

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 + <sup>TM</sup>语言

参 考 手 册

## 前 言

Pascal/MT + <sup>TM</sup>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—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。

Pascal/MT+ 对数据库应用和实时控制应用提供有用

Pascal/MT+ 系统，包括将编译程序，连接程序和一系列

它在各种操作系统和微型计算机上实现。因为这种

目标语言与不同操作系统之间容易进行移植。Pascal/MT+

系统可以开发这样的软件，它们能够在以ROM为基础的系统中

使用，又可以在有高级语言支持的操作系统中使用。

关于编译程序，连接程序，汇编程序和Pascal/MT+编程工具以及

# 总 目 录

Pascal/MT + TM语言参考手册

Pascal/MT + TM语言编程指南 — 运行  
在CP/M—86族操作系统下

ASMT + 86TM浮动汇编程序参考手册

# 目 录

<b>第一节 Pascal/MT+ 程序</b> .....	1
1.1 程序结构.....	1
1.1.1 程序首部.....	2
1.1.2 说明与定义.....	3
1.1.3 程序体.....	4
1.1.4 模块.....	4
1.2 标识符作用域.....	5
1.3 注释.....	6
<b>第二节 标识符与常量</b> .....	8
2.1 标识符.....	8
2.2 常量.....	9
2.2.1 数值常量.....	10
2.2.2 字符串常数.....	11
2.2.3 命名常量.....	11
<b>第三节 变量与数据类型</b> .....	13
3.1 类型定义.....	13
3.2 变量说明.....	13
3.3 简单类型.....	14
3.3.1 布尔类型.....	16
3.3.2 字符类型.....	16
3.3.3 整数类型与长整数类型.....	17
3.3.4 实数类型.....	18
3.3.5 字节与字类型.....	18
3.3.6 用户定义的序数类型.....	18

3.3.7	指示器类型	19
3.4	构造类型	20
3.4.1	数组类型	21
3.4.2	串类型	22
3.4.3	集合类型	23
3.4.4	记录类型	25
<b>第四节</b>	<b>运算符与表达式</b>	29
4.1	算术表达式	30
4.2	布尔表达式	31
4.3	逻辑表达式	33
4.4	集合表达式	34
<b>第五节</b>	<b>语句</b>	37
5.1	赋值语句	37
5.2	CASE语句	38
5.3	空语句	39
5.4	FOR语句	40
5.5	GOTO语句	42
5.6	IF语句	43
5.7	REPEAT语句	45
5.8	WHILE语句	45
5.9	WITH语句	46
<b>第六节</b>	<b>过程与函数</b>	49
6.1	过程定义	50
6.2	参数	52
6.3	适应数组参数	55
6.4	预定义函数与过程	56
ABS	函数	62
ADDR	函数	62



10	ARCTAN 函数	63
20	✓ ASSIGN 过程	64
30	✓ BLOCKREAD、BLOCKWRITE 过程	66
40	CHAIN 过程	67
50	CHR 过程	67
60	✓ CLOSE过程	67
70	CONCAT 函数	68
80	COPY 函数	69
90	COS 函数	70
100	DELETE过程	71
110	DISPOSE 过程	72
120	✓ EOLN、EOF 函数	72
130	✓ EXIT 过程	75
140	EXP函数	77
150	FILLCHAR 过程	77
160	✓ GET过程	78
170	HI、LO、SWAP函数	79
180	INLINE 过程	80
190	INSERT 过程	80
200	✓ IORESULT 函数	82
210	LENGTH 函数	82
220	LN函数	83
230	MAXAVAIL、MEMAVAIL 函数	83
240	MOVE、MOVERIGHT、MOVELEFT 过程	84
250	NEW过程	87
260	ODD函数	88
270	✓ OPEN过程	89
280	ORD 函数	90
290	PACK、UNPACK 过程	90
300	PAGE过程	91

84	POS 函数	91
18	PRED 函数	92
38	✓ PURGE 过程	93
78	✓ PUT 过程	93
78	✓ READ, READLN 过程	94
78	✓ READHEX, WRITEHEX, LWWRITEHEX 过程	95
83	✓ RESET 过程	96
88	✓ REWRITE 过程	96
07	RIM85 函数、SIM85 过程	97
14	ROUND 函数	97
87	✓ SEEKREAD, SEEKWRITE 过程	98
87	SHL, SHR 函数	99
31	SIN 函数	100
77	SIZEOF 函数	100
87	SQR 函数	102
87	SQRT 函数	102
87	SUCC 函数	103
08	TRUNC 函数	103
08	TSTBIT 函数、SETBIT 和 CLRBIT 过程	104
84	WAIT 过程	105
28	WNB, GNB 函数	106
88	✓ WRITE, WRITELN 过程	107
78	@ BDOS 函数	110
18	@ BDOS86 函数	110
78	@ CMD 函数	111
88	@ ERR 过程	112
88	@ HLT 过程	112
08	@ HERR 函数	112
08	@ MRK 函数	113
18	@ RLS 函数	113

<b>第七节 输入与输出</b> .....	115
7.1 Pascal/MT+ 输入/输出的基础 .....	115
7.2 常规的输入/输出 .....	117
7.3 INP和OUT数组.....	120
7.4 用户自定输入/输出方式 .....	121
7.5 顺序的输入/输出 .....	126
7.5.1 TEXT 文件 .....	126
7.5.2 打印机输出.....	130
7.6 随机存取输入/输出 .....	132

## 附 录

A. 保留字与预定义标识符 .....	138
B. Pascal/MT+ 语法.....	141
C. 与ISO标准的区别 .....	142
D. 参考文献 .....	144

## 图、表格与表\*

### 图

1-1 在Pascal/MT+ 中的分程序结构 .....	1
7-1 TEXT文件中的一些行 .....	127
7-2 一个文件中的记录 .....	134

### 表格

3-1 预定义数据类型.....	15
4-1 Pascal/MT+ 运算符一览表.....	29

4-2	布尔运算表	33
4-3	逻辑运算符表	33
6-1	预定义函数与过程一览表	57
6-2	设备名	65
6-3	一个TEXT文件EOLN、EOF的值	74
6-4	一个非TEXT文件的EOF值	75
A-1	Pascal/MT+ 保留字	138
A-2	Pascal/MT+ 预定义标识符	139

### 表\*

1-1	简单的Pascal/MT+ 程序	2
1-2	说明与定义的程序例	4
1-3	标识符作用域规则说明的程序例	6
1-4	有注释的程序例	7
3-1	使用集合的程序	24
4-1	集合表达式程序例	36
6-1	FORWARD 说明程序例	52
6-2a	传送给过程的参数	53
6-2b	VALVAR 程序的输出	53
6-3	过程参数程序例	55
6-4	适应数组参数与程序例	56
7-1	文件的输入与输出	120
7-2	用户自定输入/输出方式	126
7-3	TEXT文件处理	130
7-4	打印机输出及数的输出格式	132
7-5	随机文件的输入/输出	138

\* 表原文为LISTING, 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 executable 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。

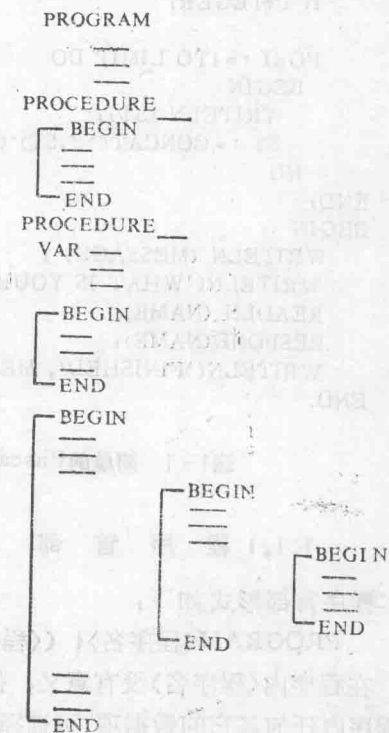


图1-1 在Pascal/MT + 中的分程序结构

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;
FOR I := 1 TO 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.OE-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MODULE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不同标识符变量。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
```