

1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013 2014

BELO HORIZONTE, JUNHO DE 1975

ROBERTO DA SILVA BIGONHA

1. PROGRAM

<PROGRAM>

```

::= <BLOCK> . !
    <STAT NAME> : <BLOCK> . !
    <PROCEDURE DCL> . !
    <FUNCTION DCL> . !
    <MONITOR DCL> .

```

2. STATEMENT

<STATEMENT>

```

::= <STAT NAME> : <UNNAMED STAT> !
    <UNNAMED STAT>

```

<STAT NAME>

```

::= <IDENTIFIER>

```

<UNNAMED STAT>

```

::= <BLOCK> !
    <COMPOUND STAT> !
    <PROCEDURE STAT> !
    <EXIT STAT> !
    <REPETITIVE STAT> !
    <SELECTIVE STAT> !
    <ASSIGNMENT STAT> !
    <INPUT/OUTPUT STAT> !
    <EMPTY>

```

3. BLOCK

<BLOCK>

```

::= BEGIN <INTRODUCTION LIST> ; <STAT LIST> <END>

```

<INTRODUCTION LIST>

```

::= <INTRODUCTION> !
    <INTRODUCTION LIST> ; <INTRODUCTION>

```

<INTRODUCTION>

```

::= <TYPE DEFINITION> !
    <DECLARATION>

```

<STAT LIST>

```

::= <STATEMENT> !
    <STAT LIST> ; <STATEMENT>

```

<END>

```

::= END !
    END <STAT NAME>

```

4. COMPOUND STATEMENT

<COMPOUND STAT>

```

::= BEGIN <STAT LIST> <END>

```

5. PROCEDURE STATEMENT

<PROCEDURE STAT>

```

::= <PROC REFERENCE> ( <ACTUAL PAR LIST> ) !
    <PROC REFERENCE>

```

<PROC REFERENCE>

```

::= <PROCEDURE NAME> !

```

<ACTUAL PAR LIST>

```

::= <MONITOR NAME> . <PROCEDURE NAME>
::= <ACTUAL PAR> !

```

<ACTUAL PAR>

```

::= <ACTUAL PAR LIST> , <ACTUAL PAR>
::= <EXP>

```


6. EXIT STATEMENT

<EXIT STAT>

::= EXIT <STAT NAME> !
 EXIT <PROCEDURE NAME> !
 EXIT <FUNCTION NAME> WITH <EXP>

7. REPETITIVE STATEMENT

<REPETITIVE STAT>

::= <REPEAT STAT> !
 <WHILE STAT> !
 <FOR STAT>

7.1 REPEAT STATEMENT

<REPEAT STAT>

::= REPEAT <STATEMENT> UNTIL <EXP>

7.2 WHILE STATEMENT

<WHILE STAT>

::= WHILE <EXP> DO <STATEMENT>

7.3 FOR STATEMENT

<FOR STAT>

::= <FOR CLAUSE> DO <STATEMENT>

<FOR CLAUSE>

::= <FOR PART> <FROM PART> <TO PART> <BY PART>

<FOR PART>

::= FOR <VARIABLE> !

<FROM PART>

::= FROM <EXP> !

<TO PART>

::= TO <EXP> !

<BY PART>

::= <EMPTY>
 ::= BY <EXP> !
 ::= <EMPTY>

8. SELECTIVE STATEMENT

<SELECTIVE STAT>

::= <IF STAT> !
 <WHEN STAT> !
 <CASE STAT> !
 <INDEX STAT>

8.1 IF STATEMENT

<IF STAT>

::= <IF CLAUSE> <STATEMENT> ELSE <STATEMENT>

<IF CLAUSE>

::= IF <EXP> THEN

8.2 WHEN STATEMENT

<WHEN STAT>

::= <WHEN CLAUSE> <STATEMENT>

<WHEN CLAUSE>

::= WHEN <EXP> THEN


```

<TYPE DEFINITION> ::= <RECORD TYPE DEF> !
                   <ARRAY RECORD TYPE DEF>
                   <ARRAY TYPE DEF>

```

12.1 RECORD TYPE DEFINITION

```

<RECORD TYPE DEF> ::= RECORD <RECORD TYPE NAME> ( <FIELD SPECIF LIST> )
<RECORD TYPE NAME> ::= <IDENTIFIER> !
<FIELD SPECIF LIST> ::= <FIELD SPECIF> !
                   <FIELD SPECIF LIST> ; <FIELD SPECIF>
<FIELD SPECIF> ::= <TYPE> <ELEMENT LIST>
<ELEMENT LIST> ::= <ELEMENT> !
                   <ELEMENT LIST> , <ELEMENT>
<ELEMENT> ::= <IDENTIFIER> !
                   <IDENTIFIER> ( <RANGE> )
<RANGE> ::= <CONSTANT> : <CONSTANT>

```

12.2 ARRAY OF RECORD TYPE DEFINITION

```

<ARRAY RECORD TYPE DEF> ::= <ARRAY SPECIF> <RECORD TYPE DEF>
<ARRAY SPECIF> ::= [ <BOUND PAIR LIST> ]
<BOUND PAIR LIST> ::= <BOUND PAIR> !
                   <BOUND PAIR LIST> , <BOUND PAIR>
<BOUND PAIR> ::= : !
<SIGNED INTEGER CTE> ::= <INTEGER CTE> !
                   - <INTEGER CTE> !
                   + <INTEGER CTE>

```

12.3 ARRAY TYPE DEFINITION

```

<ARRAY TYPE DEF> ::= ARRAY <ARRAY SPECIF> <SIMPLE TYPE> <ARRAY TYPE NAME>
<SIMPLE TYPE> ::= INTEGER !
                   REAL !
                   BOOLEAN !
                   CHAR !
                   ALPHA !
                   REGISTER !
                   ION <ION LENGTH> !
                   UNPACKED ION <ION LENGTH> !
                   POINTER ( <TYPE LIST> )
<ARRAY TYPE NAME> ::= <IDENTIFIER>
<ION LENGTH> ::= <INTEGER CTE>
<TYPE LIST> ::= <TYPE> !
                   <TYPE LIST> , <TYPE>
<TYPE> ::= <SIMPLE TYPE> !
                   <RECORD TYPE DEF> !
                   <ARRAY RECORD TYPE DEF> !
                   <ARRAY TYPE DEF> !
                   <ARRAY SPECIF> <TYPE NAME> !
                   <ARRAY SPECIF> <SIMPLE TYPE> !
                   <TYPE NAME>
<TYPE NAME> ::= <IDENTIFIER>

```


<DECLARATION> ::= <VARIABLE DCL> !
 <PROCEDURE DCL> !
 <FUNCTION DCL> !
 <MONITOR DCL> !

13.1 VARIABLE DECLARATION

<VARIABLE DCL> ::= <TYPE> <ELEMENT LIST> !

13.2 PROCEDURE DECLARATION

<PROCEDURE DCL> ::= <SCOPE MODE> <PROCEDURE HEADING> : <BODY>
 <SCOPE MODE> ::= SHARED !
 <PROCEDURE HEADING> ::= PROCEDURE (<FORMAL PAR LIST>) <PROCEDURE NAME> :
 <FORMAL PAR LIST> ::= <FORMAL PAR> !
 <FORMAL PAR> ::= <FORMAL PAR LIST> , <FORMAL PAR>
 <CALLING MODE> ::= <CALLING MODE> <VARIABLE DCL>
 <CALLING MODE> ::= VALUE !
 RESULT !
 VALUE RESULT !
 REFERENCE !
 <PROCEDURE NAME> ::= <IDENTIFIER>
 <BODY> ::= <STATEMENT> !
 EXTERNAL

13.3 FUNCTION DECLARATION

<FUNCTION DCL> ::= <SCOPE MODE> <FUNCTION HEADING> <BODY>
 <FUNCTION HEADING> ::= <SIMPLE TYPE> FUNCTION (<FORMAL PAR LIST>)
 <FUNCTION NAME> ::= <IDENTIFIER>

13.4 MONITOR DECLARATION

<MONITOR DCL> ::= <MONITOR HEADING> <INTRODUCTION LIST> <MONEND>
 <MONITOR HEADING> ::= MONITOR <MONITOR NAME> :
 <MONITOR NAME> ::= <IDENTIFIER>
 <MONEND> ::= MONEND !
 MONEND <MONITOR NAME>

14. EXPRESSION

<EXP> ::= <SIMPLE EXP> !
 <CASE EXP> !
 <INDEX EXP> !
 <IF EXP>

CENTRO DE COMPUTAÇÃO DA U.F.M.G.

14.1 SIMPLE EXPRESSION

simple exp

<SIMPLE EXP>

::= <TERM> | <EXP> ELSE <EXP>

<TERM>

::= <SIMPLE EXP> <TERM>

<SIMPLE TERM>

::= <TERM> <ADD OP> <SIMPLE TERM>

<CONSTANT>

::= <FACTOR> !

<FACTOR>

::= <SIMPLE TERM> <MUL OP> <FACTOR>

<PRIMARY>

::= <PRIMARY> !

<FACTOR> <EXP OP> <PRIMARY> !

<UNARY OP> <PRIMARY>

::= <VARIABLE> !

<CONSTANTE> !

<FUNCTION DESIGN> !

(<EXP>)

::= < !

LT !

<= !

LE !

> !

GT !

>= !

GE !

= !

NE !

::= + !

= !

OR !

XOR !

::= * !

/ !

AND !

SHR !

SHL !

::= ** !

MOD !

CAT !

::= + !

- !

NOT !

REF !

ABS !

14.2 CASE EXPRESSION

<CASE EXP>

::= <CASE CLAUSE> <CASE EXP BODY>

<CASE EXP BODY>

::= <CASE EXP LIST> , OUT => <EXP>

<CASE EXP LIST>

::= <EXP> => <EXP> !

<CASE EXP LIST> , <EXP> => <EXP>

14.3 INDEX EXPRESSION

<INDEX EXP>

::= <INDEX CLAUSE> <INDEX EXP BODY>

<INDEX EXP BODY>

::= <INDEX EXP LIST> , OUT => <EXP>

<INDEX EXP LIST>

::= <INTEGER CTE> => <EXP> !

<INDEX EXP LIST> , <INTEGER CTE> => <EXP>

<IF EXP>

::= <IF CLAUSE> <EXP> ELSE <EXP>

15. CONSTANT

<CONSTANT>

::= <INTEGER CTE> !

<REAL CTE> !

<LOGICAL CTE> !

<ION CTE>

15.1 INTEGER CONSTANT

<INTEGER CTE>

::= <DIGIT> !

<INTEGER CTE> <DIGIT>

15.2 REAL CONSTANT

<REAL CTE>

::= <UNSCALED REAL> !

<UNSCALED REAL> <SCALE FACTOR> !

<INTEGER CTE> <SCALE FACTOR>

::= <INTEGER CTE> . <INTEGER CTE> !

<INTEGER CTE> .

::= E <INTEGER CTE> !

E + <INTEGER CTE> !

E - <INTEGER CTE>

15.3 LOGICAL CONSTANT

<LOGICAL CTE>

::= TRUE !

FALSE

15.4 ION CONSTANCE

<ION CTE>

::= B <BINARY STRING> !

O <OCTAL STRING> !

X <HEXA STRING> !

<STRING>

<BINARY STRING>

::= <BINARY DIGIT> !

<BINARY STRING> <BINARY DIGIT>

<BINARY DIGIT>

::= 0 !

1

<OCTAL STRING>

::= <OCTAL DIGIT> !

<OCTAL STRING> <OCTAL DIGIT>

<OCTAL DIGIT>

::= <BINARY DIGIT> !

2 !

3 !

4 !

5 !

6 !

7

<HEXA STRING>

::= <HEXA DIGIT> !

<HEXA STRING> <HEXA DIGIT>

<HEXA DIGIT> ::= <DIGIT> !
 <LETTER> A !
 B !
 C !
 D !
 E !
 F !
 <DIGIT> ::= <OCTAL DIGIT> !
 8 !
 9 !
 <STRING> ::= ' <PROPER STRING> ' !
 <PROPER STRING> ::= ANY SEQUENCE OF CHARACTERS NOT CONTAINING '

16. VARIABLE

<VARIABLE> ::= <SIMPLE VAR> !
 <FIELD DESIGN>

16.1 SIMPLE VARIABLE

<SIMPLE VAR> ::= <IDENTIFIER> !
 <SUBSCRIPTED VAR> ::= <IDENTIFIER> <SUBSCRIPTS> !
 <SUBSCRIPTS> ::= [<EXP LIST>] *Field design? c subscript?*
 <EXP LIST> ::= <EXP> !
 <EXP LIST> ::= <EXP LIST> , <EXP>
 <POINTER QUALIF> ::= <QUALIFIER> @ <TYPE> !
 <SIMPLE VAR> @ <TYPE> !
 <POINTER QUALIF> @ <TYPE>

16.2 FIELD DESIGNATOR

<FIELD DESIGN> ::= <QUALIFIER> !
 <QUALIFIER> ::= <SIMPLE VAR> , <FIELD IDENT> !
 <FIELD IDENT> ::= <SIMPLE VAR> !
 <FIELD IDENT> ::= <FIELD IDENT> , <SIMPLE VAR>

17. FUNCTION DESIGNATOR

<FUNCTION DESIGN> ::= <FUNCTION REFERENCE> (<ACTUAL PAR LIST>) !
 <FUNCTION REFERENCE> ::= <FUNCTION REFERENCE>
 <FUNCTION REFERENCE> ::= <FUNCTION NAME> !
 <FUNCTION NAME> ::= <MONITOR NAME> , <FUNCTION NAME>

18. IDENTIFIER

<IDENTIFIER> ::= <LETTER> !
 <IDENTIFIER> <LETTER> !
 <IDENTIFIER> <DIGIT>

B !
X !
Y !
Z

19.COMMENT

ANY SEQUENCE OF CHARACTER BETWEEN #'S

3. ESCOPO DE VARIÁVEIS EM MONITOR E PROCEDIMENTOS E FUNÇÕES DECLARADAS

DENTRO DE UM MONITOR.

INTERAMENTE A UM PROCEDIMENTO OU FUNÇÃO CONDIÇÃO AOS DECLARADOS EM SEUS PARÂMETROS E A VARIÁVEIS LOCAIS. ISTO É, A DECLARAÇÃO EM SEU BLOCO DENTRO DO ESCOPO DE UM PROCEDIMENTO OU FUNÇÃO LIMITA O ACESSO A VARIÁVEIS E FUNÇÕES DE REFERÊNCIA DE UMA VARIÁVEL LOCAL OU EXTERNA, EXCETO AS PASSÁVEIS DE PARÂMETROS.

4. ESCOPO DE VARIÁVEIS EM MONITOR E PROCEDIMENTOS E FUNÇÕES DECLARADAS

DENTRO DE UM MONITOR.

OS PROCEDIMENTOS E FUNÇÕES COM ATRIBUTO "MONITOR" TEM A SEU BLOCO DE CONEXÕES ADICIONAIS QUE CONTEM A DECLARAÇÃO DE VARIÁVEIS E FUNÇÕES. KATIS PROCEDIMENTOS E FUNÇÕES ADICIONAIS SÃO CONECTADOS DENTRO DO MONITOR.

OS PROCEDIMENTOS E FUNÇÕES DECLARADOS DENTRO DO MONITOR TEM O DIREITO AO ACESSO AS VARIÁVEIS GLOBAIS DO MONITOR. ESTA É A ÚNICA SITUAÇÃO EM QUE UM PROCEDIMENTO OU FUNÇÃO PODE REFERENCIAR UMA VARIÁVEL QUE NÃO É UM PARÂMETRO FORMAL OU VARIÁVEL LOCAL DO PROCEDIMENTO OU FUNÇÃO. POR OUTRO LADO, NÃO É PERMITIDO ACESSO EXTERNO AO MONITOR.

1. IDENTIFICACAO DE UM PROGRAMA EM LAPA

QUANDO UM PROGRAMA E' UM <STATEMENT> O SEU NOME E' O <STAT NAME> E SE ESTE FOR OMITIDO TOMA-SE "BY DEFAULT" O NOME "MAINLAPA". NO CASO DE <PROCEDURE DCL>, <FUNCTION DCL> OU <MONITOR DCL> TEM-SE A CRIACAO DE SUBPROGRAMA E SEU NOME E' O NOME DO SUBPROGRAMA DEFINIDO. EM PARTICULAR NO CASO DE MONITOR, OS NOMES DOS PROCEDIMENTOS E/OU FUNCOES COM ATRIBUTO "SHARED" INTERNOS A ELE, TAMBEM SAO CONSIDERADOS EXTERNOS.

2. ESCOPO DE VARIAVEIS EM RELACAO A BLOCOS

DENTRO DE UM BLOCO TEM-SE VARIAVEIS LOCAIS, QUE SAO ALOCADAS NO MOMENTO DA ATIVACAO DO BLOCO E PODE SE TER AINDA VARIAVEIS GLOBAIS QUE POR SUA VEZ SAO LOCAIS A ALGUM BLOCO ENVOLVENTE AO BLOCO EM QUESTAO. PORTANTO UMA VARIAVEL DECLARADA EM UM BLOCO E' LOCAL A ESTE E GLOBAL A TODOS OS BLOCOS MAIS INTERNOS, A NAO SER QUE HAJA INTERNAMENTE A DECLARACAO DE OUTRA VARIAVEL COM O MESMO NOME.

3. ESCOPO DE VARIAVEIS EM RELACAO A PROCEDIMENTOS E FUNCOES DECLARADAS

FORA DE UM MONITOR.

INTERNAMENTE A UM PROCEDIMENTO OU FUNCAO SOMENTE SAO CONHECIDOS OS SEUS PARAMETROS E AS VARIAVEIS LOCAIS, ISTO E', DECLARADAS EM ALGUM BLOCO DENTRO DO PROCEDIMENTO OU FUNCAO. NESTES PROCEDIMENTOS OU FUNCOES A UNICA POSSIBILIDADE DE REFERENCIAR UMA VARIAVEL DECLARADA EXTERNAMENTE E' ATRAVES DA PASSAGEM DE PARAMETROS.

4. ESCOPO DE VARIAVEIS EM RELACAO A PROCEDIMENTOS E FUNCOES DECLARADAS

DENTRO DE UM MONITOR

OS PROCEDIMENTOS E FUNCOES COM ATRIBUTO "SHARED" TEM OS SEUS NOMES CONHECIDOS NO BLOCO QUE CONTEM A DECLARACAO DO MONITOR. OS DEMAIS PROCEDIMENTOS E FUNCOES SOMENTE SAO CONHECIDOS DENTRO DO MONITOR.

OS PROCEDIMENTOS E FUNCOES DECLARADAS DENTRO DO MONITOR TEM O DIREITO AO ACESSO AS VARIAVEIS GLOBAIS DO MONITOR. ESTA E' A UNICA SITUACAO EM QUE UM PROCEDIMENTO OU UMA FUNCAO PODE REFERENCIAR UMA VARIAVEL QUE NAO SEJA UM PARAMETRO FORMAL OU VARIAVEL LOCAL DO PROCEDIMENTO OU FUNCAO. POR OUTRO LADO NAO E' PERMITIDO ENCAIXAMENTO DE MONITORES.

5. ATIVACAO E DESATIVACAO DE UM BLOCO

UM BLOCO E' ATIVADO NO MOMENTO EM QUE O SEU PROLOGO (BEGIN) SEJA PROCESSADO PELA EXECUCAO SEQUENCIAL DO PROGRAMA. POR ATIVACAO DE UM BLOCO ENTENDE-SE A REQUISICAO DE ESPACO PARA ALOCACAO DE SUAS VARIAVEIS LOCAIS.

UM BLOCO E' DESATIVADO QUANDO O SEU EPILOGO (END) E' PROCESSADO, O QUE CORRESPONDE A LIBERACAO DO ESPACO OCUPADO POR SUAS VARIAVEIS LOCAIS.

6. ATIVACAO E DESATIVACAO DE PROCEDIMENTOS OU FUNCOES

UM PROCEDIMENTO OU FUNCAO E' ATIVADO PELA REFERENCIA AO SEU NOME. O PROCESSO DE ATIVACAO COMPREENDE AS SEGUINTE ETAPAS:

A) REQUISICAO DE ESPACO PARA OS PARAMETROS FORMAIS. (ENDERECO OU VALOR)

B) REQUISICAO DE ESPACO PARA OS PARAMETROS IMPLICITOS SA ENDERECO DE RETORNO, ENDERECO DA AREA DE DADOS DO MONITOR (SE FOR O CASO) E OUTROS PARAMETROS NECESSARIOS PARA A GERENCIA DE MEMORIA.

C) TRANSFERENCIA DO CONTROLE DO PROCESSADOR PARA A PRIMEIRA INSTRUCAO DO PROCEDIMENTO.

A DESATIVACAO OCORRE QUANDO FOR PROCESSADO O COMANDO EXIT, MOMENTO EM QUE AS AREAS REQUISITADAS SAO LIBERADAS E O CONTROLE VOLTA AO Ponto de Chamada.

7. ATIVACAO E DESATIVACAO DO MONITOR

A ATIVACAO DO MONITOR COMPREENDE DUAS FASES:

A) A PRIMEIRA FASE DA ATIVACAO OCORRE QUANDO O BLOCO QUE CONTEM O MONITOR E' ATIVADO, MOMENTO EM QUE ESPACO PARA AS VARIAVEIS DECLARADAS DENTRO DO MONITOR E' REQUISITADO.

B) A SEGUNDA FASE OCORRE QUANDO E' FEITA A ATIVACAO DE UM PROCEDIMENTO OU FUNCAO "SHARED" DO MONITOR.

A DESATIVACAO DE UM MONITOR OCORRE SIMULTANEAMENTE COM A DESATIVACAO DO BLOCO QUE O CONTEM. O RETORNO DE UMA PROCEDURE DO MONITOR NAO DESATIVA O MONITOR.

8. DECLARACOES E DEFINICOES DE TIPO

COMO DECLARACAO ENTENDE-SE POR ASSOCIAR UM ATRIBUTO OU TIPO A UM IDENTIFICADOR E PORTANTO CRIAR UMA VARIAVEL, A QUAL REPRESENTARA, DURANTE O TEMPO DE EXECUCAO DO PROGRAMA O ENDERECO DE UMA AREA DE MEMORIA, QUE SERA MANIPULADA PELO PROGRAMA. POR OUTRO LADO POR DEFINICAO DE TIPO ENTENDE-SE POR ASSOCIAR UMA FORMA DE ESTRUTURA DE DADOS A UM IDENTIFICADOR. UMA ESTRUTURA DE DADOS CONSTITUI-SE POR REUNIR UM CONJUNTO ORDENADO DE COMPONENTES, ONDE CADA UM POSSUE UM TIPO. E O IDENTIFICADOR ASSOCIADO A ESTA ESTRUTURA PASSA A CONSTITUIR UM NOVO TIPO OU ATRIBUTO QUE PODERA SER UTILIZADO PARA DECLARAR VARIAVEIS. OBVIAMENTE DEFINIR UM TIPO NAO IMPLICA EM ALOCACAO DE MEMORIA PELO COMPILADOR.

9. MODOS DE PASSAGEM DE PARAMETROS DE PROCEDIMENTO OU FUNCAO

VALUE - O VALOR DESCRITO PELO PARAMETRO ATUAL E' COPIADO NA AREA DE DADOS DO PROCEDIMENTO OU FUNCAO. FUNCIONA DAI' PARA FRENTE COMO UMA VARIAVEL LOCAL. E' O "DEFAULT" PARA VARIAVEIS ESCALARES

RESULT - O PROCEDIMENTO OU FUNCAO TRATA O PARAMETRO FORMAL COMO SE FOSSE UMA VARIAVEL LOCAL E NO MOMENTO DO RETORNO O VALOR CORRENTE DO PARAMETRO FORMAL E' COPIADO NO ENDEREÇO DESCRITO PELO PARAMETRO ATUAL.

VALUE RESULT - E' UMA COMBINACAO DOS DOIS ANTERIORES.

REFERENCE - O ENDEREÇO DESCRITO PELO PARAMETRO ATUAL E' COPIADO DENTRO DA AREA DE DADOS DO PROCEDIMENTO OU FUNCAO NA POSICAO CORRESPONDENTE AO PARAMETRO FORMAL E ENTAO QUALQUER REFERENCIA AO PARAMETRO E' FEITA INDIRETAMENTE AO ENDEREÇO DO PARAMETRO FORMAL. E' O MODO "DEFAULT" PARA AGREGADO DE DADOS.

10. DEFINICAO DE ARQUIVOS

EXISTE NO COMPILADOR LAPA O SEGUINTE TIPO RECORD DEFINIDO IMPLICITAMENTE:

"RECORD" FILE ("INTEGER" FILENUMBER, RECSIZE, KEY)

SEND0:

"FILE" - UMA PALAVRA RESERVADA

"FILENUMBER" - NUMERO LOGICO DO ARQUIVO

"RECSIZE" - TAMANHO DO REGISTRO DE TAMANHO FIXO

"KEY" - NUMERO DO REGISTRO QUE DEVE SER ACESSADO

USANDO O TIPO "FILE" DECLARA-SE OS ARQUIVOS (FILENAMES) E ATRAVES DE DE COMANDOS DE ATRIBUICAO PREENCHE-SE OS CAMPOS DO RECORD COM ATRIBUTO FILE. PARA ABRIR OS ARQUIVOS EXISTE A PRIMITIVA "OPEN" (FILENAME) E PARA FECHAR A "CLOSE" (FILENAME).

NOS ARQUIVOS EM MEIO MAGNETICO A TRANSMISSAO E' FEITA SEM CONVERSAO DE DADOS. O ENDEREÇO DO PRIMEIRO REGISTRO E' FEITO INICIALMENTE IGUAL A 1, SENDO PERMITIDO AO USUARIO MODIFICAR A QUALQUER MOMENTO ESTE VALOR. APOS CADA ACESSO TEM-SE "KEY=KEY + 1".

11. OPERADORES EM EXPRESSOES

AS EXPRESSOES EM LAPA SAO CLASSIFICADAS EM ARITMETICAS, LOGICAS OU IONICAS, DEPENDENDO DO TIPO DE DADO PRODUZIDO NA SUA AVALIACAO. ESTA CLASSIFICACAO NAO FOI CONSIDERADA NA GRAMATICA PARA QUE FOSSE DIMINUIDO O SEU TAMANHO.

NAO E' PERMITIDO A MISTURA DE TIPOS DE OPERANDOS EM UMA EXPRESSAO; QUANDO NECESSARIO, A CONVERSAO DEVE SER FEITA EXPLICITAMENTE PELO PROGRAMA. A PRECEDENCIA DOS OPERADORES, RESPEITADA A HIERARQUIA DADA PELOS PARENTHESES E' DADA PELA TABELA:

ARITMETICOS

1- +, -, ABS, REF
2- **, MOD
3- *, /, SHR, SHL
4- +, -
5- >, >=, <, <=, =, NE

IONICOS

NOT
CAT
AND, SHR, SHL
OR, XOR
>, >=, <, <=, =, NE

ONDE:

+ - OPERADOR DE ADICAO OU SINAL (BINARIO OU UNARIO)
- - OPERADOR DE SUBTRACAO OU SINAL (BINARIO OU UNARIO)
ABS - VALOR ABSOLUTO (UNARIO)
REF - OBTEM O ENDERECC DE UMA VARIABEL (UNARIO)
** - EXPONENCIACAO (BINARIO)
* - OPERADOR DE MULTIPLICACAO (BINARIO)
/ - OPERADOR DE DIVISAO (BINARIO)
SHR - DESLOCAMENTO DE BIT PARA A DIREITA (BINARIO) .O PRIMEIRO OPE-
RANDO E' O FONTE E O SEGUNDO DA' O NUMERO DE BITS.
SHL - IDEM PARA A ESQUERDA.
>, >=, <, <=, =, NE - OPERADORES DE RELACAO (BINARIO)
NOT - OPERADOR LOGICO (UNARIO)
CAT - CONCATENACAO (BINARIO)
AND - OPERADOR LOGICO (BINARIO)
OR - OPERADOR LOGICO (BINARIO)
XOR - OPERADOR LOGICO (OU EXCLUSIVO) (BINARIO).

12. ALOCACAO DE MEMORIA PARA OS TIPOS ELEMENTARES

TIPO	DESCRICAO
INTEGER	UMA POSICAO DE MEMORIA (PALAVRA)
REAL	DUAS POSICOES DE MEMORIA (PALAVRAS)
POINTER	UMA POSICAO DE MEMORIA (PALAVRA)
REGISTER	UM REGISTRADOR RAPIDO DE USO GERAL E' SEMPRE UM "INTEGER".
ION N	N BITS CONSECUTIVOS EM UMA OU MAIS PALAVRAS.
UNPACKED ION N	N PALAVRAS ONDE SOMENTE E' CONSIDERADO O PRIMEIRO BIT DE CADA PALAVRA.
BOOLEAN	MESMO QUE ION 1
CHAR	MESMO QUE ION 8
ALPHA	MESMO QUE ION 24

13. ALOCACAO DE MEMORIA PARA OS TIPOS DE DADOS ESTRUTURADOS

OCORRE COMO UMA REPETICAO DE TIPOS ELEMENTARES.

14. COMANDO ASSIGNMENT

O RESULTADO DA EXPRESSAO DO LADO DIREITO E' ATRIBUIDO A(AS) VARIABEL (VARIAVEIS) DO LADO ESQUERDO SEM CONVERSAO DE DADOS. UMA MENSAGEM SE-
RA' EMITIDA DURANTE A COMPILACAO SE OS TIPOS FOREM DIFERENTES. QUANDO
OCORRER MAIS QUE UMA VARIABEL DO LADO ESQUERDO A AVALIACAO DE ENDERE-
CO SERA' FEITA DA DIREITA PARA A ESQUERDA.

EXEMPLO:

I:= 1;
A[I]:=I:=2;

EQUIVALE A

I:=1;
I:=2;
A[I]:=2;

15. COMANDO EXIT

15. COMANDOS SELETIVOS

A CONSTRUÇÃO "EXIT <STAT NAME>" CAUSA O TERMINO DA EXECUÇÃO DO <STATEMENT>. O PROXIMO COMANDO A SER EXECUTADO SERA O SEGUINTE DO COMANDO RECENTE TERMINADO.

A CONSTRUÇÃO "EXIT <PROCEDURE NAME>" CAUSA A DESATIVACAO DO PROCEDIMENTO.

A CONSTRUÇÃO "EXIT <FUNCTION NAME> WITH <EXP>" CAUSA A DESATIVACAO DA FUNCAO RETORNANDO O VALOR CALCULADO DE <EXP>.

16. COMANDO CHAMADA DE PROCEDIMENTO

OS PROCEDIMENTOS (ANALOGAMENTE PARA FUNCAO) QUE ESTAO DENTRO DE UM MONITOR E QUE PODEM SER ATIVADOS NO BLOCO ONDE O MONITOR FOI DECLARADO, DEVEM NO MOMENTO DE CHAMADA TER SEUS NOMES PRECEDIDOS PELO NOME DO MONITOR (<MONITOR NAME>).

PROCEDIMENTOS OU FUNCOES NAO PODEM SER PASSADOS COMO PARAMETROS.

NO CASO DO PARAMETRO SER DO TIPO RECORD, ADMITE-SE QUE O PARAMETRO ATUAL SEJA MAIOR QUE O CORRESPONDENTE PARAMETRO FORMAL E NESTE CASO A PARTE INICIAL DO RECORD DO PARAMETRO ATUAL DEVE COINCIDIR COM O FORMAL.

QUANDO UM ARRAY E' PARAMETRO FORMAL NAO SE PODE ESPECIFICAR QUAIS SAO OS VALORES DOS INDICES, MAS SOMENTE O NUMERO DE INDICES E SERA SEMPRE PASSADO "BY REFERENCE".

17. COMANDOS REPETITIVOS

NOS COMANDOS "REPEAT" E "WHILE", <EXP> DEVEM PRODUZIR UM RESULTADO BOOLEANO (TRUE OU FALSE).

O COMANDO "REPEAT" EXECUTA <STATEMENT> ATÉ QUE <EXP> ASSUMA O VALOR "TRUE". O TESTE E' FEITO NO FINAL.

O COMANDO "WHILE" EXECUTA <STATEMENT> ENQUANTO <EXP> FOR TRUE. O TESTE E' FEITO NO INICIO.

NO COMANDO "FOR" TEM-SE:

- .<VARIÁVEL> DO "FOR" (CONTADOR) NÃO FICA DISPONÍVEL APÓS A EXECUÇÃO DO COMANDO.
- .SE FALTAR <FROM PART> ASSUME-SE 0
- .SE FALTAR <TO PART> ASSUME-SE +INFINITO
- .SE FALTAR <BY PART> ASSUME-SE 1
- .SE FALTAR A <FOR CLAUSE>, OS "DEFAULT" SÃO ASSUMIDOS E TEM-SE UM "DO FOREVER".

18. COMANDOS SELETIVOS

NOS COMANDOS "IF" E "WHEN" <EXP> DEVE TER RESULTADO BOOLEANO. O "WHEN" É O COMANDO "IF" SEM O "ELSE".

O COMANDO "CASE" PESQUISA SEQUENCIALMENTE O VALOR DA <EXP> DA <CASE STAT LIST> ATÉ ENCONTRAR UMA QUE SATISFAÇA E ENTÃO EXECUTA O COMANDO QUE ESTÁ NA FRENTE DA ->. A CLAUSULA "OUT" É OBRIGATORIA.

O COMANDO "INDEX" DESVIA O CONTROLE DIRETAMENTE NÃO GERANDO COMPARAÇÃO SUCESSIVAS.

19. COMANDOS DE ENTRADA/SAÍDA

A AUSÊNCIA DE <FILE DESIGN> IMPLICA NA LEITORA DE CARTÕES (READ) OU NA IMPRESSORA (WRITE).

A AUSÊNCIA DE <FORMAT PART> IMPLICA NO FORMATO LIVRE.

20. EXPRESSÕES

QUANTO AO TIPO DE RESULTADOS AS EXPRESSÕES PODEM SER ARITMÉTICAS (INTEIRA OU REAL) OU LÓGICAS E QUANTO A MANEIRA COM QUE É AVALIADA É DITA:

.EXPRESSÕES SIMPLES - AVALIADA SEGUNDO AS PRIORIDADES DOS OPERADORES E PARENTESES.

.EXPRESSÕES CONDICIONAIS - AS QUAIS PODEM SER:

- EXPRESSÕES IF
- EXPRESSÕES CASE OU
- EXPRESSÕES INDEX.

.EXPRESSÕES MISTAS - QUANDO REUNIR DOIS OU MAIS DOS TIPOS ACIMA.

O RESULTADO DE UMA EXPRESSÃO "IF" DEPENDE DO RESULTADO DE UMA EXPRESSÃO BOOLEANA.

PARACAO DE UMA EXPRESSAO COM UM DADO CONJUNTO DE EXPRESSOES ANALO-
GAMENTE PARA A "INDEX".

21. CONSTANTES IONICAS

SAO ALOCADAS NA MEMORIA SEMPRE DO MODO COMPACTADO E PODEM SER DO
TAMANHO MAXIMO POSSIVEL PARA UM INTEIRO.

22. OS ASPECTOS DA PROGRAMACAO CONCORRENTE SERAO TRATADOS EM OUTRA OPOR-
TUNIDADE.