

计算机科学资料

程序语言 PL/I 报告

中国科学院数学研究所计算站

一九七三年五月

引言	1
本语言的目的	1
PL/1的基本特征	2
突出的特征	2
分程序结构	2
数据描述	2
存貯分配	2
数据转换	3
数据组织结构	3
输入/输出	3
多重任务操作	3
编译时设备	4
表处理	4
本手册中的语法记号	4
第一章：程序元素	7
基本语言结构	7
语言字符集	7
60字符集	7
48字符集	8
定义符	8
运算符	8
算术运算符	8
比较运算符	9
位行运算符	9
行运算符	9
圆括号	9
分隔符和别的定义符	9
数据字符集	10
对照序列	10
标识符	10
标识符长度	11
字键	11
语句标识符	11
属性	11
分隔字键	11

内部函数名字	11
随选	12
条件	12
空格的用法	12
注解	12
基本程序结构	13
简单语句	13
复合语句	13
前缀	14
标号前缀	14
条件前缀	14
组	15
分程序	15
END 语句的用法	18
程序	19
第二章：数据元素	20
数据的组织结构	20
纯量项	20
常数	20
纯量变量	20
数据集合	20
数组	21
构件	21
构件数组	22
命名	23
简单名字	23
下标名字	24
数组截面	24
定位名字	25
下标定位名字	26
数据类型	27
问题数据	27
算术型数据	27
实算术型常数	28
虚算术型常数	29
算术型变量	30
行型数据	30
字符行型数据	30
位行型数据	30
行型变量	31

程序控制数据	31
标号型数据	31
语句标号型常数	31
语句标号型变量	31
任务型数据	32
事件型数据	32
指示器型数据	32
指示器定位	32
区域型数据	33
第三章：数据制作	34
表达式	34
纯量表达式	34
算术运算	34
混合特征	34
算术运算的结果	35
算术转换	37
位行运算	37
比较运算	38
连接运算	39
类型转换	39
位行型到字符行型	39
字符行型到位行型	39
字符行型到算术型	39
位行型到算术型	40
算术型到字符行型	40
算术型到位行型	40
数组表达式	40
前缀运算符和数组	41
中介运算符和数组	41
纯量——数组运算	41
数组——数组运算	41
包含构件的数组表达式	42
构件表达式	43
表达式的计算	43
表达式计算的次序	44
第四章：数据描述	45
属性	45
说明	45
显式说明	45
DECLARE 语句	46

提出属性公共因子·····	46
多重说明和不唯一的访问·····	47
上下文联系地说明·····	48
隐式说明·····	51
说明的作用域·····	51
外名字的作用域·····	51
使用名字的基本规则·····	54
属性·····	54
数据属性·····	54
算术型数据·····	54
基底·····	55
标度·····	55
方式·····	55
精度·····	56
算术型数据的缺席条件·····	57
PICTURE 属性·····	57
行属性·····	60
LABEL 属性·····	62
TASK 属性·····	62
EVENT 属性·····	63
维属性·····	64
SECONDARY 属性·····	65
ABNORMAL 和 NORMAL 属性·····	65
USES 和 SETS 属性·····	67
入口名字属性·····	69
ENTRY 属性·····	69
GENERIC 属性·····	70
BUILTIN 属性·····	71
RETURNS 属性·····	72
作用域属性·····	72
存贮类属性·····	73
ALIGNED 和 PACKED 属性·····	75
DEFINED 属性·····	76
定义对应·····	77
定义复盖·····	77
求值次序·····	79
定义的例子·····	79
INITIAL 属性·····	80
LIKE 属性·····	82
文件描述属性·····	84

FILE 属性	84
文件处理属性	85
功能属性	85
PRINT 属性	86
存取属性	86
缓冲属性	86
BACKWARDS 属性	87
EXCLUSIVE 属性	87
ENVIRONMENT 属性	87
KEYED 属性	88
表处理属性	88
AREA 属性	88
POINTER 属性	90
属性赋与标识符	91
缺席属性的应用	92
构件说明和属性	93
层数	93
构件和维属性	93
构件和数据属性	94
构件和作用域属性	94
构件和存贮类属性	94
第五章: 过程, 函数和子程序	95
形式参数	95
过程访问	95
函数访问和函数过程	96
广函数	96
内部函数	97
子程序访问和子程序过程	97
过程访问中的变元	100
ENTRY 属性的用法	100
变元转向入口	101
专用的过程属性 RECURSIVE	102
第六章: 动态程序结构	103
程序控制	103
分程序的启动和终止	103
动态后代	103
动态包括	103
数据分配和存贮类	103
定义和规则	103
存贮类	105

静态存储类	105
自动存储类	105
受控存储类	105
异步操作和任务	107
同步操作和异步操作	107
两个异步操作的同步	108
任务和事件	108
任务的产生	109
任务的终止	109
任务中数据的分配	110
中断操作	110
条件前缀的目的	110
条件前缀的作用域	111
ON 语句的用法	111
系统中断动作	112
REVERT 语句的用法	114
程序员定义的 ON 条件	114
程序检验设备	115
第七章：输入/输出	116
文件开启和文件属性	116
开启文件	117
显式开启	117
隐式开启	117
属性的合并	117
数据流传输	118
表式传输	119
数据式传输	119
编排式传输	119
数据流数据指明部分	119
数据表	119
重复指明部分	120
数据表元素的传输	121
表式数据指明部分	122
表式输入	122
表式输入格式	123
表式输出	124
表式输出域格式	124
数据式数据指明部分	126
数据式传输流中的数据	126
数据式传输数据域长度	129

编排式数据指明部分	130
格式表	131
数据格式项	131
控制格式项	135
空位格式项	135
打印格式项	135
间接格式项	136
数据流传输语句	137
记录传输	138
记录传输语句	139
RECORD 传输操作	140
标准文件	142
第八章：语句	143
语句的关系	143
分类	143
赋值语句	143
控制语句	143
数据说明语句	143
错误控制和找错语句	143
输入/输出语句	143
文件准备语句	144
记录状态语句	144
数据指明语句	144
数据传输语句	144
程序结构语句	144
存贮分配语句	144
控制序列	144
伪变量	146
语句字母顺序表	147
ALLOCATE 语句	147
赋值语句	151
BEGIN 语句	158
CALL 语句	159
CLOSE 语句	160
DECLARE 语句	162
DELAY 语句	162
DELETE 语句	162
DISPLAY 语句	163
DO 语句	164
END 语句	167

ENTRY 语句	168
EXIT 语句	169
FORMAT 语句	169
FREE 语句	170
GET 语句	171
GO TO 语句	172
IF 语句	174
LOCATE 语句	175
空语句	176
ON 语句	176
OPEN 语句	180
PROCEDURE 语句	182
PUT 语句	183
READ 语句	185
RETURN 语句	187
REVERT 语句	188
REWRITE 语句	189
SIGNAL 语句	191
STOP 语句	192
UNLOCK 语句	192
WAIT 语句	193
WRITE 语句	194
第九章: 程序变化	196
宏变量	196
宏DECLARE 语句	196
宏表达式	198
可执行宏语句	198
宏赋值语句	198
宏空语句	198
宏 GO TO 语句	199
宏 IF 语句	199
宏处理机的动作	199
第十章: 专题	201
变元和参数的关系	201
变元下标的计算	201
虚拟变元的用法	201
入口属性的用法	202
参数和变元的对应	203
参数分配	204
参数, 界和长度	204

关于界或长度的星号记法	204
表达式作为界或长度	205
序 (PROLOGUES)	205
跨任务的数据分配 (DATA ALLOCATION ACROSS TASKS)	206
任务名字和事件名字的分配	207
反常性 (ABNORMALITY)	207
表处理 (LIST PROCESSING)	208
基本概念 (BASIC CONCEPTS)	208
附加条件	212
用作基本变量的构件	212
指示器值和基本变量的关系	214
数据链的预防办法	214
附录 1: 内部函数	216
算术型广函数 (ARITHMETIC GENERIC FUNCTIONS)	216
浮点算术型广函数	219
行型广函数 (STRING GENERIC FUNCTIONS)	220
处理数组的广函数 (GENERIC FUNCTIONS FOR MANIPULATION OF ARRAYS)	222
数组内部函数和构件内部函数	223
条件内部函数	223
表处理内部函数	224
其他内部函数	224
附录 2: 图象指明部分表	226
数字指示和定义字符子域 (DIGIT POINT AND SUBFIELD DELIMITING CHARACTERS)	226
删除另字符 (ZERO SUPPRESSION CHARACTERS)	226
漂移编排符号 (DRIFTING EDITING SYMBOLS)	227
漂移字符 (DRIFTING CHARACTERS)	228
编排字符 (EDITING CHARACTER)	228
条件编排字符	228
符号字符 (SIGN CHARACTERS)	229
标度因子指明部分	229
英镑图象 (STERLING PICTURES)	229
关于字符行型图象	230
附录 3: ON 条件	231
条件的分类 (CLASSIFICATION OF CONDITIONS)	231
计算条件	232
输入/输出条件	233
程序检验条件 (PROGRAM CHECKOUT CONDITIONS)	235

表处理条件(LIST PROCESSING CONDITIONS)	238
程序员指定的条件	238
系统动作条件	238
附录 4：允许的字键缩写	239
附录 5：48字符集	240
附录 6：有注释的例子	242
索引	248
表1. 算术型基底和标度转换.....	37
表2. “外名字作用域”例 1 中名字的作用域和用法.....	53
图1. 重复指明部分的一般形式	120
图2. 表式输入转换	123
图3. 数据式输入和数据式输出的例子	130
图4. DO 语句的一般形式	164

引 言

本 语 言 的 目 的

在电子数据处理的整个相对说来还比较短的历史中，一定的计算机是固定地使用在特定的工作领域中的，或者用作商用计算，或者用作科学计算。

程序员一般也固定在某个工作领域。强调了这种分离的高级语言发展成两个不同的方向：一个是商用计算的程序设计语言，一个是科学计算的程序设计语言。

直到最近，这种分工还没有发现什么问题。每一种语言对它的使用者来说都是够用的。商用程序员用到相对来说比较少的计算而处理大量的数据，科学程序员则使用少量的数据而完成复杂的计算。

然而现在情况在变化。商业和工业已经发现了计算机的新用途。商用程序员发现自己要和统计预测以及运筹学线性规划中更复杂的计算打交道。

在科学和工程中，程序员需要一种语言以便简化报告的准备以及分类并编排技术数据。他们发现对输入和输出的操作更需要了。工程师特别需要在诸如电路分析的应用中有能力处理位一级 (bit level) 的数据。

今天已经设计了新的计算系统去应付所有这些计算问题。它们拥有新的能力和新的速度，处理起商用程序和科学程序来是同样地容易。它们对诸如分时数据处理、程序异步执行、实时处理等新技术提供了设备。

然而，传统的高级语言，都无法有效地使用这些新计算机所拥有的全部能力。

PL/1 就是为此而设计的多用途程序设计语言。它不仅能为商用的和科学的程序员所使用，也能为实时的程序员和系统的程序员所使用。这是一个从效率出发而设计的语言，它能使程序员真正使用了他的计算机的全部能力。

PL/1 的组织结构使任何一个程序员，不管他的经验如何，都能按他自己的水平很容易地使用它。

本书是整个语言的参考手册，指出了 PL/1 的能力和范围，它处理最复杂的计算问题的能力。

虽然如此，事实上 PL/1 不会比它用到的程序更复杂。

设计本语言的初始目标之一是可调节性。亦即对不同的应用和不同的复杂程度提供语言的不同子集。程序员使用一个子集，甚至可以不知道他不使用的其他设备。

虽然 PL/1 相对于机器是独立的，但这种可调节性可相比于完整装备起来的数据处理中心。一个程序设计的新手仅使用系统的一小部分，他可以不管余下的其他设备。当然更复杂的程序需要更多的设备，某种程序要用到某种程度的设备，另一种程序要用到另一种程度的设备，要用到全部设备的程序如果有也是绝少的。

在 PL/1 中，变量的每一种属性（或描述），每一种随选和每一种指明部分，都已给出了一个“缺席”的解释。无论什么地方，当语言提供了一个或多个不确定的意思而程序员又

沒有指定选择的话，编译程序就作出“缺席”解释（或“缺席”假设）。在每一种情形之下，被语言的设计所选取的假定是程序员最喜欢要的一种，该程序员不需知道其他可能的存在。

在“可调节性”和“缺席规则”基础上建立了 PL/1 的简易性，它们是语言的能力的一部分。

PL/1 的基本特征

设计本语言的整个目的在于使程序员能自由地处理他的计算系统。

表达的自由度：若记号的特殊组合有着有用的意义，则该意义是允许的。虽然语言中的真正语句必须使用特定的字符集写出。数据可用各个计算机的构形所允许的任何字符组成。PL/1 是以自由域的格式（free-field format）写出，程序员可自由地设计他自己的列表形式。

完全接近机器和操作系统设备：PL/1 程序员需要依靠汇编语言编码的情况，如果有也是绝少的。

突出的特征

本语言的一部分当然是基于早期的程序设计语言。它的某些方面是以前使用的概念的扩充，另一部分则是 PL/1 专有的。以下几节简明地描述一下某些这样的特征，在正文中将完整地讨论它们。

分程序结构

PL/1 程序的语句组成叫作“分程序”的程序节。一个程序可由一个或多个分程序结合而成。分程序可以沒有公共语句而彼此独立，也可以一个嵌套在另一个之中。

分程序提供两种重要的逻辑功能：（1）定义数据变量和别类名字的应用范围。从而同名量在不同的分程序中可使用于不同的目的而不致混淆。（2）仅当执行分程序时允许给数据变量分配存贮而在分程序结束时释放存贮以作他用。

某些分程序叫作“过程”分程序，可以在程序中远离它的不同地点引用它（即调来执行），它提供处理变元和代回值的方法。

数据描述

本语言中，数据被描述成具有叫做属性的某种特征。例如数值数据具有 BINARY 属性或 DECIMAL 属性；行数据或者是 CHARACTER 行或者是 BIT 行。

存贮分配

PL/1 程序中任何数据变量在计算机中的存贮可以在执行程序的整个过程中静态地分

配，也可以在执行时动态地分配。

PL/1 程序员可利用两类动态存贮，即自动的动态存贮和控制的动态存贮。当变量具有控制的存贮属性时，程序员可在他希望的任何时候分配或释放该变量的存贮。具有自动存贮属性的变量的存贮是在分程序入口分配而在出口释放。

数据转换

为保持 PL/1 的自由度，混合表达式是允许的。下例中 F 被说明为定点数，G 为浮点数，而 H 是长度为十个字符的字符行。

```
DECLARE F FIXED, G FLOAT, H CHARACTER (10);  
H = F + G;
```

在计算这个例子的第二个语句时，F 将转换为浮点值，然后进行浮点加，将结果转换为具有十个字符的字符行，最后将它作为值赋给 H。

数据组织结构

数据变量可结合成数组或构件。数组由具有相同特征的元素组成。构件是变量和数组的集合体，这些变量和数组不需要有相同的特征。构件还可包含别的构件。数组的个别项是带下标的名字。构件的个别项是名字，这些名字有时必须加以修饰以避免混淆。

在 PL/1 中数组和构件按他们本身的性质作为变量来处理。它们当中的每一个都可以作为表达式中的运算对象。于是该表达式就是一个数组表达式或构件表达式，其最后结果是一个数组或构件。

输入/输出

在输入/输出设备上 PL/1 的可调节性是特别明显的。一个使用 PL/1 的程序员，可以随他要求控制输入/输出动作。他可以处理通常的传输和简单的转换，或者他可以使用语言的全部能力去控制更复杂的输入和输出问题，

多重任务操作

在 PL/1 中，过程的集合体叫作程序。执行一个程序（或多个程序或一个程序的一部分）去完成一件特定的工作任务（job）叫做一个任务（task）。

PL/1 提供并行处理两个或多个任务的设备。在使用任何具有多重处理能力的计算机系统时，这种设备当然是极为重要的。对于具有实时操作设备的单信息处理系统这种设备也是有价值的。

当执行一个过程时，正执行的任务（executing task）可以指明一个从属的任务开始对某个数据执行（即正执行的任务引用别的任务）。叫作从属任务（attached task）的新任

务也可以引用别的任务。从而所有任务是并行地并从效果上说是同时地进行的。

PL/1的多重任务设备允许从属任务通过变元(argument)和通过在归属任务(attach-ing task)中被分配的数据同它的原来的(originating)或归属的任务通讯。原来的任务也可以在任何时候检查从属任务是否完成了,如果需要可以延搁它自己的执行去等待这种完成。

编译时设备

大多数程序设计语言是仅仅在一个层次之上写出的。象语句这进计算机使用所提供的数
据完成某项操作。PL/1 不仅指示计算机对数据进行操作,同时使用一个宏设备指示编译系
统对程序本身进行操作。

程序员可以把信息包含在他的程序中帮助编译程序产生更有效的代码、文件和诊断。

表处理

PL/1 提供表处理的设备。这些设备当引入指示器变量和基本变量使得程序员能将数
组、构件、纯量结合在一个单一的表中时是非常灵活的。

PL/1 表处理设备完整的列举可在索引的“表处理”一项里找到(也可见第10章“表处
理”)。

本手册中的语法记号

在本手册中,不论何处讨论到一个 PL/1 的语句或元素的某种其他组合时,都是用一
致的记号系统来写出该语句或成语的。

这些记号不是 PL/1 的一部分。它是能够用来描述任何程序设计语言的语法或结构的标
准记号。它提供语言允许的一般模式(Pattern)的一个简洁而严格的说明。它并不描述语
言元素的意义而仅仅描述他们的结构。亦即它指出元素可以出现或必须出现的次序,所要求
的标点,和允许的选择等。

以下的规则说明了这些普适于任何程序设计语言的记号的用法,仅仅例子特别用到了
PL/1:

1. 变量记号是程序设计语言中元素的一个通用类的名字。变量记号必须包括:

a. 小写字母、十进数字、和连字号(hyphen)并且必须以字母开始。

b. 小写和大写字母的组合。其中必须有一部分全是小写字母,一部分全是大写字母,
而这两部分必须用一连字号隔开。

所使用的全部这样的变量在本手册中或者用这种记号形式地进行定义,或者用散文的形
式进行定义。

例:

a. 数字。表示出现一个数字，它可以是 0，1，…，9。

b. 文件名字。表示出现了名字叫做文件名字的变量记号。在本手册别的地方给出了文件名字的解释。

c. DO 一语句。表示出现了 DO 语句。使用大写字母是为了强调起见。

2. 常数记号表示出现了代表字符的一个文字。常数记号或者全由大写字母组成或者用专用字符构成。

例：

DECLARE 标识符 FIXED;

表示出现单词 DECLARE，其后跟着一个变量“标识符”，它将在别处定义，再跟着出现单词 FIXED，最后跟着出现分号（;）。

3. 在后面规则中使用的“语法单位”一词定义为下述两种情形之一：

a. 单个的变量或常数。

b. 由变量、常数、语法语言记号和以花括号、方括号括起来的保留单词所构成的任一集合体。

4. 花括号 { } 表示组合。

例：

标识符 { FIXED
 FLOAT }

垂直叠放的语法单位指出要作一种选择。上例指出变量“标识符”之后必须跟着出现单词 FIXED 或单词 FLOAT。

5. 垂直一竖 | 表示要作一种选择。

例：

标识符 { FIXED | FLOAT }

同前个例子完全是一个意思。本手册中两种方法都采用，表示二者取其一。

6. 方括号 [] 表示随意选择。含在方括号 [] 中的任何事物可以出现一次或完全不出现。

例:

CHARACTER (长度) [VARYING]

表示出现单词 CHARACTER, 后跟一个括在圆括号中的变量“长度”, 后面可随意地出现或不出现单词 VARYING。如果在规则 4 中两个二者取一的选择也是随意的, 则可以垂直叠放在方括号中, 而不需要花括号。

7. 三点...表示它紧前面的语法单位连续出现一次或多次。

例:

[数字]...

变量“数字”因为是括在方括号中, 故它可以出现也可以不出现, 如果出现则它可以重复一次或多次。

8. 当所描述的一个元素同语法语言中的一个元素有冲突的话, 则以下面划线来表示正被描述的元素。

例:

运算对象 { $\mathcal{E}$ | $\perp$ } 运算对象

表示变量“运算对象”或者被“和”(E)或者被“或”(|)隔开。常数 $\perp$ 划了下线是为了区别 PL/1 语言中的“或”记号和语法语言中的“或”记号。

9. min max。这两个字以及同它们连带在一起的数值合在一起指明语法单位可以出现的最小及最大次数。若只用了 min 而无 max 意味着 max 为无穷大, 若只用了 max 而无 min 意味着 min 为另。

例:

a. min 2 max 6 { 数字 | 字母 }

表示用任意组合法混起来的“数字”或“字母”最少出现 2 次, 最多出现 6 次。

b. min 5 { 数字 | 字母 }

变量“数字”或“字母”用任意组合法混起来出现的次数至少 5 次, 而超过 5 次以上出现的次数是沒有限制的。

c. max 3 标号

变量“标号”连续出现的次数不能大于 3。它可以完全不出现, 或者出现一次, 两次或三次。