算法语言 ALGOL 68报告

A. Van. 维思加登 主编

陆汝钤 译

内 容 简 介

ALGOL68是一种新的语言，是在ALGOL60基础上发展起来的。ALGOL68与ALGOL60相比，功能强，速度快，效果好．用在大型机器上，功效尤为显著。

A. Van. Wijngaarden

NUMERISCHE MATHEMATIK

Band 14 Heft 2 1969

算法语言 ALGOL 68 报告

A. Van. 维恩加登 主编

陆汝钤 译

*

科 学 出 版 社 出版

北京朝阳门内大街137号

西安新华印刷厂印刷

新华书店北京发行所发行 各地新华书店经售

*

1977年12月第 一 版 开本：787×1092 1/32

1977年12月第一次印刷 印张：8

印数：0001—11,270 字数：177,000

统一书号：13031·611

本社书号：888·13—1

定价：0.82 元

目　　录

• ▼ •

0. 导　　引

0.1. 设计的目的和原则

a) 在设计算法语言 ALGOL 68 时，国际信息处理协会关于 ALGOL 的 2.1 工作小组相信，一个公共的并为各国人民服务的程序设计语言是有价值的.

b) ALGOL 68 的设计是用来交流算法，在各种不同的机器上执行这些算法，并把它们教给学生.

c) 工作小组的成员根据在 ALGOL 60 及其它程序设计语言方面的多年工作经验，确定其目标如下：

0.1.1. 描述的完整性和明确性

工作小组希望在解决明确而完整地描述一种语言的问题上作出贡献. 本报告采用的方法是基于一种严格的语言，它包含一个语言核心，这个核心的描述被紧缩到最小限度，而语言的其它部份都可用这个核心的术语来描述，其中语义的描述减缩了，相应地增加了语法部分的负担.

我们认识到，这个方法对初学者可能是比较困难的. 因此，应工作小组的要求，C. H. Lindrey 和 S. G. van der Meuler 编写了题为“ALGOL 68 非形式导引”的姊妹作. 关于这个语言的某些方面的有关著作，亦将问世.

0.1.2. 正交设计

为了使语言便于描述、学习和实现，把独立的初始概念的数目缩小到最少程度；另一方面，这些概念又被“正交”地运用，以便尽可能扩大语言的表达能力，而又不致于有害地引进

过多的东西.

0.1.3. 可靠性

在 ALGOL 68 的设计中,几乎所有的语法错误以及许多其它错误都可检查出来,以免导致不良后果,并且犯这些错误的可能性也大大减少.

0.1.4. 质量

ALGOL 68 使程序员可以编写出能在现代计算机上有效运行的程序,而不要求编译系统具有复杂的耗费时间的优化功能. 例如可参看 11.8.

0.1.4.1. 静态模式检验

ALGOL 68 的语法使程序运行时不必进行模式检验,除非是加工一致性关系的情况. 而只有在程序员明显地利用由联合模式功能所提供的灵巧性时,才需要使用一致性关系.

0.1.4.2. 静态作用域检验

ALGOL 68 的语法不要求在程序运行时的作用域检验,除非程序员明显地利用由于在语法上对作用域不加限制而带来的灵活性.

0.1.4.3. 与模式无关的语法分析

ALGOL 68 的语法允许对一个程序进行语法分析而不依赖于它的组成部分的模式. 不止如此,还有一个算法,可以在有限步内确定一个任意的符号序列是否是一个真正的程序.

0.1.4.4. 独立的编译

ALGOL 68 的语法使得主程序和过程可以互相独立地编译而不影响目的程序的质量,其条件是在进行这种独立编译时所有非局部量的模式均已指明. 参看 2.3.c. 后的注记.

0.1.4.5. 循环优化

ALGOL 68 的叠代过程是这样设计的，它通过直接使用熟知的优化技术而在运行时得到很大好处，又不显著增编加译时间.

0.1.4.6. 表示

ALGOL 68 符号的表示是这样选择的，它使得这个语言可以在只有最小的字符集合的计算机上实现，同时编译者也可通过使用较大的字符集而得到好处，如果有这种条件的话.

0.2. 与 ALGOL 60 的比较

a) ALGOL 68 是一种比 ALGOL 60 应用得更广泛、能力更强的语言，虽然受到 ALGOL 60 许多影响，但 ALGOL 68 并非作为 ALGOL 60 的扩充来设计的. 它是一种完全新型的语言，是建立在对于计算过程来说是本质的、基础的概念作透彻分析以及新的描述技术之上的.

b) ALGOL 60 的卓有成效的功能重新出现在 ALGOL 68 中，但它仅仅是作为更一般的结构的特例出现的. ALGOL 68 还有其它一些新的功能. 因此，要说清楚两个语言之间的所有差别是困难的，下面几节将指出它们之间较重要的差别.

0.2.1. ALGOL 68 的值

a) ALGOL 60 只有三种值的类型：**integer**, **real** 和 **boolean**, ALGOL 68 拥有无限多种“模式”，它们是“类型”概念的推广.

b) 一个简单值或者是算术型的（即整模式或实模式，在这种情况下，它们属于几种长度之一），或者是布尔模式或字符模式.

c) 在 ALGOL 60 中，值可以组成数组. 在 ALGOL 68 中，除了这种“多重”值外，还定义和处理“结构”值，它们由可以

是不同模式的值所组成. 多重值的一个例子是字符组，类似于 ALGOL 60 中的字符串;结构值的例子如复数，看作字位和字节序列的机器字以及符号公式.

d) ALGOL 68 引进"名字"的概念,亦即一个值,它"关联于"另一个值，这样一个名字——值的偶相当于 ALGOL 60 的变量. 但每个名字又可以是另一个名字——值的偶中的值,这样,就可以建立起一个间接地址的链.

e) ALGOL 60 中过程体概念在 ALGOL 68 中推广为"子程序"概念,它也含有形式参数. 子程序自身又是一个值，因此可以象其它值一样处理. ALGOL 68 的"格式"概念在 ALGOL 60 中找不到对应的概念.

f) 与简单值或是由简单值构成的多重值或结构值相反，一个名字或子程序或格式的意义,以及由名字,子程序或格式(可能还有其它的值)构成的多重值或结构值的意义，一般说来,与它们出现之处的上下文有关. 因此,使用名字、子程序、格式时自然要受到"作用域"的限制.

0.2.2. ALGOL 68 的说明

a) ALGOL 60 有类型说明、数组说明、开关说明和过程说明,而 ALGOL 68 只有"等同说明"，但它的表达能力比以上所有说明的总和还强. 事实上，等同说明不仅用于说明任意模式的变量,而且还说明任意模式的常数,它还构成一个高质量的和强有力的参数机制的基础.

b) ALGOL 68 还有"模式说明",可以根据已有的模式构造新的模式，特别是多重值和结构值的模式可以用这种方式定义. 此外,还可定义模式的联合，把它用在等同说明中，就可以有这样的名字,使得每一个值,只要它具有这些联合起来的模式中的任意一种模式,都可被此名字所关联.

c）最后，在 ALGOL 68 中有“优先说明”和“运算说明”，利用它们可以引进新的运算符，定义这些运算符的运算以及把已建立的运算符的运算数类加以扩充或修改。

0.2.3. ALGOL 68 的动态存储分配

ALGOL 60（除所谓“固有动态数组”以外）采用“堆阵”式存储分配系统，对单个值或多重值的数目是静态（即在编译时）确定的情况，这种办法已足够。ALGOL 68 比它多一种功能，可以产生一批值，其数量是动态决定的。这种功能需要使用额外的存储分配技术。

0.2.4. ALGOL 68 的并行加工

在 ALGOL 60 中，语句是“依次执行”的。在 ALGOL 68 中，“短语”可以“顺序加工”，也可以“并行加工”，后者在许多情况下可导致质量较高的目的程序，并提高语言的表达能力。引进了平行程序设计的功能，鉴于目前技术水平的限制，只限于一些基本的方面。

0.2.5. ALGOL 68 的标准说明

在 ALGOL 68 中，除包含 ALGOL 60 的全部标准函数外，还有一些其它的标准说明，其中有“环境调查”，利用它可以确定编译系统的某些性质。还有“传输”说明，利用它可在运行时由外部设备得到数据或向外部设备输出结果。

0.2.6. ALGOL 68 的某些特殊结构

a）在 ALGOL 60 中分程序、复合语句以及带括号的表达式等概念在 ALGOL 68 中统一为“闭子句”。一个闭子句可以是一个表达式，并有一个值。与此类似，ALGOL 68 的

“赋值”是 ALGOL 60 赋值语句的推广，它可以是一个表达式，因而也占有一个值.

b) ALGOL 60 的下标化概念在 ALGOL 68 中推广为“指标化”，它不仅可用来从一个数组中选出一个元素，而且可以选出一个子数组. 这个子数组可以同维数，也可以是任一较低维的子数组，其界可以变化.

c) ALGOL 68 不仅提供如 0.2.1.c 中所说的多重值，而且还有“并行表达式”，用来从较简单的值中组成多重值或结构值.

d) ALGOL 60 的循环语句改变为更简洁有力的“重覆语句”.

e) ALGOL 60 的条件表达式和条件语句统一为“条件子句”，但有改进：规定以一个闭符号结尾，其中两个选择子句有相同的语法功能. 同时，条件子句推广为“情况子句”，允许在一组任意数目的子句中根据一个整表达式或一个一致性关系的值进行有效的选择.

f) ALGOL 60 中成效不大的概念（如固有量和整标号）没有收进 ALGOL 68. 还有一些概念，如命名表达式和开关，在 ALGOL 68 中也不出现，它们的表达能力已包含在其它更一般的构造之中.

1. 语言和元语言

1.1. 描 述 方 法

1.1.1. 严格语言、扩充语言和表示语言

a) ALGOL 68 是一种语言，可以用它来构造某种“计算机”，亦即“自动机”或“人”的“程序”. 它通过三个阶段来定义，即“严格语言”、“扩充语言”和“表示语言”.

b) 定义中用到了“英语”和一种“形式语言”. 对于这两种语言以及“严格语言”和“扩充语言”，使用了印刷和语法的某些标记，这些标记与表示语言中所用的标记没有直接关系.

1.1.2. 严格语言的语法

a) 严格语言以“语法”和“语义”定义. 语法是一组“概念”的“产生规则”，它用下述标记表达：“小语法标记”，在本报告中是“a”，“b”，“c”，“d”，“e”，“f”，“g”，“h”，“i”，“j”，“k”，“l”，“m”，“n”，“o”，“p”，“q”，“r”，“s”，“t”，“u”，“v”“w”，“x”，“y”，“z”；

“大语法标记”，在本报告中是“A”，“B”，“C”，“D”，“E”，“F”，“G”，“H”，“I”，“J”，“K”，“L”，“M”，“N”，“O”，“P”，“Q”，“R”，“S”，“T”，“U”，“V”，“W”，“X”，“Y”，“Z”；

“其它语法标记”，在本报告中是“．”(“句号”)，“，”(“逗号”)，“：”(“冒号”)，“；”(“分号”)和“＊”(“星号”).

{注意，这些标记与本句子中所用标记的字体是不同的[1].}

---

1) 所谓“本句子中所用标记的字体”，即普通英文字体.

b）“原始概念”是一个非空的、可能是无限的、小语法标记的序列；概念是一个具有产生规则的原始概念；“符号”是一个以'符号'结尾的原始概念。

c）一个概念的产生规则由下列内容顺序组成：最前面可能有一个星号，概念本身，一个冒号、一个“概念表”{见d}和一个句号。这个概念表称为这个概念的“直接产生式”。

d）概念表是一个非空的、由逗号隔开的“成员”序列；一个成员要末是一个概念，在这种情况下称为产生性的{，或非终点性的}，或一个符号{，这是终点性的}，或者为空；要末是其它某个原始概念{在这种情况下，以此原始概念为其概念表成员的产生规则称为一条“死胡同”}。

e）一个概念的“产生式”要末是这个概念的直接产生式，要末是一个概念表。这个概念表是这样得来的：把这个概念的某个产生式的一个产生性成员换为该成员的一个直接产生式。

f）一个概念的“终点产生式”是这个概念的一个产生式，它的每个成员要末是符号，要末是空。

{在严格语言的产生规则（5.1.2.1. b）

'variable point numeral: integral part option, fractional part.'中，概念表

'integral part option, fractional part' 是概念 'variable point numeral' 的一个直接产生式，它包含两个成员，两个都是产生性的。这个概念的一个终点产生式是 'digit zero symbol, point symbol, digit one symbol'。成员 'digit zero symbol' 是一个符号的例子，它是终点性的。原始概念 'twas brillig and the slithy toves' 既非符号，又非本报告意义下的概念，因为它既不以'符号'结尾，又不存在它的产生规则（1.1.5.b.c.）。}