RELATÓRIO REVISTO SOBRE A LINGUAGEM ALGORÍTMICA ALGOL 60

Descrição da linguagem de Referência

(Conclusão)

3. Expressões.

Na linguagem, os componentes principais dos programas que descrevem processos algorítmicos são expressões *aritméticas*, *booleanas* e *designatórias*. Os elementos destas expressões, com excepção de certos limitadores, são valores lógicos, números, variáveis, indicadores de função, operações de relação, operadores aritméticos, lógicos e sequenciais. Como a definição sintática das variáveis e dos indicadores de função contém expressões, a definição de expressões e dos seus componentes é necessàriamente recursiva.

< expressão > : : =
< expressão aritmética > |
< expressão booleana > |
< expressão designatória >

3. 1. *Variáveis.*

3. 1. 1. *Sintaxe.*

< identificador de variável > : : =
< identificador >
< variável simples > : : =
< identificador de variável >
< expressão em índice > : : =
< expressão aritmética >
< lista de índices > : : =
< expressão em índice > |
< lista de índices > ,
< expressões em índice >
< identificador de tabela > : : =
< identificador >
< variável indiciada > : : =
< identificador de tabela >
[< lista de índices >]
< variável > : : = < variável simples > |
< variável indiciada >

3. 1. 2. *Exemplos.*

epsilon
det A
a 17
Q [7,2]
x [sen (n × pi/2), Q [3, n, 4]]

3. 1. 3. *Semântica.*

Uma variável é uma designação dada a um valor único. Este valor pode ser utilizado em expressões para formar outros valores e pode ser alterado à vontade mediante instruções de afectação (cf. *4.* 2.). O tipo de valor de uma variável particular é definido na declaração relativa a esta própria variável (cf. *5.* 1. Declarações de tipo) ou no identificador de tabela correspondente (cf. *5.* 2. Declarações de tabela).

3. 1. 4. *Índices.*

3. 1. 4. 1. As variáveis indiciadas designam valores que são componentes de tabelas a

várias dimensões (cf. *5. 2.* Declarações de tabela). Cada expressão aritmética duma lista de índices ocupa uma posição de índice de uma variável indiciada e tem o nome de *índice*. A lista completa dos índices está contida nos parêntesis de índices [ ]. A componente da tabela à qual a variável indiciada se refere é determinada pelo verdadeiro valor numérico dos seus índices (cf. *3. 3.* Expressões aritméticas).

3. 1. 4. 2. Todo o índice actua como uma variável do tipo *inteiro* e a atribuição de um valor a este índice equivale a uma afectação a essa variável fictícia (cf. *4. 2. 4.*). O valor da variável indiciada é definido apenas no caso do valor do índice estar compreendido entre os limites dos índices da tabela (cf. *5. 2.* Declarações de tabela).

3. 2. Indicadores de função.

3. 2. 1. Sintaxe.

< identificador de procedimento > : : =
< identificador >
< parâmetro efectivo > : : =
< cadeia > | < expressão > |
< identificador de tabela > |
< identificador de comutação > |
< identificador de procedimento >
< cadeia de letras > : : = < letra > |
< cadeia de letras > < letra >
< limitador de parâmetros > : : = , |
) < cadeia de letras > : (
< lista de parâmetros efectivos > : : =
< parâmetro efectivo > |
< lista de parâmetros efectivos >
< limitador de parâmetros >
< parâmetro efectivo >
< parte de parâmetro efectivo > : : =
< vazio > |
(< lista de parâmetros efectivos >)
< indicador de função > : : =
< indicador de procedimento >
< parte de parâmetro efectivo >

3. 2. 2. Exemplos.

sen (*a* — *b*)
J [*v* + *s*, *n*]
R
S (*s* — 5) *Temperatura*: (*T*) *Pressão*: (*P*)
compor (': = ') *Feixe*: (*Q*)

3. 2. 3. Semântica.

Os indicadores de função definem valores simples, numéricos ou lógicos, que resultam da aplicação dum conjunto determinado de regras dadas por uma declaração de procedimento (cf. *5. 4.* Declarações de procedimento) a determinados conjuntos de parâmetros efectivos. Em *4. 7.* (Instruções de procedimento), são dadas as regras que determinam as especificações dos parâmetros efectivos. Toda a declaração de procedimento não define obrigatòriamente o valor de um indicador de função.

3. 2. 4. Funções padrão.

Podemos reservar certos indicadores para as funções padrão da análise que deverão exprimir-se sob a forma de procedimento. Recomenda-se a lista seguinte:

abs (*E*) para o módulo (valor absoluto) do valor da expressão *E*
sign (*E*) para o sinal do valor de *E* (+1 para $E > 0$, 0 para $E = 0$, −1 para $E < 0$)
sqrt (*E*) para a raiz quadrada do valor de *E*
sin (*E*) para o seno do valor de *E*
cos (*E*) para o coseno do valor de *E*
arctan (*E*) para o valor principal do arco-tangente do valor *E*

ln {*E*} para o logarítmo neperiano do valor de *E*

exp {*E*} para a função exponencial do valor de $E(e^E)$.

Considera-se que estas funções operam indiferentemente sobre argumento do tipo *real* ou *inteiro*. Elas fornecem todos os valores do tipo *real* excepto para *sign* {*E*} que possue valores do tipo *inteiro*. Numa representação particular, estas funções podem ser directamente disponíveis sem declaração explícita (cf. **5.** Declarações).

3. 2. 5. *Funções de transferência.*

Podemos definir funções de transferência entre qualquer par de quantidades ou de expressões. Entre as funções padrão, recomenda-se particularmente: *inteiro* {*E*}, que «transfere» uma expressão do tipo real para outra do tipo inteiro, afectando-a do valor que é o maior inteiro, menor ou igual ao valor de *E*.

3. 3. *Expressões aritméticas.*

3. 3. 1. *Sintaxe.*

< operador aditivo > : : = + | −
< operador multiplicativo > : : = × | / | :
< primário aritmético > : : =
  < número sem sinal > | < variável > |
  < indicador de função > |
  (< expressão aritmética >)
< factor > : : = < primário aritmético > |
  < factor > ↑ < primário aritmético >
< termo > : : = < factor > |
  < termo > < operador multiplicativo >
  < factor >
< expressão aritmética simples > : : =
  < termo > |
  < operador aditivo > < termo > |
  < expressão aritmética simples >
  < operador aditivo > < termo >
< proposição se > : : =
  se < expressão booleana > *então*
< expressão aritmética > : : =
  < expressão aritmética simples > |
  < proposição se >
  < expressão aritmética simples >
  senão < expressão aritmética >

3. 3. 2. *Exemplos.*

Primários aritméticos

$7.394_{10} - 8$
soma
w i + 2 , 8]
cos { y + z × 3}
{a − 3/y + v u ↑ 8}

Factores

omega
soma ↑ cos { y + z × 3}
$7.394_{10} - 8 \uparrow w[i + 2, 8] \uparrow$
{a − 3/y + v u ↑ 8}

Termos

U
omega × soma ↑ cos { y + z × 3} / 7.394$_{10}$ *−*
8 ↑ w [i + 2 , 8] ↑ {a − 3/y + v u ↑ 8}

Expressões aritméticas simples

U − Y u + omega × sum ↑ cos { y + z × 3} /
$7.394_{10} - 8 \uparrow w[i+2,8] \uparrow$ *{a − 3/y +*
+ v u ↑ 8}

Expressões aritméticas

w × u − Q {S + C u} ↑ 2
se q > 0 então S + 3 × Q / A senão
2 × S + 3 × Q
se a < 0 então U + V senão se a × b > 17
então U/V senão se k ≠ y então V/U
senão 0
a × sin {omega × t}

$0\,57_{10}\,12 \times a[N \times (N-)1/2\,,0]$
$\{A \times arctan\,|\,y|\,+\,Z|\,\uparrow\,|7\,+\,Q|$
se q então n − 1 senão n
se a < 0 então A/B senão se b = 0 então B/A senão z

3. 3. 3. *Semântica.*

Uma expressão aritmética é uma regra para o cálculo dum valor numérico. Quando se trata de expressões aritméticas simples, obtém-se este valor executando as operações aritméticas indicadas sobre os valores numéricos efectivos dos primários aritméticos da expressão (como se indica com pormenor em **3**. 3. 4.). No caso de números o valor numérico efectivo dum primário aritmético é evidente.

Para as variáveis trata-se do valor corrente (sentido dinâmico), e para os indicadores de função trata-se do valor proveniente da aplicação das regras de cálculo definidas pelo processo aplicado (cf. **5**. 4 4. Valor dos indicadores de função) aos valores correntes dos parâmetros de procedimento definidos na expressão. Enfim, para as expressões aritméticas entre parêntesis, deve exprimir-se o seu valor pelo método recursivo em função dos valores dos primários aritméticos das três outras categorias.

Nas expressões aritméticas mais gerais, que compreendem proposições se, de entre todas as diversas expressões aritméticas simples apenas uma se escolhe em função dos valores actuais das expressões booleanas (cf. **3**. 4. Expressões booleanas). Esta escolha faz-se, na realidade, da seguinte maneira: as expressões booleanas das proposições se são avaliadas uma a uma, em sequência, da esquerda para a direita, até à obtenção de um valor *verdadeiro*. O valor da expressão aritmética é então o valor da primeira expressão aritmética que segue esta expressão booleana (entende-se por isto, a maior expressão aritmética que se encontra no local considerado).

A construção:

senão < expressão aritmética simples >

é equivalente à construção

senão se verdadeiro então < expressão aritmética simples >

3. 3. 4. *Operadores e Tipos.*

Exceptuando-se as expressões booleanas das proposições se, os constituintes das expressões aritméticas simples devem ser do tipo *real* ou *inteiro* (cf. **5**. 1. Declarações de tipo). O significado dos operadores de base e os tipos das expressões aos quais eles conduzem, obedecem às regras seguintes:

3. 3. 4. 1. As operações $+$, $-$, $\times$, têm o sentido usual de adição, de subtracção ou de multiplicação.

O tipo da expressão será *inteiro*, se os dois operandos são do tipo *inteiro*. Em todos os outros casos será *real*.

3. 3. 4. 2. As operações < termo > / < factor > e < termo > ÷ < factor > representam a divisão como multiplicação do termo pelo inverso do factor segundo as regras de de prioridade definidas em **3**. 3. 5. Assim, por exemplo

$$a/b \times 7/|\,p - q|\,\times v/s$$

significa

$$\{\{\{a \times \{b^{-1}\}\} \times 7\} + \{\{p - q\}^{-1}\}\} \times v\} \times \{s^{-1}\}$$

O operador / é definido para as quatro combinações possíveis dos tipos *real* e *inteiro* e fornece um resultado do tipo *real* em todos os casos. O operador ÷ é definido apenas para os dois operadores do tipo *inteiro* e fornece um resultado do tipo *inteiro*, definido da maneira seguinte:

$a \div b = sign\,\{a/b\} \times inteiro\,\{abs\,\{a/b\}\}$

(cf. **3.** 2. 4. e **3.** 2. 5.).

3. 3. 4. 3. A operação < factor > ↑ < primário aritmético > representa a elevação à potência de que o factor é a base e o primário aritmético o expoente. Assim, por exemplo:

$2 \uparrow n \uparrow k$ significa $(2^n)^k$

enquanto que

$2 \uparrow (n \uparrow m)$ significa $2^{(n^m)}$

Designando por *i* um número do tipo *inteiro*, por *r* um número no tipo *real*, e por *a* um número que é do tipo, ou *inteiro* ou *real*, o resultado obtém-se por meio das regras seguintes:

$a \uparrow i$ se $i > 0$, $a \times a \times \cdots \times a$ (*i* vezes) do mesmo tipo que *a*

se $i = 0$, se $a \neq 0, 1$ do mesmo tipo que *a*

se $a = 0$, indefinido

se $i < 0$, se $a \neq 0, 1/\{a \times a \times \cdots \times a\}$ (o denominador tem $-i$ factores do tipo *real*)

se $a = 0$, indefinido

$a \uparrow r$ se $a > 0$, $exp\,\{r \times ln\,\{a\}\}$ do tipo *real*

se $a = 0$, se $r > 0$, 0 do tipo *real*

$r < 0$: indefinido

se $a < 0$ sempre indefinido

3. 3. 5. *Prioridade dos operadores*

A sequência das operações contidas numa expressão faz-se, em geral, da esquerda para a direita. Convém no entanto ter em consideração as seguintes regras adicionais:

3. 3. 5. 1. São válidas as seguintes regras de prioridade (de acordo com as regras de sintaxe enunciadas em **3.** 3. 1.)

primeiro	↑
segundo	× / ÷
terceiro	+ −

3. 3. 5. 2. As expressões contidas entre parentesis são calculadas em separado e os seus valores são utilizados na sequência dos cálculos.

Consequentemente pode sempre escolher-se a ordem de execução das operações por conveniente disposição dos parêntesis.

3. 3. 6. *Aritmética das quantidades do tipo real*

Atribui-se aos números e às variáveis do tipo *real* o sentido da análise numérica, isto é, as entidades são definidas com uma precisão determinada. De modo análogo, admite-se a possibilidade dum desvio do resultado definido matemàticamente numa expressão aritmética. Não se faz, porém, especificação aritmética exacta e as diferentes representações máquina poderão avaliar as expressões aritméticas de maneiras diferentes. O controlo das consequências decorrentes de tais diferenças deve ser realizado por métodos de análise numérica. Este controlo deve ser considerado como uma parte do processo descrito e deve portanto poder exprimir-se em termos da própria linguagem.

3. 4. *Expressões booleanas*

3. 4. 1. *Sintaxe*

< operador de relação > : : =
< | ≤ | = | ≥ | > | ≠

< relação > : : =
< expressão aritmética simples >
< operador de relação >
< expressão aritmética simples >

< primário booleano > : : =
< valor lógico > | < variável > |
< indicador de função > |
< relação > | (< expressão booleana >)

< secundário booleano > : : =
< primário boeleano > |
¬ < primário booleano >

< factor booleano > : : =
< secundário booleano > |
< factor booleano > ∧
< secundário booleano >

< termo booleano > : : =
< factor booleano > |
< termo booleano >
∨ < factor booleano >

< implicação > : : =
< termo booleano > |
< implicação > ⊃ < termo booleano >

< Booleano simples > : : =
< implicação > |
< Booleano simples > ≡ < implicação >

< expressão booleana > : : =
< Booleano simples > |
< proposição se >
< Booleano simples >
senão < expressão booleana >

3. 4. 2. *Exemplos.*

$x = -2$

$y > V \vee z < q$

$a + b > -5 \wedge z - d > q \uparrow 2$

$p \wedge q \vee x \neq y$

$g \equiv \neg a \wedge b \wedge \neg c \vee d \vee e \supset \neg f$

se $k < l$ *então* $s > w$ *senão* $h < c$

se se se a então b senão c então d
senão f então g senão $h < k$

3. 4. 3. *Semantica.*

Uma expressão booleana é uma regra para o cálculo de um valor lógico. Os princípios de cálculo são inteiramente análogos aos dados pelas expressões aritméticas em **3.** 3. 3.

3. 4. 4. *Tipos.*

As variáveis e os indicadores de função introduzidos como primários booleanos devem ser declarados *booleanos* (cf. **5.** 1. Declarações de tipo e **5.** 4. 4. Valores dos indicadores de função).

3. 4. 5. *Os operadores.*

As relações tomam o valor *verdadeiro* sempre que a relação correspondente seja satisfeita para as expressões nela contidas; caso contrário toma o valor *falso*.

O significado dos operadores lógicos ¬ (não), ∧ (e), ∨ (ou), ⊃ (implica), ≡ (equivalente), é dado pela seguinte tabela de valores (1).

$b1$	*f*	*f*	*v*	*v*
$b2$	*f*	*v*	*f*	*v*
$\neg b1$	*v*	*v*	*f*	*f*
$b1 \wedge b2$	*f*	*f*	*f*	*v*
$b1 \vee b2$	*f*	*v*	*v*	*v*
$b1 \supset b2$	*v*	*v*	*f*	*v*
$b1 \equiv b2$	*v*	*f*	*f*	*v*

3. 4. 6. *Prioridade dos operadores.*

A sequência de operações numa expressão processa-se geralmente da esquerda para a direita, com as seguintes regras adicionais:

3. 4 6. 1. Adoptam-se as seguintes regras de prioridade (de acordo com as regras de sintaxe enunciadas em **3.** 4. 1.).

(1) *v* significa *verdadeiro*
f significa *falso*

primeiro	expressão aritmética segundo 3. 3. 5.
segundo	$< \leqslant = \geqslant > \neq$
terceiro	$\neg$
quarto	$\wedge$
quinto	$\vee$
sexto	$\supset$
sétimo	$\equiv$

3. 4. 6. 2. A utilização dos parentesis deve entender-se no mesmo sentido que em 3. 3. 5. 2.

3. 5. *Expressões de designação.*

3. 5. 1. *Sintaxe.*

< etiqueta > : : =
  < identificador > | < inteiro sem sinal >
< identificador de comutação > : : =
  < identificador >
< indicador de comutação > : : =
  < identificador de comutação >
  [< expressão em índice >]
< expressão de designação simples > : : =
  < etiqueta > |
  < indicador de comutação > |
  (< expressão de designação >)
< expressão de designação > : : =
  < expressão de designação simples > |
  < proposição se >
  < expressão de designação simples >
  senão < expressão de designação >

3. 5. 2. *Exemplos.*

17
p 9
escolher [*n* — *1*]
cidade [*se y* < *0 então N senão N* + *1*]
se A b < *c então 17 senão q* [*se w* ⩽ *0 então 2 senão n*]

3. 5. 3. *Semântica.*

Uma expressão de designação é uma regra para obter a etiqueta duma instrução (cf. 4. Instruções). O princípio de avaliação é inteiramente análogo ao que respeita as expressões aritméticas (cf. 3. 3. 3.). No caso geral, as expressões booleanas das proposições se selecionarão uma expressão de designação simples. Se esta é uma etiqueta o resultado que se pretende é imediatamente determinado. Um indicador de comutação refere-se à correspondente declaração de comutação (cf. 5. 3. Declaração de comutação); de acordo com o valor numérico efectivo da sua expressão em índice, ele seleciona uma das expressões de designação que figuram na declaração de comutação, por meio de uma contagem feita da esquerda para a direita. Como a expressão de designação assim escolhida pode ainda ser um indicador de comutação, esta avaliação é evidentemente um processo recursivo.

3. 5. 4. *A expressão em índice.*

A avaliação da expressão em índice é análoga à das variáveis indiciadas (cf. 3. 1. 4. 2.). O valor de um indicador de comutação é definido apenas no caso de a expressão em índice possuir um dos valores positivos $1, 2, 3, \cdots, n$, em que n é o número de entradas da lista de comutação.

3. 5. 5. *Etiquetas representadas por números sem sinal.*

Estas etiquetas têm a seguinte propriedade: os zeros à esquerda não intervém no seu significado. Por exemplo *00217* representa a mesma etiqueta que *217*.

4. Instruções.

As ordens elementares da linguagem têm o nome de *instruções*. São geralmente executadas em sequência pela ordem por que são escritas. Esta sequência pode, porém, ser interrompida por instruções *ir a* que definam explìcitamente os seus sucessores e por instruções condicionais que possibilitam o salto de certas instruções.

As instruções podem ser providas de etiquetas afim de se poder definir a sequência de sucessão dinâmica.

A definição duma instrução deve ser necessàriamente recursiva para que a sequência de instruções possam grupar-se em instruções compostas e blocos. Mas como as declarações, definidas em 5, figuram de uma maneira fundamental na estrutura sintática, a definição sintática das instruções deve supor as declarações já definidas.

4.1. Instruções compostas e blocos.

4.1.1. Sintaxe.

< instrução não etiquetada de base > : : =
  < instrução de afectação > |
  < instrução *ir a* > | < instrução vazia > |
  < instrução de procedimento >
< instrução de base > : : =
  < instrução não etiquetada de base > |
  < etiqueta > : < instrução de base >
< instrução incondicional > : : =
  < instrução de base > |
  < instrução composta > | < bloco >
< instrução > : : =
  < instrução incondicional > |
  < instrução condicional > |
  < instrução *para* >
< fim de instrução composta > : : =
  < instrução > *fim* | < instrução > ;
  < fim de instrução composta >
< cabeça de bloco > : : = começo
  < declaração > | < cabeça de bloco > ;
  < declaração >
< composta não etiquetada > : : =
  começo < fim de instrução composta >
< bloco não etiquetado > : : =
  < cabeça de bloco > ;
  < fim de instrução composta >
< instrução composta > : : =
  < composta não etiquetada > |
  < etiqueta > : < instrução composta >
< bloco > : : = < bloco não etiquetado > |
  < etiqueta > : < bloco >
< programa > : : = < bloco > |
  < instrução composta >

Esta sintaxe pode exemplificar-se como segue: representemos por I, D e E respectivamente instruções, declarações e etiquetas arbitrárias; as unidades sintáticas de base tomam então as formas

Instrução composta

$$E : E : \cdots \text{ começo } I ; I ; \cdots I ; I \text{ fim}$$

Bloco

$$E : E : \cdots \text{ começo } D ; D ; \cdots D ; I ; I ; \cdots I ; I \text{ fim}$$

Não deve esquecer-se que cada instrução I pode por seu lado ser uma instrução composta ou um bloco.

4.1.2. Exemplos.

Instruções de base

```
a : = p + q
ir a Nápoles
Partida : Continua : W : = 7.993
```

Instrução composta

```
começo x : = 0 ; para y : = 1 passo 1 até n
  fazer x : = x + A[y];
```

```
se x>q então ir a Final senão se x<w−2
então ir a S
Aw: St: W: = x + bob fim
```

Bloco

```
Q: começo inteiro i,k; real w;
para i: = 1 passo 1 até m fazer
para k: = i + 1 passo 1 até m fazer
começo w: = A[i,k]
   A[i,k]: = A[k,i]
   A[k,i]: = w fim para i e k
fim bloco Q
```

4. 1. 3. *Semântica.*

Cada bloco introduz automàticamente um novo nível de nomenclatura. Tal processa-se como segue: todo o identificador que figure no interior do bloco pode, por conveniente declaração (cf. **5.** Declarações) ser classificado de local para o bloco em questão. Isto significa que

(*a*) a quantidade representada por este identificador no interior do bloco não tem existência fora dele e

(*b*) qualquer quantidade representada por este identificador fora do bloco é completamente inacessível ao interior dele.

Os identificadores (excepto os que representam etiquetas) que figuram num bloco e que não são declarados neste bloco, serão não locais para ele, isto é, representarão a mesma quantidade no interior do bloco e ao nível que lhe é imediatamente exterior. Uma etiqueta separada de uma instrução pelo símbolo : isto é, etiquetando esta instrução, comporta-se como se tivesse sido declarada na cabeça do menor bloco que a contém, isto é, o menor bloco cujos parêntesis começo e *fim* confinam a instrução. Neste sentido, um corpo de procedimento deve ser considerado como se estivesse confinado pelos símbolos começo e *fim* e tratado como um bloco.

Como uma instrução dum bloco pode de novo e por sua vez ser um bloco, os conceitos «local» e «não-local» para um bloco devem ser compreendidos recursivamente. Assim, um identificador que seja não-local para um bloco A, pode ser ou não ser não-local para o bloco B no qual A é uma instrução.

4. 2. *Instruções de afectação.*

4. 2. 1. *Sintaxe.*

```
< parte esquerda > : : = < variável > : = |
   < identificador de procedimento > : =
< lista de partes esquerdas > : : =
   < parte esquerda > |
   < lista de partes esquerdas >
   < parte esquerda >
< instrução de afectação > : : =
   < lista de partes esquerdas >
   < expressão aritmética > |
   < lista de partes esquerdas >
   < expressão booleana >
```

4. 2. 2. *Exemplos.*

```
s: = q[0]: = n: = n + 1 + s
n: = n + 1
A: = B/C − v − q × S
S[v,k + 2]: = 3 − arctan |s × zeta|
V: = Q > Y ∧ Z
```

4. 2. 3. *Semântica.*

As instruções de afectação servem para dar o valor duma expressão a uma ou mais variáveis ou identificadores de procedimento. No caso geral, a instrução será interpretada em três fases como segue:

4. 2. 3. 1. Todas as expressões em índice que figuram nas variáveis da parte esquerda, são avaliadas sucessivamente da esquerda para a direita.

4. 2. 3. 2. A expressão à direita da instrução é calculada.

4. 2. 3. 3. O valor da expressão é atribuído a todas as variáveis da parte esquerda, e os índices tem os valores calculados em *4. 2. 3. 1.*

4. 2. 4. Tipos.

Todas as variáveis e identificadores de procedimento duma lista de partes esquerdas devem ser declarados do mesmo tipo. Se as variáveis são do tipo *Booleano*, a expressão deve ser ela própria booleana. Se as variáveis são do tipo *real* ou *inteiro*, a expressão deve ser aritmética. Se o tipo da expressão aritmética difere do das variáveis e identificadores de procedimento, a mudança de tipo realiza-se automàticamente. Para a transferência do tipo *real* para o tipo *inteiro*, a função de transferência deve conduzir a um resultado equivalente a

$$inteiro\ \{E + 0.5\}$$

em que E é o valor da expressão. O tipo associado a um identificador de procedimento é dado pelo declarador que aparece como primeiro símbolo da declaração de procedimento correspondente.

4. 3. Instruções Ir a.

4. 3. 1. Sintaxe.

< instrução *ir a* > : : =
ir a < expressão de designação >

4. 3. 2. Exemplos.

ir a 8
ir a saída [*n* + 1]
ir a cidade [*se y*<0 *então N senão N*+1]
ir a se A b<*c então 17 senão q* [*se w*<0 *então 2 senão n*]

4. 3. 3. Semântica.

Uma instrução *ir a* interrompe a sequência normal das operações, definida pela ordem de escrita das instruções, definindo o seu sucessor explìcitamente por meio do valor duma expressão de designação. A instrução seguinte a executar será a que tem por etiqueta o valor desta expressão.

4. 3. 4. Restrição.

Como as etiquetas são inerentemente locais, uma instrução *ir a* não pode conduzir do exterior ao interior dum bloco. Uma instrução *ir a* pode, porém, conduzir do exterior ao interior de uma instrução composta.

4. 3. 5. ir a indicador de comutação não definido.

Uma instrução *ir a* é equivalente a uma instrução vazia se a expressão de designação é indicador de comutação cujo valor não está definido.

4. 4. Instruções vazias.

4. 4. 1. Sintaxe.

< instrução vazia > : : = < vazio >

4. 4. 2. Exemplos.

L : começo ⋯ ; Jose : fim

4. 4. 3. Semântica.

Uma instrução vazia não executa nada. Pode servir para colocar uma etiqueta.

4. 5. Instruções condicionais.

4. 5. 1. Sintaxe

< proposição se > : : =
se < expressão booleana > *então*

< instrução incondicional > : : =
< instrução de base > |
< instrução composta > | < bloco >
< instrução se > : : = < proposição se >
< instrução incondicional >
< instrução condicional > : : =
< instrução se > | < instrução se >
senão < instrução > |
< proposição se > < instrução *para* > |
< etiqueta > :
< instrução condicional >

4. 5. 2. Exemplos.

Se $x > o$ *então* $n := n + 1$
se $v > u$ *então* V: $q := n + m$ *senão ir a* R
se $s < O < P \leqslant Q$ *então* AA : *começo se*
$q < v$ *então* $a := v/s$
senão $y := 2 \times a$ *e senão se* $v > x$
então $a := v - q$
senão se $v > s - 1$ *então ir a* S

4. 5. 3. Semântica.

As instruções condicionais provocam a execução ou o salto de certas instruções de acordo com os valores das expressões booleanas especificadas.

4. 5. 3. 1. Instrução se.

A instrução incondicional de uma instrução se será executada se a expressão booleana da proposição se é verdadeira. Caso contrário, ela será saltada, passando-se à instrução seguinte.

4. 5. 3. 2. Instrução condicional.

Quanto à sintaxe, há duas formas possíveis de instruções condicionais, que podem ser representadas como segue:

Se B1 *então* S1 *senão se* B2 *então* S2
senão S3 ; S4

e

Se B1 *então* S1 *senão se* B2 *então* S2
senão se B3 *então* S3 ; S4

Aqui B1 a B3 são expressões booleanas, S1 a S3 instruções incondicionais. S4 é a instrução que segue a instrução condicional completa.

A execução duma instrução condicional pode descrever-se como segue: as expressões booleanas das proposições se são avaliadas uma a seguir à outra em sequência da esquerda para a direita até a obtenção de uma, cujo valor é *verdadeiro*. Então a instrução incondicional que segue esta expressão booleana é executada. A menos que esta instrução defina o seu sucessor explìcitamente, a próxima instrução a ser executada será S4, quere dizer a instrução que segue a instrução condicional. Assim, pode dizer-se que o efeito do limitador *senão* é o de dar como sucessor à instrução que segue, a instrução que segue a instrução condicional inteira.

A construção

senão < instrução incondicional >

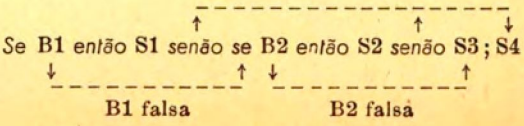
pode ser substituída por

senão se verdadeiro então
< instrução incondicional >

que lhe é equivalente.

Se nenhuma das expressões booleanas das proposições se é verdadeira o efeito da instrução condicional completa será o de uma instrução vazia.

Para maior clareza, pode ser útil o esquema seguinte:



4. 5. 4. *Ir a* instrução condicional.

O efeito de uma inscrição *ir a* que conduza ao interior de uma instrução condicional, resulta directamente da explicação anterior do efeito de *senão*.

4. 6. *Instrução para.*

4. 6. 1. *Sintaxe.*

< elemento duma lista de *para* > : : =
< expressão aritmética > |
< expressão aritmética > *passo*
< expressão aritmética > *até*
< expressão aritmética > |
< expressão aritmética > *enquanto que*
< expressão booleana >
< lista de *para* > : : =
< elemento duma lista de *para* > |
< lista de *para* >,
< elemento de uma lista de *para* >
< proposição para > : : = *para*
< variável > : =
< lista de *para* > *fazer*
< instrução *para* > : : =
< proposição para > < instrução > |
< etiqueta > : < instrução *para* >

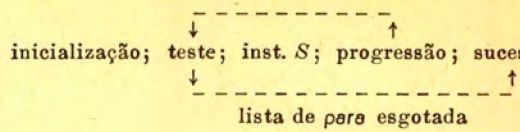
4. 6. 2. *Exemplos.*

para $q := 1$ *passo* s *até* n *fazer* $A[q] := B[q]$
para $k := 1, V1 \times 2$ *enquanto que* $V1 < N$
fazer para $J := I + G, L, 1$ *passo* 1 *até* $N, C + D$ *fazer* $A[k, j] := B[k, j]$

4. 6. 3. *Semântica.*

Uma proposição para provoca a execução repetida de zero ou várias vezes da instrução S que lhe sucede. Além disso, dá sequência de valores à variável que controla.

O mecanismo pode ser visualisado pelo esquema seguinte:



Neste esquema, a palavra inicialização significa cálculo do primeiro valor da proposição para; progressão significa: cálculo do valor seguinte da proposição para. O teste determina se o valor final foi calculado. Se sim, passa-se ao sucessor da instrução *para*. Se não, é executada a instrução que segue a proposição *para*.

4. 6. 4. *Os elementos duma lista de para.*

A lista dá uma regra para obter os valores dados sucessivamente à variável controlada. Esta sucessão de valores obtém-se a partir de elementos da lista tomando estes um por um pela ordem por que são escritos. A sequência dos valores gerados pelas três espécies de elementos duma lista de *para* e a execução correspondente da instrução são descritos pelas regras seguintes:

4. 6. 4. 1. *Expressão aritmética.*

Este elemento provoca o cálculo dum valor: o valor da expressão aritmética calculada imediatamente antes da execução correspondente da instrução S.

4. 6. 4. 2. *Elemento com passo e até.*

O efeito de um elemento da forma A *passo* B *até* C em que A, B e C são expressões aritméticas, pode ser descrito de maneira mais concisa em termos de instruções ALGOL como segue:

$V := A$;
$L1$: *se* $\{V - C\} \times sign\ \{B\} < 0$ *então ir a elemento esgotado*;

Instrução S;

V := *V* + *B*;
ir a L1;
elemento esgotado: ···

em que *V* é a variável controlada da proposição para, e «elemento esgotado» designa ou o cálculo de *V* de acordo com o elemento que segue na lista de *para*, ou, a expressão que segue no programa, se o elemento considerado é o último.

4. 6. 4. 3. Elemento com *enquanto que.*

O efeito de um elemento da forma *E enquanto que F*, em que *E* é uma expressão aritmética e *F* uma expressão booleana, pode ser assim descrito em termos de instrução ALGOL.

L3: *V* := *E* e *p*;
se ¬ *F* então *ir a elemento esgotado.*

Instrução S

ir a L3; *elemento esgotado*: ···

com as mesmas notações que em *4. 6. 4. 2.*

4. 6. 5. Valor da variável controlada à saída.

Na saída da instrução *S* (suposta composta) com uma instrução *ir a*, o valor da variável controlada será aquele imediatament anterior ao da execução da instrução *ir a*.

Se a saída se deve ao esgotamento da lista de *para*, o valor da variável controlada não é definido depois da saída.

4. 6. 6. Ir a conducente a uma instrução para.

O efeito de uma instrução *ir a*, exterior a uma instrução *para*, que reenvia a uma etiqueta interior à instrução *para*, não é definido.

4. 7. Instruções de procedimento.

4. 7. 1. Sintaxe.

< parâmetro efectivo > : : = < cadeia > |
< expressão > |
< identificador de tabela > |
< identificador de comutação > |
< identificador de procedimento >
< cadeia de letras > : : = < letra > |
< cadeia de letras > < letra >
< limitador de parâmetro > : : = , |
) | < cadeia de letras > : (
< lista de parâmetros efectivos > : : =
< parâmetro efectivo > |
< lista de parâmetros efectivos >
< limitador de parâmetros >
< parâmetro efectivo >
< parte-parâmetro efectivo > : : =
< vazio > |
(| < lista de parâmetros efectivos >) |
< instrução de procedimento > : : =
< identificador de procedimento >
< parte-parâmetro efectivo >

4. 7. 2. Exemplos.

Norma (*A*) *Ordem*: (*7*) *Resultado em*: (*V*)
Transposição (*W*, *v* + *1*)
Maxabs (*A*, *N*, *M*, Y_y, *I*, *K*)
Produto escalar (*A*[*t*, *P*, *u*], *B*[*P*], *10*, *P*, *Y*)

Estes exemplos correspondem aos exemplos dados em *5. 4. 2.*

4. 7. 3. Semântica.

Uma instrução de procedimento tem por objectivo provocar a execução do corpo dum procedimento (cf. *5. 4.* Declarações de procedimento). Se o corpo do procedimento é uma instrução escrita em ALGOL, o efeito desta execução será o de efectuar as seguintes operações sobre o programa:

4. 7. 3. 1. Afectação de valor (chamada por valor).

Todos os parâmetros formais presentes na parte valor da cabeça da declara-

ção de procedimento, recebem os valores (cf. 2. 8. Valores e tipos) dos parâmetros efectivos correspondentes, passando-se tudo como se estas afectações de valor fossem efectuadas explìcitamente antes de entrar no corpo do procedimento. O efeito produzido é aquele que se obteria com a criação de um bloco suplementar que englobasse o corpo de procedimento e no qual estas afectações seriam feitas a variáveis locais para este bloco fictício, e de tipo em acordo com as especificações (cf. 5. 4. 5.). Como consequência, variáveis chamadas por valor devem ser consideradas como não locais para o corpo de procedimento, mas locais para o bloco fictício (cf. 5. 4. 3.).

4. 7. 3. 2. *Substituição do nome (chamada por nome).*

Todo o parâmetro formal não presente na parte valor é substituído, em todo o corpo do procedimento, pelo parâmetro efectivo correspondente, depois colocado entre parêntesis deste último sempre onde tal seja sintàticamente possível. Eventuais conflitos entre os identificadores introduzidos por esta substituição e outros identificadores já presentes no corpo de procedimento serão evitados sistemàticamente por modificação conveniente dos identificadores formais ou locais contidos no corpo.

4. 7. 3. 3. *Substituição pelo corpo e execução.*

Enfim o corpo de procedimento, assim modificado, é posto à disposição da instrução de procedimento e executado. Se o procedimento é empregado em local que se encontre fora do campo duma quantidade não local do corpo de procedimento os conflitos entre os identificadores introduzidos pelo processo de substituição do corpo e os identificadores cujas declarações são válidas no local onde se encontra a instrução procedimento ou o indicador de função, serão evitados por uma alteração sistemática conveniente destes últimos identificadores.

4. 7. 4. *Correspondência parâmetro efectivo — parâmetro formal.*

A correspondência entre os parâmetros efectivos da instrução de procedimento e os parâmetros formais da cabeça do procedimento, faz-se como segue: a lista dos parâmetros efectivos da instrução de procedimento deve ter o mesmo número de elementos que a lista dos parâmetros formais da cabeça da declaração de procedimento. Obtém-se a correspondência tomando os elementos das duas listas na mesma ordem.

4. 7. 5. *Restrições.*

Para que uma instrução de procedimento tenha um sentido, é necessário evidentemente que as operações sobre o corpo de procedimento definidos em 4. 7. 3. 1. e 4. 7. 3. 2. conduzam a uma instrução ALGOL correcta. Isto impõe como restrição que o género e o tipo de cada parâmetro efectivo sejam compatíveis com o género e o tipo do parâmetro formal correspondente. Eis alguns casos particulares importantes desta regra:

4. 7. 5. 1. Se uma cadeia é introduzida como parâmetro efectivo numa instrução de procedimento ou um indicador de função cujo corpo de procedimento seja uma instrução de ALGOL (por oposição a um código não ALGOL, cf. 4. 7. 8.), então esta cadeia não pode ser utilizada no corpo de procedimento a não ser como parâmetro efectivo duma outra chamada de procedimento. Finalmente, ela não pode ser utilizada senão em corpo de procedimento escrito num código não ALGOL.

4.7.5.2. Um parâmetro formal que apareça como variável na parte esquerda duma instrução de afectação e que não é chamado por valor, apenas pode corresponder a um parâmetro efectivo que seja uma variável (caso particular de expressão).

4.7.5.3. Um parâmetro formal, utilizado no corpo do procedimento como identificador de tabela, apenas pode corresponder a um parâmetro efectivo que seja identificador duma tabela da mesma dimensão. Além disso, se o parâmetro formal é chamado por valor, a tabela local criada pelo chamamento terá os mesmos terminais de índice que a tabela efectiva.

4.7.5.4. Um parâmetro formal chamado por valor não pode em geral corresponder a um identificador de comutação ou de procedimento, porque estes últimos não têm valor (a excepção é o identificador de procedimento duma declaração de procedimento que tem uma parte — parâmetros formais vasia (cf. 5.4.1.) e que define o valor dum indicador de função (cf. 5.4.4.). Este identificador de procedimento é em si mesmo uma expressão completa).

4.7.5.5. Todo o parâmetro formal pode comportar restrições sobre o tipo do parâmetro efectivo que lhe é associado (estas restrições podem ser ou não dadas pelas especificações da cabeça de procedimento). Na instrução procedimento, tais restrições devem evidentemente ser observadas.

4.7.6. Suprimido na nova redacção.

4.7.7. Limitador de parâmetros.

Todos os limitadores de parâmetros são equivalentes. Nenhuma correspondência entre os limitadores utilizados numa instrução de procedimento e os utilizados na cabeça do procedimento, é examinada; apenas o seu número deve ser o mesmo. Assim, é facultativa a informação contida na forma elaborada.

4.7.8. Corpo de procedimento expresso em código.

As restrições impostas sobre uma instrução de procedimento, chamando um procedimento que tenha o seu corpo escrito num código diferente de ALGOL, dependem evidentemente das características do código utilizado e da intenção do utilizador e cai assim fóra do domínio da linguagem de referência.

5. Declarações.

As declarações servem para definir certas propriedades das quantidades utilizadas no programa. Uma declaração para um identificador é válida para um bloco. No exterior deste bloco o referido identificador pode ser utilizado para outros fins (cf. *4.1.3.*).

Dinâmicamente isto implica as regras seguintes: no momento de entrada num bloco (por meio de *começo*, visto que as etiquetas no interior são locais e portanto inacessíveis do exterior), todos os identificadores declarados para o bloco tomam o significado implícito na natureza das declarações dadas. Se estes identificadores já foram definidos por outras declarações do exterior, a partir deste momento tomam novo significado. Por outro lado, os identificadores que não são declarados para um bloco mantém o seu antigo significado.

No momento da saída dum bloco (por meio de *fim* ou por uma instrução *ir a*) todos os identificadores que foram declarados para o bloco perdem de novo o seu significado.

Uma declaração pode ser marcada com o declarador adicional *próprio*. Isto tem o efeito

seguinte: a partir do momento em que se entra de novo no bloco, os valores das quantidades remanescentes (declaradas com *próprio*) permanecerão inalteradas em relação aos que elas tinham à última saída, ao passo que os valores das variáveis declaradas, que não são indicadas remanescentes, ficam indefinidas.

Postos de parte as etiquetas e os parâmetros formais das declarações de procedimento e levando em conta a possível excepção dos identificadores de funções padrão (cf. **3**. 2. 4. e **3**. 2. 5.), todos os identificadores dum programa devem ser declarados. Nenhum identificador pode ser declarado mais que uma vez, em não importa que cabeça de bloco.

Sintaxe.

< declaração > : : =
< declaração de tipo > |
< declaração de tabela > |
< declaração de comutação > |
| < declaração de procedimento >

5. 1. *Declarações de tipo.*

5. 1. 1. *Sintaxe.*

< lista de tipo > : : =
< variável simples > |
< variável simples > ,
< lista de tipo >
< tipo > : : = *real* | *inteiro* | *Booleano*
< declaração de tipo > : : =
< tipo local ou remanescente >
< lista de tipo >
< tipo local ou remanescente > : : =
< tipo > | *próprio* < tipo >

5. 1. 2. *Exemplos.*

Inteiro p, q, s
próprio Booleano Acryl, n

5. 1. 3. *Semântica.*

As declarações de tipo servem para declarar que certos identificadores representam variáveis simples dum dado tipo. As variáveis declaradas reais podem tomar apenas valores positivos e negativos, inclusivé o zero. As variáveis declaradas inteiras podem tomar apenas valores inteiros positivos ou negativos, inclusivé o zero. As variáveis declaradas booleanas podem tomar apenas os valores *verdadeiro* ou *falso*.

Nas expressões aritméticas, toda a posição que possa ser ocupada por uma variável declarada real, pode ser ocupada por uma variável declarada inteira. Para a semântica de *própria* ver o quarto parágrafo de **5**. Declarações.

5. 2. *Declaração de tabela.*

5. 2. 1. *Sintaxe.*

< terminal inferior > : : =
< expressão aritmética >
< terminal superior > : : =
< expressão aritmética >
< par de terminais > : : =
< terminal inferior > :
< terminal superior >
< lista de pares de terminais > : : =
< pares de terminais > |
< lista de pares de terminais > ,
< pares de terminais >
< secção de tabela > : : =
< identificador de tabela >
[< lista de pares de terminais >]
< identificador de tabela > ,
< secção de tabela >
< lista de tabela > : : =
< secção de tabela > |
< lista de tabela > ,
< secção de tabela >
< declaração de tabela > : : =
tabela < lista de tabela > |
< tipo local ou remanescente >
tabela < lista de tabela >

5. 2. 2. Exemplos.

Tabela a, b, c [7 : n, 2 : m], s[— 2 : 10]
próprio inteiro tabela A[se c<0 então 2 senão 1 : 20]
tabela real q[— 7 : — 1].

5. 2. 3. Semântica.

Uma declaração de tabela tem por função declarar que um ou vários identificadores representam tabelas de variáveis indiciadas e dão as dimensões das tabelas, os terminais dos índices e o tipo das variaveis.

5. 2. 3. 1. Terminais de índices.

Os terminais de índices para as tabelas são dados nos primeiros parêntesis de índices segundo o identificador desta tabela, sob a forma duma lista de pares de terminais.

Cada entidade desta lista dá o terminal superior e o terminal inferior dum índice, sob a forma de duas expressões aritméticas separadas pelo limitador : . A lista de pares de terminais, dá os terminais de todos os índices tomados em sequência da esquerda para a direita.

5. 2. 3. 2. Dimensões.

As dimensões são dadas pelo número de elementos nas listas de pares de terminais.

5. 2. 3. 3. Tipos.

Todas as tabelas declaradas numa declaração devem ser do mesmo tipo. Se nenhum declarador de tipo é dado adopta-se o tipo real.

5. 2. 4. Terminais superiores e inferiores.

5. 2. 4. 1. As expressões correspondentes serão analisadas da mesma maneira que as expressões em índice (cf. *3. 1. 4. 2*).

5. 2. 4. 2. As expressões apenas podem depender das variáveis e dos procedimentos que são não locais para o bloco no qual a declaração de tabela é válida. Em consequência, no bloco mais exterior dum programa apenas se podem encontrar declarações de tabela com terminais constantes.

5. 2. 4. 3. Uma tabela é definida apenas quando os valores de todos os terminais superiores de índices não são inferiores aos dos terminais inferiores correspondentes.

5. 2. 4. 4. As expressões serão avaliadas por uma vez em cada entrada no bloco.

5. 2. 5. Identidade de variáveis indiciadas.

A identidade duma variável indiciada não tem relação alguma com os terminais de índices dados na declaração de tabela. Mas, mesmo que uma tabela seja declarada remanescente, os valores das variáveis indiciadas correspondentes serão, de cada vez, definidos apenas pelos daquelas variáveis que têm índices no interior dos terminais de índices calculados por último.

5. 3. Declaração de comutação.

5. 3. 1. Sintaxe.

< lista de comutação > : : =
< expressão de designação > |
< lista de comutação > ,
< expressão de designação >
< declaração de comutação > : : =
comutador
< identificador de comutação > . =
< lista de comutação >

5. 3. 2. Exemplos.

comutador S : = S1, S2, Q[m], se v > — 5 então S3 senão S4
comutador Q : = p1, w

5. 3. 3. Semântica.

Uma declaração de comutação define o conjunto de valores correspondentes a um indicador de comutação.

Estes valores são dados um a um como os valores das expressões de designação que compõem a lista de comutação. A cada uma destas expressões de designação está associado um inteiro positivo $1, 2, \cdots$ etc., obtido por contagem dos elementos da lista da esquerda para a direita. O valor do indicador de comutação correspondente a um valor dado da expressão em índice (cf. *3. 5.* Expressão de designação) é o valor da expressão de designação na lista de comutação que tem este valor dado como inteiro associado.

5. 3. 4. Avaliação das expressões na lista de comutação.

Uma expressão na lista de comutação será avaliada cada vez que se faz referência ao elemento da lista na qual a expressão aparece utilizando os valores correntes de todas as variáveis.

5. 3. 5. Influência dos campos.

Se um indicador de comutação aparece fora do campo duma quantidade que intervem numa expressão de designação da lista de comutação, e se uma avaliação do indicador de comutação escolhe esta expressão de designação, então os conflitos entre os identificadores das quantidades desta expressão e os identificadores cujas declarações são válidas no local do indicador de comutação serão ultrapassados por uma alteração sistemática conveniente destes últimos identificadores.

5. 4. Declarações de procedimento.

5. 4. 1. Sintaxe.

< parâmetro formal > : : =
  < identificador >

lista de parâmetros formais > : : =
  < parâmetro formal > |
  < lista de parâmetros formais >
  < limitador de parâmetro >
  < parâmetro formal >
< parte-parâmetro formal > : : =
  < vazio > |
  | < lista de parâmetros formais > |
< lista de identificadores > : : =
  < identificador > |
  < lista de identificadores > ,
  < identificador >
< parte-valor > : : = *valor*
  < lista de identificadores > ; | < vazio >
< especificador > : : = *cadeia* | < tipo > |
  tabela | < tipo > *tabela etiqueta* |
  comutador | *procedimento* | < tipo >
  procedimento
< parte-especificação > : : = < vazio > |
  < especificador >
  < lista de identificadores > ; |
  < parte-especificação >
  < especificador >
  < lista de identificadores > ;
< cabeça de procedimento > : : =
  < identificador de procedimento >
  < parte de parâmetro formal > ;
  < parte-valor > < parte-especificação >
< corpo de procedimento > : : =
  < instrução > | < código >
< declaração de procedimento > : : =
  procedimento
  < cabeça de procedimento >
  < corpo de procedimento > |
  < tipo > *procedimento*
  < cabeça de procedimento >
  < corpo de procedimento >

5. 4. 2. Exemplos (ver também os exemplos no final deste relatório).

```
procedimento Traço {a} Ordem : {n}
   Resultado : {s} ; valor n ;
tabela a ; inteiro n ; real s ;
começo inteiro k ;
```

```
s := 0;
para k : 1 passo 1 até n fazer s := s + a[k,k]
fim
procedimento transpor {a} ordem : {n} ;
  valor n ; tabela a ; inteiro n ;
começo real w ; inteiro i , k ;
para i := 1 passo 1 até n fazer
  para k := 1 + i passo 1 até n fazer
  começo w := a [i,k] ;
    a [i,k] := a [k,i] ;
    a [k,i] := w
  fim
fim transpor

procedimento inteiro passo {u} ; real u ;
  Passo := se 0≤u∧u≤1 então 1 senão 0

procedimento absmax {a} dimensão :
  {n,m} resultado : {y} índices : {i,k} ;
comentário: O maior elemento em valor
  absoluto da matriz a de dimensão n sobre
  m é transferido em y e os índices deste
  elemento em i e k;
tabela a; inteiro n,m,i,k; real y;
começo inteiro p,q;
  y:=0;
para p:=1 passo 1 até n fazer para
  q:=1 passo 1 até m fazer
se abs{a[p,q]} > y então começo
  y:=abs{a[p,q]};
  i:=p; k:=q fim fim Absmax

procedimento Produto escalar {a,b}
  Ordem: {k,p} Resultado: {y};
  valor k; inteiro k,p; real y,a,b;
  começo real s; s:=0;
    para p:=1 passo 1 até k fazer
    s:=s+a×b;
    y:=s
fim produto escalar
```

5. 4. 3. *Semântica.*

Uma declaração de procedimento serve para definir o procedimento associado a um identificador de procedimento. O constituinte principal de uma declaração de procedimento é uma instrução ou um conjunto de códigos, o corpo do procedimento, que pelo emprego de instruções de procedimento e/ou de indicadores de função, pode ser comandado depois de outras partes dum bloco na cabeça do qual aparece a declaração de procedimento. Associado ao corpo, encontra-se uma cabeça que precisa certos identificadores que figuram no corpo na situação de parâmetros formais. Quando o procedimento é activado (cf. *3. 2.* Indicadores de função e *4. 7.* Instruções de procedimento), os parâmetros formais do corpo de procedimento tomarão os valores dos parâmetros efectivos ou então serão substituídos por estes últimos. Os identificadores do corpo de procedimento que não são formais serão, quer locais quer não-locais para o corpo, segundo tenham sido ou não declarados neste corpo. Entre estes identificadores, os que não são locais para o corpo podem ser locais para o bloco, na cabeça do qual aparece a declaração de procedimento. O corpo de procedimento age sempre como um bloco quer tenha ou não a forma respectiva. Em consequência, o campo de toda a etiqueta actuando uma instrução no corpo ou o próprio corpo em si nunca pode estender-se ao exterior do corpo de procedimento. Além disso, se o identificador dum parâmetro formal é declarado no interior do corpo de procedimento (compreendendo o caso da sua utilização na situação de etiqueta como em *4. 1. 3.*) dá-se-lhe um significado local e os parâmetros efectivos correspondentes são inacessíveis no campo desta quantidade local interior.

5. 4. 4. *Valores dos indicadores de função.*

Para que uma declaração de procedimento defina o valor de um indicador de função, é necessário que no corpo de procedimento apareçam uma ou mais afectações de valor

para o identificador de procedimento. Pelo menos uma destas afectações deve ser efectuada. O tipo associado ao identificador de procedimento deve ser declarado por meio dum declarador de tipo que será o primeiro símbolo da declaração de procedimento. O último valor assim afectado é utilizado para continuar a avaliação da expressão na qual aparece o indicador de função. Toda outra ocorrência do identificador de procedimento no corpo de procedimento em local diferente de uma parte esquerda duma instrução de afectação significa a chamada do procedimento.

5. 4. 5. *Especificações.*

Pode-se incluir na cabeça uma parte especificação que forneça informações sobre os géneros e tipos dos parâmetros formais por meio de notações evidentes. Nesta parte nenhum parâmetro formal pode aparecer mais que uma vez. As especificações de parâmetros formais chamados por valor devem ser dadas enquanto que as especificações dos parâmetros formais chamados por nome podem ser omissas.

5. 4. 6. *Corpo de procedimento em código.*

É óbvio que o corpo de procedimento pode exprimir-se numa linguagem diferente do ALGOL. Tal emprego releva uma representação máquina e nenhuma indicação mais precisa relativa a este código pode ser dada na linguagem de referência.

Exemplos de declarações de procedimento.

Exemplo 1.

```
procedure Euler (fct, sum, eps, tim); value
eps, tim; integer tim;
real procedure fct; real sum, eps;
comment: Euler calcula a soma de fct(i) desde
i=0 a i=∞ por meio de uma conveniente
transformação de Euler. A adição termina
desde que se obtenha para valor absoluto
dos termos da série transformada, tim vezes
consecutivas um valor menor que eps. Forne-
ce-se portanto uma função fct com 1 só argu-
mento inteiro, um terminal superior eps e um
inteiro tim. A saída é a som a som. Euler é
particularmente eficaz no caso de séries alter-
nadas lentamente convergentes ou divergentes;
begin integer i,k,n,t; array m[0:15]; real
mn, mp, ds;
i:=n:=t:=0; m[0]:=fct(0); sum:='m[0]/2;
termo seguinte: i:=i+1; mn:=fct(i);
  for k:=0 step 1 until n do
    begin mp:=(mn+m[k])/2; m[k]:=mn;
    mn:=mp end;
  if (abs(mn)<abs(m[n]))∧(n<15) then
    begin ds:=mn/2; n:=n+1, m[n]:=mn
    end aceita
  else ds:=mn;
  som:=som+ds;
  if abs(ds)<eps then t:=t+1 else t:=0;
  if t<tim then go to termo seguinte
end Euler
```

Exemplo 2.

```
procedure RK(x,y,n,FKT,eps,eta,xE,yE,fi);
    value x,y; integer n;
Boolean fi; real x,eps,eta,xE; array y,yE;
    procedure FKT;
comment: RK integra o sistema
y'k=fk(x,y1,y2,···yn) (k=1,2,···,n) de
equações diferenciais pelo método de Runge-
Kutta com uma escolha automática do valor
mais conveniente do passo da integração. Os
parâmetros são: os valores iniciais de x e
y[k] para x e as funções incógnitas yk(x).
A ordem n do sistema. O procedimento
FKT(x,y,n,z) que representa o sistema a
integrar, isto é o conjunto das funções fk.
```

```
Os erros admissíveis eps e eta que coordenam
a precisão das integrações numéricas. O fim
do intervalo de integração xE. O parâmetro
de saída yE que representa a solução para
x=xE. A variável booleana fi que deve
tomar sempre o valor true para uma entrada
isolada ou para uma primeira entrada em RK.
Mas, se as funções y devem ser determinadas
em diferentes pontos x0, x1, x2, então o pro-
cedimento pode ser chamado sucessivamente
{com x=xk, xE=xk+1, para k=0,1,···,n—1}
e então as chamadas ulteriores poderão apa-
recer com fi=false para ganhar tempo de
cálculo. Os parâmetros FKT devem ser x, y, n
o parâmetro de saída z representa o conjunto
das derivadas z[k]=fk{x,y[1],y[2],···,y[n]}
para x e o valor efectivo de y. Um procedi-
mento comp intervem como identificador não
local;
begin
  array z,y1,y2,y3[1:n]; real x1,x2,x3,H;
    Boole an out;
  integer k,j; own real s,Hs;
  procedure RK1ST{x,y,h,xe,ye}; real
    x,h,xe; array y,ye;
    comment: RK1ST integra um passo iso-
    lado de Runga-Kutta com va'ores iniciais
    x,y[k] que conduzam aos parâmetros de
    saída xe:=x+h e ye[k], sendo o último
    a solução em xe.
    Importante: os parâmetros n, FKT, z
    entram em RK1ST como entidades não
    locais;
    begin
      array w[1;n], a[1:5]; integer k,j;
      a[1]:=a[2]:=a[5]:=h/2;
      a[3]:=a[4]:=h;
      xe:=x;
      for k:=1 step 1 until n do ye[k]:=
      w[k]:=y[k];
      for j:=1 step 1 until 4 do
      begin
        FKT{xe,w,n,z};
        xe:=x+a[j];
        for k:=1 step 1 until n do
        begin
          w[k]:=y[k]+a[j]×z[k];
          ye[k]:=ye[k]+a[j+1]×z[k]/3
        end k
      end j
    end RK1ST;

Começo do programa.

    if fi then begin H:=xE—x; s:=0
    end else H:=Hs;
    out:=false;
AA: if {x+2.01×H—xE>0}={H>0} then
    begin Hs:=H; out:=true; H:=
    {xE—x}/2 end if;
    RK1ST{x,y,2×H,x1,y1};
BB: RK1ST{x,y,x2,y2};
    RK1ST{x2,y2,H,x3,y3};
    for k:=1 step 1 until n do
      if comp{y1[k],y3[k],eta}>eps then
      go to CC;
    comment: comp{a,b,c} é um indicador
    de função cujo valor é o valor absoluto
    da diferença das mantissas de a e b uma
    vez que os expoentes destas quantidades
    tenham sido tomados iguais ao maior dos
    expoentes dos parâmetros iniciais dados
    a,b,c;
    x:=x3; if out then go to DD;
    for k:=1 step 1 until n do y[k]:=y3[k];
    if s=5 then begin s:=0; H:=2×H
    end if;
    s:=s+1; go to AA;
CC: H:=0,5×H; out:=false; x1:=x2;
    for k:=1 step 1 until n do y1[k]:=y2[k];
    go to BB;
DD: for k:=1 step 1 until n do yE[k]:=yE[k]
    end RK
```

Índice alfabético das definições dos conceitos e das unidades sintáticas

Todas as referências são dadas por número de secção. As referências são divididas em três grupos:

def A seguir à abreviatura def indica-se a referência da definição sintática (se ela existe).

sint A seguir à abreviatura sint indicam-se as reefrências das ocorrências em fórmulas metalinguísticas. As referências já mencionadas no grupo def não são retomadas.

text A seguir à abreviatura text indicam-se as referências das definições dadas no texto.

Os símbolos de base representados por sinais diferentes das letras em futura fina são reunidos no começo. Os exemplos não foram considerados neste índice.

+, ler: mais
−, ler: menos
×, ler: multiplicação
/, ÷, ler: divisão
↑, ler: exponenciação
<, ≤, =, ≥, >, ≠, ler: <operador de relação>
≡, ⊃, ∨, ∧, ¬, ler: <operador lógico>
., ler: ponto
$_{10}$, ler: dez
:, ler: dois pontos
;, ler: ponto e vírgula
:=, ler: sinal de afectação
␣, ler: espaço
{ }, ler: parentesis
[ ], ler: parentesis de índices
‹ ›, ler: parentesis de cadeias

Centésimo número da «Gazeta de Matemática»

A Redacção da «Gazeta de Matemática» prepara o seu centésimo número.

Comemorando tal facto dirigiu a Institutos, Centros de Cálculo e Professores de outros países uma carta circular de que se transcreve o seguinte:

.....

«En outre il faut faire connaître au public portugais le role et l'importance de l'activité scientifique, notamment celle de la Mathématique, dans le développement économique des nations et des peuples. Dans ces conditions, la «Gazeta de Matemática» se fait un devoir d'y apporter sa contribution en présentant les points de vue de mathématiciens nationaux et étrangers sur une question si importante pour l'avenir scientifique de notre pays.

Nous serions donc très heureux de pouvoir compter sur votre collaboration à la tâche que nous nous sommes imposée; elle nous donnerait non seulement le précieux apport de votre expérience personnelle mais aussi des renseignements autorisés sur la solution trouvée à ce problème dans votre pays.

Sur ce dernier point, et dans le désir d'unification que vous voudrez bien comprendre, nous nous permettons de vous proposer une liste non limitative des points intéressant particulièremente notre pays:

1 — L'orientation et le développment actuels des diverses branches des mathématiques en vue des

a) problèmes nouveaux posés par la vie humaine dans la deuxiéme moitie du XX ème siècle.

b) solutions nouvelles qui résultent des découvertes scientifiques et des realisations téchnologiques dans la deuxiéme moitié du XX ème siècle.

c) techniques nouvelles de calcul et traitement des problèmes qui ont trouvé le jour dans la deuxiéme moitié du XX ème siècle.

2 — Les problèmes et les aspects des incidences réciproques entre la recherche scientifique, notammeut dans le domaine de la mathématique, et les réalisations économiques, dans les domaines de l'industrie et de l'agriculture. Particulièrement des besoins de l'activité scientifique dans les domaines de la recherche fondamentale orienteé, de la recherche appliquée et des operations de mise au point technique.

3 — L'orientation et le développement à imprimer, dans le futur prochain, à la recherche scientifique dans le domaine des mathématiques, en vue de l'évolution prévisible des aspects cités em 1. a), 1. b) et 1. c).

Il va de soi que la réponse à ces questions (et toutes les autres qui vous sembleraient importantes) aurait d'autant plus de valeur pour le public portugais qu'elle lui parviendrait par l'intermédiaire de votre irremplaçable expérience personnelle.

En sollicitant de votre bienveillance un article de cette nature, nous comptons sur votre esprit de collaboration scientifique et le désir d'apporter une contribution valable et très importante à la coopération et l'établissemente des bonnes relations personnelles parmi les scientifiques des divers pays.»

.....

A Redacção da «Gazeta de Matemática» espera com vivo interesse dos portugueses vinculados à actividade matemática no País, particularmente dos Professores a vários níveis de ensino, válida contribuição para a discussão e o esclarecimento dum problema vital para o mesmo Ensino.