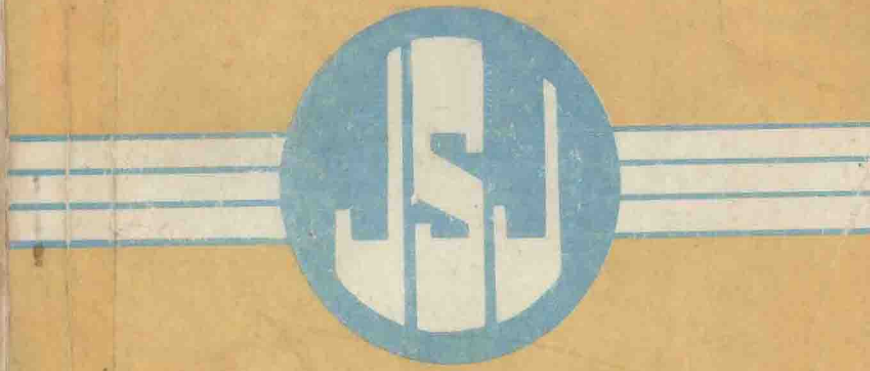
CP/M—86
Pascal/MT + 语言

0520资料出版联合体



CP/M—86
Pascal/MT+语言

译　戴维堤
校　孙辑正
技术校对　殷兆麟

0520资料出版联合体

Pascal/MT+™语言

参考手册

前　言

Pascal/MT＋™语言不仅完全实现了国际标准组织（ISO）公布的标准 Pascal 文本DPS/7185，而且比标准的Pascal扩充了一些功能。这些扩充的功能使Pascal/MT＋更适用于商业上的编程，有利于开发高质量、维护方便的软件。Pascal/MT＋扩充的功能分为下列四类：

- 扩充了输入/输出操作；
- 扩充了数据类型；
- 对运行时的系统进行访问；
- 模块与复盖；

Pascal/MT＋对数据处理应用和实时控制应用都是很有用的。

Pascal/MT＋系统，包括有编译程序、连接程序和一些编程工具。它能在各种操作系统和微处理器上实现。因为这种语言在不同的实现中都是一致的，所以Pascal/MT＋程序在不同目标处理机与不同操作系统之间较容易进行移植。Pascal/MT＋系统可以开发这样的软件，它们既能在以ROM为基础的环境中使用，又可以在有操作系统支持和无操作系统支持下的环境下使用。

这本手册描述Pascal/MT＋语言，着重介绍 Pascal/MT＋的独特的特性。这本手册介绍的所有与语言有关的论题和该语言的实现无关。

关于编译程序、连接程序、Pascal/MT＋编程工具以及与

操作系统有关内容，包括在《Pascal/MT＋语言编程指南》之中，这些内容与用户所使用的Pascal/MT＋语言实现是有关的。

这本手册假定用户已经熟悉了一般的Pascal语言。如果用户还不熟悉Pascal语言，可以参考附录C中列出的教科书的参考文献。

这本手册使用巴科斯范式（BNF）形式描述Pascal语言的语法。如果用户不熟悉巴科斯范式，请参阅附录B。

总 目 录

目　录

附 录

图、表格与表*

图

表格

表*

* 表原文为LIST ING，实际指程序例或程序清单（校者注）

第一节　Pascal/MT＋程序

1.1 程　序　结　构

Pascal/MT＋是一个分程序结构的语言，也就是说，用户能把一个或多个逻辑上有关的语句集中在叫做分程序的块内。每个分程序块有一个首部、一个可选择的说明和定义部份，以及一系列的可执行语句构成。在Pascal/MT＋程序中，最外层的分程序块就是主程序。

在Pascal/MT＋程序内可以嵌套分程序块，也就是说，用户能把一个分程序块放在另一个分程序块内，但嵌套的分程序不能彼此　盖。在分程序块内，过程和函数可以彼此嵌套(见第六节)。图1－1说明Pascal/MT＋典型的分程序块的结构。

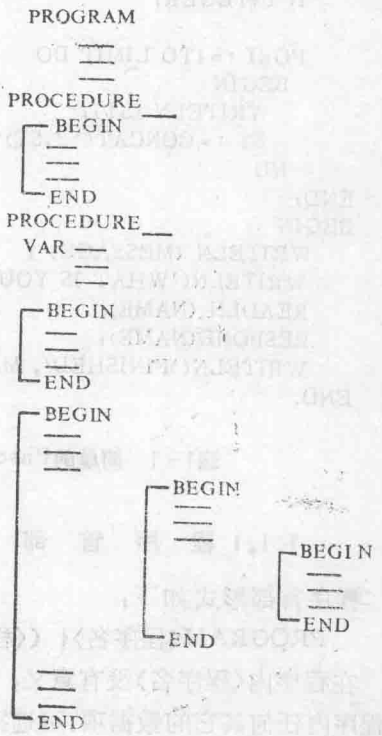


图1－1　在Pascal/MT＋中的分程序结构

表1－1表示包含有一个嵌套分程序块的比较小的 Pascal/MT＋程序。

```
PROGRAM FIRST-1;
CONST
   LIMIT = 10;
   MESSAGE = 'TESTING PASCAL/MT+';
VAR
   NAME:STRING;
PROCEDURE RESPOND (ST:STRING);
 VAR
   I: INTEGER;

   FORI := 1TO LIMIT DO
    BEGIN
      WRITELN (ST);
      ST := CONCAT(' ',ST)'(SHIFTS NAME TO RIGHT')
    ND
END;
BEGIN
   WRITELN (MESSAGE) ;
   WRITELN('WHAT IS YOUR NAME?');
   READLN (NAME);
   RESPOND(NAME);
   WRITELN('FINISHED', MESSAGE)
END.
```

表1－1　简单的Pascal/MT＋程序

1.1.1 程 序 首 部

程序首部形式如下：

PROGRAM〈程序名〉{ (〈程序参数〉) };

在程序内〈程序名〉没有意义，但用户不能使用这个名字作为程序内任何其它的数据项，可选择的〈程序参数〉在Pascal/MT＋中没有特殊的含义这和Pascal的一些其它版本一样。

1.1.2 说 明 与 定 义

程序中使用一个标识符之前，除非该标识符是语言预定义的（见附录A）、否则用户必须定义它。表1－2是一个程序说明与定义部份的例子，其主要说明项如下：

1）LABEL说明

2）CONSTANT说明

3）TYPE定义

4）VAR说明

5）PROCEDURE与FUNCTION定义

注意，LABEL、CONSTANT、TYPE和VAR说明可以为任何顺序，在一个模块内，这四种类型可以多次出现。PROCEDURE与FUNCTION说明必须最后出现，而且每个模块只能有一部份。

第三节介绍了各种数据类型的定义。

```
LABEL
  34, 356, 765, 1000;
CONST
   TOP        = 100;
   BOTTOM     = -TOP;
   LIMIT      = 1.0E-16;
   MESSAGE    = 'THANK YOU FOR NOT SMOKING';
TYPE
   COLOR      = (RED,YELLOW,BLUE,GREEN,ORANGE);
   INDEX      = BOTTOM..TOP;
   PERPt      = ^PERSON;
   PERSON     = RECORD
                  NAME,
                  ADDRESS : STRING;
                  PHONE   : STRING[8]
                END;
```

```
VAR
   COLR : COLOR;
   I,   J : INTEGER;
   LIST  : ARRAY [INDEX] OF PERPT;
PROCEDURE ECHO (ST : STRING);
BEGIN
   WRITELN (ST, ' ', ST)
END;
```

表1－2 说明与定义的程序例

1.1.3 程　序　体

在一个分程序块内，保留字BEGIN和END之间的部分称为程序体，程序体可以包括零个或多个语句。如果这个分程序块是主程序块，那么保留字END之后必须加上一个句点。在程序体内，用分号分隔各个语句。

1.1.4 模　　块

模块是程序独立编译的一个部份，模块编译后再把它连接到主程序去。一个模块的一般形式和一个程序相同，但是模块没有主程序体在模块内仅可执行的代码，应包括在过程和函数内，下面的例子说明一个简单的单过程模块。

```
MODULE SIMPLE;
PROCEDURE MARK (CALL NUM:INTEGER);
BEGIN
   WRITELN( ' IN MODULE SIMPLE, CALLED
FROM: ' ,CALL NUM)
END;
MODEND.
```

注意，保留字MODULE代替PROGRAM，保留字MODEND代替主程序体。

关于模块和模块程序的详细内容，请参考《Pascal/MT＋语言编程指南》

1.2 标识符作用域

Pascal/MT＋程序中的每个标识符，有一定的作用域。所谓标识符的作用域，是程序能够进行合法地引用这个标识符的所有分程序块的集合。一个标识符的正常作用域，是定义它的分程序块，从它实际定义的地方开始。

然而，当一个嵌套的分程序块重复定义同一标识符作为变量时，那么外层分程序的变量是不能引用内层分程序块定义的相同标识符变量。当同一个标识符有多次定义时，最内层的定义确定其作用域。

这本手册使用术语“全局”和“局部”这两个术语。程序中最外层的说明，是全局说明；分程序块内的说明，对这个分程序块说来是局部的。如果一个变量在某一分程序块内说明，那么该变量对该分程序块说来是局部的。在一个嵌套的分程序块内，在外层分程序块内的变量说明对其内层分程序都可用的，但它不是局部于嵌套的内层分程序块的。在内层分程序块内，对外层分程序块内变量的引用，叫做“外层引用”。

表1－3列举了一个程序，它包含嵌套的分程序块对同一个标识符的多次定义。这个程序内的注释，解释各个地方应用的是哪个定义。

```
PROGRAM SHOWSCOPE;
VAR
```