域类型 1..100。两个集合的基础类型（underlying type）都是 INTEGER，这样以下程序中的对于集合 S 和 R 的比较及运算是合法的。

```
PROGRAM SETCOMPARE;
VAR S: SET OF 0..49;
    R: SET OF 1..100;
BEGIN
   S: =[0, 5, 10, 15, 20, 25, 30, 35, 40, 45];
   R: =[10, 20, 30, 40, 50, 60, 70, 80, 90];
   IF S=R THEN
   WRITELN('…oops…')
   ELSE
```

```
WRITELN('sets work');
    S:=S+R
END.
```

在下面的例子中，比较I=J是非法的，两个集合各自具有不同的基础类型。

```
PROGRAM ILLEGASETS;
TYPE STUFF=(ZERO, ONE, TWO);
VAR I: SET OF STUFF;
      J:  SET OF 0..2;
BEGIN
      I:=[ZERO];
    IF I=J THEN…〈〈〈〈错在这里。
END.
```

§2.4 紧凑变量(PACKED VARIABLES)

§2.4.1 压缩和拆开

Apple Pascal不需要标准Pascal的PACK和UNPACK过程，因此未提供这些过程假如一个变量是紧凑的，则所需的所有压缩和拆开是逐个元素自动进行的。

§2.4.2 紧凑文件(PACKED FILES)

Apple Pascal不提供紧凑文件类型，可以说明一个紧凑文件，但文件中的数据实际上并不是紧凑的。

§2.4.3 紧凑数组(PACKED ARRAYS)

Apple Pascal编译程序提供了在标准Pascal中定义的紧凑数组，例如，考虑如下说明：

```
A: ARRAY[0..9]OF CHAR;
B: PACKED ARRAY[0..9]OF CHAR;
```

数组A占用内存十个16位(二进制)的字，数组的每个元素占用一个字，而PACKED ARRAY B总共仅占用5个字，因为每个16位(二进制)的字包括两个8个位(二进制)的字符B的每个元素是8位(二进制)长。

紧凑数组不必仅限于CHAR类型的数组，例如：

```
C: PACKED ARRAY[0..1]OF 0..3;
D: PACKED APPAY[1..9]OF SET OF 0..15;
D2: PACKED ARRAY[0..239, 0..319]OF BOOLEAN;
```

紧凑数组C的每个元素只有2位(二进制)长，因为表示域0..3中的值仅需2位(二进制)就够了，因此C仅占用内存的一个16位(二进制)的字，并且其中12位(二进制)未用。PACKED ARRAY D要占用9个字，因为D的每一个元素是一个集合，它最少要用16位(二进制)来表示，象上例中D2那样的紧凑布尔数组PACKED ARRAY OF BOOLEAN的每一个元素仅占用一位(二进制)。

<注意>