C-PASCAL: SUPORTE PARA DESENVOLVIMENTO DE PROJETOS EM MICROCOMPUTADORES

José Carlos Martins Leite

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS (M.Sc.)

Aprovada por:

Profª LÍGIA ALVES BARROS

Profª SUELI MENDES DOS SANTOS

Prof. JOSÉ LUCAS MOURÃO RANGEL NETTO

Prof. EBER ASSIS SCHMITZ

RIO DE JANEIRO, RJ - BRASIL
FEVEREIRO DE 1981

LEITE, JOSÉ CARLOS MARTINS

C-PASCAL: Suporte para Desenvolvimento de Projetos em Microcomputadores |Rio de Janeiro| 1981.

VII, 123 p. 29,7 cm (COPPE-UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1981)

Tese - Univ. Fed. Rio de Janeiro. Fac. Engenharia

1. Assunto: Compiladores e Linguagens Formais I. COPPE/UFRJ II. Título (série).

Ā

Ana Leite.

AGRADECIMENTOS

À Profª LÍGIA ALVES BARROS, pela excelente orientação, apoio, dedicação e todo esforço empregado no desenvolvimento deste trabalho.

À Profª SUELI MENDES DOS SANTOS, pelos incentivos e todos os conhecimentos transmitidos durante os cursos.

Aos Profs. JOSÉ LUCAS MOURÃO RANGEL NETO, ESTEVAM DE SIMONE, JANO MOREIRA DE SOUZA, LÍDIA MICHELA DE ANDA, pelos conhecimentos ministrados, bem como pelas valorosas orientações prestadas nos trabalhos do curso.

Ao CEPEL, e em particular aos Engs. ANTÔNIO LUIS BOGADO e JOÃO GUEDES DE CAMPOS BARROS, pelos incentivos e recursos recebidos para elaboração deste trabalho.

Aos colegas do CEPEL, MOTTA, LÚCIA, PAULO ROBERTO, CRISTINA, pelo apoio desinteressado dado a este trabalho.

À MARIA LIA, pela dedicação e empenho na datilografia desta tese.

RESUMO

A crescente utilização dos microcomputadores para aquisição e controle de dados gerados por equipamentos de simulação, requer grandes esforços na programação das rotinas para o tratamento destes resultados. Tais sistemas, e em particular os de baixo custo, necessitam de métodos mais simples, que a linguagem de máquina, para uma programação rápida, prática e segura sem prejuízo da eficiência do sistema. As linguagens de programação de alto nível (ALGOL, PL/I, PASCAL, etc...) são bastantes dispendiosas para serem implementadas nesta categoria de microcomputadores, por terem sido projetadas para equipamentos de maior porte.

O objetivo desta tese é oferecer uma linguagem estruturada e recursiva para ser usada em microcomputadores de baixo custo. A ênfase dada neste trabalho foi a tradução do código intermediário para um código executável diretamente pelo microprocessador. A linguagem C-PASCAL, elemento básico do SUPORTE (COMPILADOR, INTERPRETADOR e TRADUTOR), foi especificada com base no PASCAL (WIRTH) e implementada para o microprocessador INTEL 8Ø85.

ABSTRACT

Microcomputers in recent years have largely been used for acquisition and control of data generated by equipments of simulation. This utilization requires a great amount of effort in programming for the management of the results obtained. Microcomputer systems, particularly low cost systems, require programming methods, simpler, faster and safer than machine languages without affecting the performance of the system. Most of the high level programming languages (ALGOL, PL/I, PASCAL, etc...) are too costly to be implemented in such systems.

The main aim of this work is to make available a structured and recursive language to be used in low cost microcomputer systems. This thesis emphasizes the translation of intermediate code into a code which runs directly in the microcomputer. C-PASCAL language, the basic element of the SUPORT (compiler, interpreter and translator), was specified with basis in the PASCAL language (WIRTH) and implemented in the INTEL 8085.

ÍNDICE

CAPÍTULO I

INTRODUÇÃO

I.1 - OBJETIVO DA TESE

A Tese de Mestrado C-PASCAL: SUPORTE PARA DESENVOLVIMENTO DE PROJETOS EM MICRO-COMPUTADORES tem como objetivo imediato, colocar a disposição dos usuários de micro-computadores, uma linguagem estruturada que permita o desenvolvimento de projetos que utilizem micro-processadores. O segundo objetivo é oferecer condições para projetar um compilador completo da linguagem PASCAL[1] , trabalhando no próprio micro-computador.

Este trabalho foi desenvolvido com apoio, técnico e científico, do CENTRO DE PESQUISAS DE ENERGIA ELÉTRICA - CEPEL, motivado pela inexistência de 'software' nacional para micro-computadores, bem como da necessidade de uma linguagem estruturada para elaboração de projetos nesta área.

Em particular, o Suporte C-PASCAL foi desenvolvido para equiparar sistemas de baixo custo[2], capazes de suprir as necessidades dos laboratórios das Universidades Brasileiras, que necessitam de aparelhagem para pesquisa e desenvolvimento na área de microprocessadores e na maioria das vezes não dispõem de recursos suficientes para importar equipamentos completos. Portanto, o Suporte C-PASCAL é uma alternativa viável para sistemas de baixo custo, a fim de gerar condições para desenvolvimento de montadores 'assembler', "macro expansor", editor de texto, compiladores,etc..., visto que sua implementação exige ' um mínimo de 16KB de memória principal para funcionar normalmente.

Finalmente, o Suporte C-PASCAL é um 'pacote' aberto, sem nenhuma reserva de direitos, e com farta documentação por se tratar de um trabalho acadêmico. O Suporte pode ser facilmente adaptado a qualquer sistema de microprocessador, sem que haja prejuizo no desempenho de suas funções. Estas característi-' cas dão ao Suporte C-PASCAL um destaque especial visto que os

outros sistemas comerciais, 'packages', não permitem o acesso aos programas fontes e são projetados para um determinado tipo de máquina. Estes fatores fazem do Suporte C-PASCAL uma ferramenta bastante confiável para desenvolvimento e pesquisa de novas técnicas utilizando o micro-computador.

I.2 - DESENVOLVIMENTO DA TESE

O Suporte C-Pascal teve como ponto de partida o sistema "TINY" PASCAL[3], projetado por K.M.Chung e H.Yuen em BASIC (versão 'NORTH STAR BASIC'). Este sistema foi projetado para o micro-computador 'ALTAIR 8800', equipado com 36KB de memória e um sistema operacional 'NORTH STAR DISK'.

Dando continuidade ao sistema acima, desenvolvemos o Suporte C-PASCAL, que teve entre as principais fontes de idéias os seguintes sistemas: PASCAL[1], PASCAL UCSD[4], PASCAL DO PDP 11/70[5], LPM da COBRA[6], e o "TINY PASCAL entre outros. Durante o desenvolvimento da tese destacamos as seguintes tarefas:

1- Estudo de alguns microprocessadores, e em particular do INTEL 8Ø85[7];

2- Análise de alguns códigos intermediários, com maior atenção nos do "TINY"PASCAL e PASCAL UCSD;

3- Especificação da linguagem C-PASCAL;

4- Projeto de máquina C-PASCAL (virtual);

5- Estudo e escolha das melhores estruturas de dados e algorítmos a serem utilizados nos módulos do Suporte;

6- Programação e depuração dos módulos (Compilador, Interpretador e Tradutor);

7- Implementação no micro da INTEL 8Ø85.

O desenvolvimento do projeto Suporte C-PASCAL teve dois ambientes de trabalho: o primeiro no PDP 11/70, usando como linguagem de programação um sub-conjunto do PASCAL (existente no PDP 11/70) compatível com o C-PASCAL; e o segundo no microcomputador (INTEL 8085) usando a linguagem C-PASCAL.

A programação, depuração e implementação dos módulos, fases 5 e 6 descritas acima, tiveram os seguintes passos:

PASSO 1 - Programação do primeiro Compilador C-PASCAL no PDP 11/70 em PASCAL. Este módulo, que identificaremos por compilador/PDP, recebe como entrada um programa C-PASCAL e gera como saída uma listagem de compilação e um arquivo com os códigos intermediários do programa fonte.

PASSO 2 - Programação do Interpretador C-PASCAL no PDP 11/70,em PASCAL. Este módulo, que identificaremos por Interpretador/PDP, recebe como entrada os códigos gerados pelo compilador/PDP, e executa-os simulando a máquina virtual C-PASCAL. Neste estágio podemos testar a geração de código do compilador /PDP e avaliar o desempenho dos códigos intermediários.

PASSO 3 - Programação do Compilador C-PASCAL em C-PASCAL no PDP 11/70. Este passo consistiu, basicamente, da adaptação do programa Compilador/PDP em PASCAL para C-PASCAL. A seguir, compilamos este programa usando o compilador/PDP, e efetuamos exaustivos testes do código obtido com o interpretador/PDP. Os testes de 'performance' e a depuração da lógica do compilador C-PASCAL foram realizados durante esta fase com a compilação de vários programas e até mesmo do próprio compilador C-PASCAL.

PASSO 4 - Programação do Tradutor C-PASCAL no PDP 11/70, em PASCAL. Este módulo, que identificaremos por tradutor/PDP, recebe como entrada o arquivo de código intermediário e gera como saída um arquivo de código de máquina 8080 ou 8085 e uma listagem com os mnemônicos das instruções geradas (assembler). Com a listagem 'assembler' estudamos e testamos pequenos programas' C-PASCAL, a fim de avaliar o código 8085 gerado para determinadas estruturas. Com este estudo fizemos uma série de modificações no tradutor/PDP com o objetivo de reduzir o código de máquina 8085.

PASSO 5 - Programação do Interpretador e do Tradutor em C-PASCAL' no PDP 11/70. Este passo consistiu, basicamente, da adaptação dos programas Interpretador/PDP e Tradutor/PDP em PASCAL para C-PASCAL. A seguir, compilamos estes programas usando o compilador/PDP, e efetuamos exaustivos testes dos códigos intermediários com apoio do interpretador/PDP.

PASSO 6 - Montagem dos módulos do Suporte C-PASCAL para carga no micro-computador 8Ø85. Esta fase consistiu em submeter os códigos intermediários do Suporte (passos 3,5) ao tradutor/PDP, a fim de obtermos três arquivos de códigos de máquina 8085 no PDP 11/70. Para o compilador usamos o arquivo de código intermediário gerado no passo 3, e para o interpretador e tradutor os códigos obtidos no passo 5.

PASSO 7 - Programação do pacote de rotinas em assembler do 8085 no PDP 11/70, com auxílio do Montador da INTEL. Este pacote será usado pelos módulos do Suporte C-PASCAL no micro-computador em tempo de execução. O código de máquina 8Ø85 gerado pelo montador para estas rotinas, é gravado em disco, juntamente com os outros arquivos do passo 6.

PASSO 8 - Programação, no PDP 11/70, de uma rotina auxiliar para efetuar a transferência dos arquivos gerados nos passos 6 e 7 para o micro-computador 8Ø85. Este utilitário foi programado em PASCAL e utiliza um sistema de protocolo simplificado para executar a transmissão. Programação no micro-computador de uma rotina auxiliar para receber e montar na memória do micro cada módulo do Suporte C-PASCAL transmitido do PDP 11/70.

PASSO 9 - Carga do Suporte C-PASCAL no micro-computador INTEL 8085. A conexão foi feita através de uma linha física entre os dois equipamentos, e a cópia realizada diretamente na memória volátil ('RAM'). Ao final de cada transmissão, a região de memória, utilizada para receber cada módulo do Suporte, é salva em disco com auxílio dos utilitários do sistema CP/M (BIOS-BASIC I/O SYSTEM).

PASSO 10 - Depuração do Suporte C-PASCAL no micro-computador. Iniciamos a depuração pelo pacote de rotinas do Suporte com a ajuda do módulo 'DDT - DYNAMIC DEBBUGING TOOL ' existente no CP/M. Por fim, testamos exaustivamente cada módulo do Suporte C-PASCAL (Compilador, Tradutor e Interpretador) diretamente no micro-computador.

O Suporte C-PASCAL está sendo implementado no micro-computador do laboratório da Engenharia Eletrônica da UFRJ. Em paralelo, estão sendo desenvolvidos, por alunos do curso acima, trabalhos de final de curso usando a linguagem C-PASCAL, para

futuramente serem utilizados no laboratório.

O CEPEL já utiliza o Suporte no laboratório de simulação, numa versão compacta (Compilador/Tradutor) para desenvolvimento de programas de aplicação. A versão completa está gravada em disco flexível e foi bastante testada no desenvolvimento deste trabalho.

I.3 - DESCRIÇÃO DA TESE

A descrição do Suporte C-PASCAL, apresentada nesta monografia, se encontra nos capítulos (II,III,IV,V) que resumidamente apresentaremos abaixo:

CAPÍTULO II - COMPILADOR C-PASCAL: Descrevemos a máquina virtual C-PASCAL, o código intermediário e a linguagem C-PASCAL, bem como o método de compilação, as estruturas de dados e a técnica usada no recuperador de erros.

CAPÍTULO III - INTERPRETADOR C-PASCAL: Apresentamos o segundo módulo do Suporte, onde descrevemos sus estrutura, seus comandos e damos um exemplo de aplicação.

CAPÍTULO IV - TRADUTOR C-PASCAL: Apresentamos a técnica de emulação da máquina virtual C-PASCAL, usando os códigos alinhavados. Apresentamos também, o roteiro geral para se projetar um tradutor, bem como o projeto específico do tradutor para o INTEL 8080 ou 8085.

CAPÍTULO V - SUPORTE C-PASCAL: Apresentamos uma visão em conjunto do Suporte C-PASCAL, e mostramos uma aplicação em equipamento de pequeno porte.

CAPÍTULO II

COMPILADOR C-PASCAL

II.1 - ESPECIFICAÇÃO DA LINGUAGEM

O desenvolvimento da linguagem C-PASCAL teve como objetivo oferecer ao usuário de micro-computador uma ferramenta para desenvolvimento de projetos. De um modo geral, os usuários destas máquinas dispõem apenas de linguagens de montagem ou linguagens de médio nível[8] sem o mecanismo de recursão[9] .

Estando o projeto desta linguagem baseado no PASCAL[1], ela assimilou algumas das suas características, tais como:clareza recursão e estruturação. Entretanto para atender ao objetivo de facilitar a programação em micro-computadores, onde a interação usuário-máquina é mais estreita,foram introduzidos novos conceitos à linguagem, tais como: chamada de sub-rotinas externas ao programa (comando CALL), acesso direto à memória (variável MEM), a não definição rígida de tipos e outros que serão explicados mais adiante. Além disto, para permitir sua utilização em equipamentos com pequena memória (mínimo 16KB) foram suprimidas certas características da linguagem base, que onerariam muito o compilador em tempo e espaço, e cujo percentual de utilização não justificam tal custo[10] para um projeto que é apenas a base para desenvolvimento de outros, inclusive o de um compilador da linguagem PASCAL completa.

II.1.1 - SUMÁRIO DA LINGUAGEM

Um programa C-PASCAL está caracterizado por duas partes:

a-) A parte descritiva, onde são definidos os dados a serem manipulados pelo programa, e que é feita através das declarações de LABEL, CONST, VAR,e das declarações dos sub-programas (PROCEDURE ou FUNCTION)

b-) A segunda parte, corpo do programa, com as ações a serem executadas, que são descritas através dos comandos.

Os dados são representados por variáveis, e seus no

mes devem constar nas declarações. Entretanto as declarações nesta linguagem não vinculam, rigidamente, o tipo à forma de usá-la, isto é, uma variável do tipo inteiro pode receber atribuição de um valor do tipo caracter ('B'), já que o valor do código ASCII do caracter está dentro do domínio dos valores possíveis de uma variável inteira. Portanto a declaração determina o domínio dos valores que a variável pode assumir, mas não restringe a forma de representá-la. Desta forma, o tipo básico de dado é o inteiro, e assume-se como definido implicitamente (mas não declarado) os tipos caracter e lógico. O caracter, representado por um símbolo entre aspas, é armazenado internamente pelo valor do seu código ASCII. O valor lógico é representado pelo 'bit' de mais baixa ordem de qualquer variável, sendo o valor 1 avaliado como condição verdadeira e o valor Ø como condição falsa.

O tipo estruturado é o arranjo linear de variáveis, todas do tipo inteiro. Este vetor tem seus limites definidos na declaração. O acesso aos elementos é feito através do cálculo de um índice, que é testado para verificar se está dentro dos limites declarados.

Para facilitar a interação usuário-máquina, a linguagem considera a memória do equipamento como um vetor predefinido de limites entre o menor e o maior endereço acessável, tal como o intervalo (Ø..16K-1), cujo nome é MEM. Portanto, o acesso a qualquer posição de memória se faz referindo-se a ' MEM [<expressão>] , onde a expressão irá representar um endereço absoluto. Para este vetor não há nenhum esquema de proteção à memória, que impeça o acesso a áreas do monitor ou do próprio código de programa. Deste modo, MEM é um instrumento poderoso , cômodo e ao mesmo tempo perigoso, para o usuário de micro-computador menos experiente.

As variáveis declaradas no programa principal, são ditas globais, e seu escopo está definido em todo programa. Variáveis definidas nos sub-programas só podem ser referenciadas dentro do corpo destes, e são chamadas de variáveis locais. A referência às variáveis não locais ao corpo do procedimento são resolvidas pela regra do escopo estático[11].

As ações, a serem efetuadas, são expressas sob a forma de comandos de atribuição. O controle destas ações é feito por

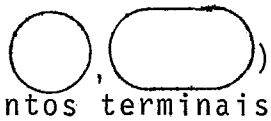
comandos condicionais e iterativos. A interação com o usuário é feita através dos comandos de leitura e escrita.

O comando de atribuição especifica um novo valor a ser recebido por uma variável. Este valor é obtido pela avaliação da expressão a direita do sinal de atribuição (:=).

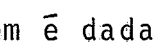
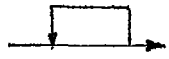
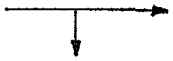
As expressões são constituidas de variáveis, constantes, operadores e funções que se relacionam e atuam segundo regras de precedência definidas na linguagem. O resultado final da avaliação é o novo valor da variável associada ao comando ' de atribuição. A linguagem C-PASCAL permite o uso de expressões 'mistas', onde podemos efetuar operações entre uma variável inteira e o resultado de uma sub-expressão lógica, bem como o tipo do resultado é determinado pelo contexto em que está a expressão.

II.1.2 - NOTAÇÃO, TERMINOLOGIA E VOCABULÁRIO

A notação utilizada para representar a sintaxe da linguagem é o Diagrama de Sintaxe [1].

As figuras circulares () envolvem os símbolos especiais, que são os elementos terminais da linguagem.

Os elementos sintáticos que são os não-terminais, como variável, expressão, etc, estão nas figuras retangulares.

A sequência na qual devem estar dispostos os componentes da linguagem é dada por uma semi-reta orientada (), as repetições indicadas por  e as alternativas por  .

O vocabulário consiste de elementos básicos classificados por letras, dígitos e símbolos especiais. Para definir os elementos básicos apresentamos abaixo na forma padronizada de BNF (BACKUS-NAUR FORM).

```
          <letra> ::= 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
          <dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<símbolo especial> ::= + | - | * | = | <> | < | > | <= | >= | ( | ) | [ | ]
                       (* | *) | := | . | , | ; | : | & | % | # | $ | DIV
                       MOD | OR | AND | NOT | IF | THEN | ELSE
                       CASE | OF | REPEAT | UNTIL | WHILE | DO
                       FOR | TO | DOWNTO | BEGIN | END | GOTO
                       CONST | VAR | LABEL | ARRAY | FUNCTION
                       PROCEDURE | PROGRAM
```

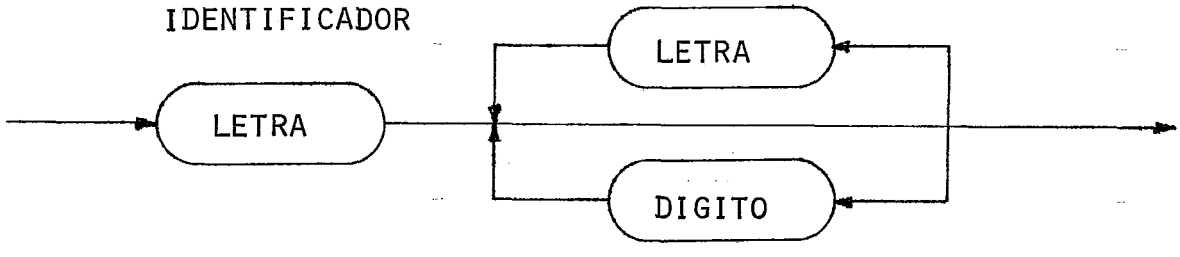
<dígito hexa> ::= Ø | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F

A construção

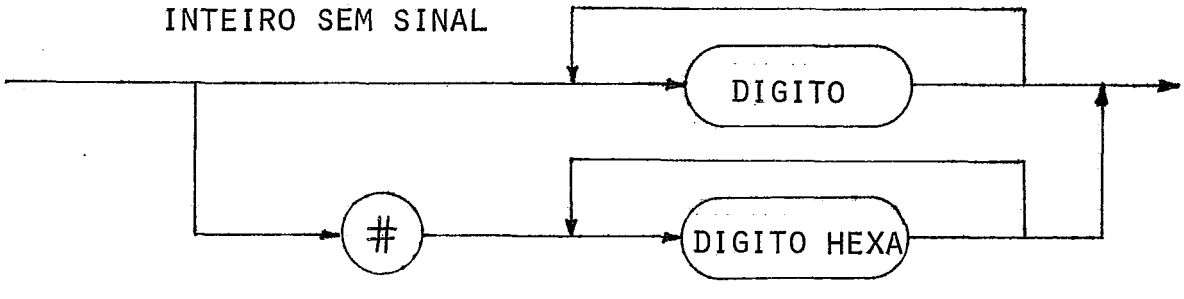
(* qualquer sequência de símbolos,menos '*)' *)

pode ser inserida entre dois identificadores, números ou símbolos especiais, sem alterar o significado do programa. Esta construção é usada para inserir comentários no programa fonte C-PASCAL.

II.1.3 - IDENTIFICADORES, NÚMEROS E CONSTANTES



Os identificadores servem para nomear constantes, variáveis, procedimentos e funções, não sendo permitido usar palavras reservadas como identificadores, e sua declaração deve ser única dentro do escopo de validade. No C-PASCAL não existe limite para o tamanho do identificador, e todos os símbolos (letras ou dígitos) são usados para sua identificação.

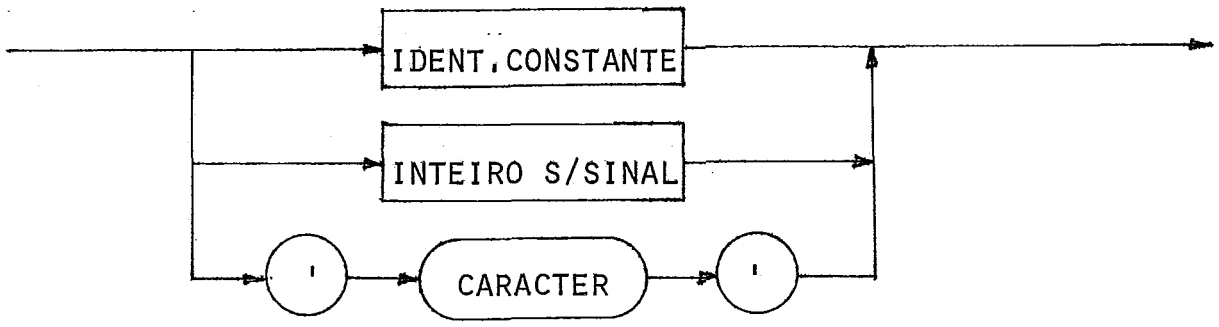


A notação decimal é usada para representar os números, assim como o formato hexadecimal. Os inteiros são avaliados e armazenados internamente na forma binária em palavras de 16-bits. O domínio dos inteiros (-32.768.. 32.767).

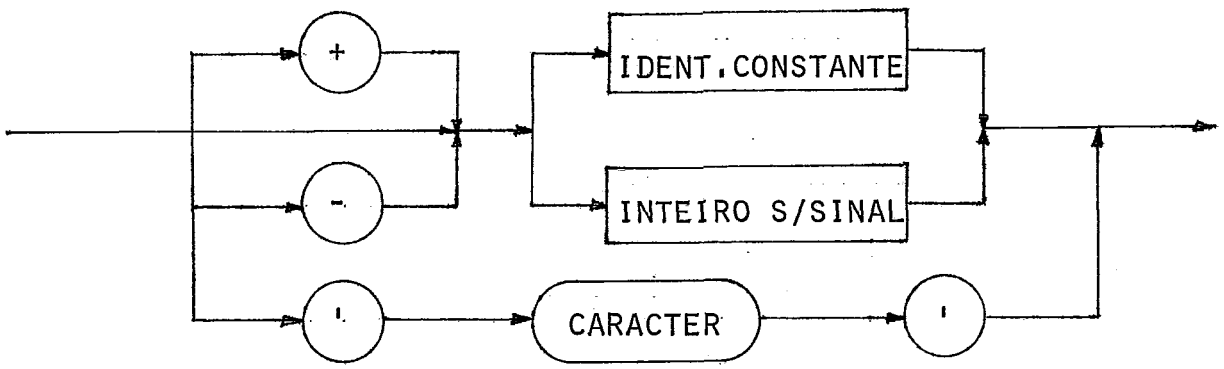
O formato hexadecimal, indicado pelo símbolo ' # ', foi introduzido para faciliatr o endereçamento de memória, bem

como o uso de máscaras em campos de 'bits' de uma variável. Os inteiros hexas são avaliados no intervalo (#ØØØØ.. #FFFF).

CONSTANTE SEM SINAL



CONSTANTE

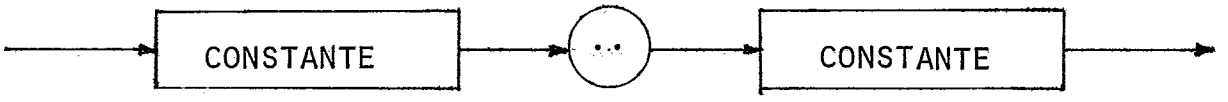


A declaração de uma constante introduz no programa um identificador como sinônimo de uma constante. As constantes negativas são representadas internamente pelo complemento a 2. Um caracter entre apóstrofos ('A') é denotado como uma constante 'CHAR' , e seu valor interno é obtido pelo respectivo código ASCII .

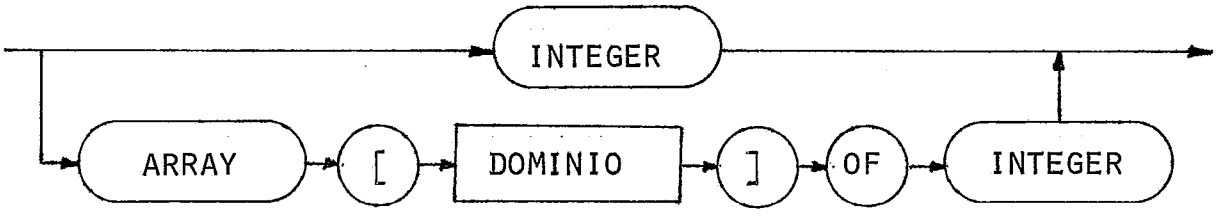
Por exemplo:	'*'	'G'	'3'	'A'
	#Ø	#FF		#A
	Ø	255	-1	10

II.1.4 - DEFINIÇÃO DE TIPOS

DOMÍNIO



TIPO



Os elementos de dados (numéricos, lógicos, caracteres, rótulos, etc...) são todos representados internamente na forma binária, em palavras de 16-bits e complemento a 2.

As linguagens de programação permitem a abstração desta representação interna, por meio da especificação dos conjuntos de elementos de dados com os respectivos operadores. A linguagem C-PASCAL não faz restrição quanto ao tipo e à forma de uso dos seus operadores com suas variáveis, que inicialmente são declaradas como inteiras.

Por exemplo:
```
FOR  I := 'A'  TO  '2'  DO  WRITE  (TTY,&I) ;
     X :=  B<>C ;
     Y :=  B = C ;
     Z :=  B AND C ;
     IF  X  THEN  <comando > ELSE < comando > ;
```

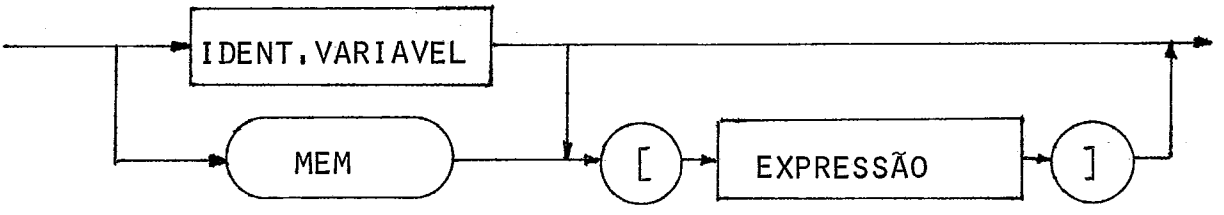
Os tipos padrões CHAR e BOOLEAN não existem explicitamente na sintaxe da linguagem, porém estas duas características podem ser usadas normalmente em programação C-PASCAL. A condição 'booleana' é avaliada pelo dígito binário de mais baixa ordem, sendo condição verdadeira o valor 1 e falsa o valor Ø.O resultado de uma expressão lógica pode ser Ø ou 1, porém o resultado de uma expressão aritmética quando analisado como lógico somente o 'bit' de mais baixa ordem é observado.

O tipo estruturado ARRAY consiste de um número fixo de elementos agrupados numa estrutura vetorial, sendo todos do tipo inteiro. Cada componente deste arranjo, pode ser explicita

mente referenciado e manipulado pelo programador.

II.1.5 - VARIÁVEIS

VARIÁVEL



As variáveis são todas representadas internamente em palavras de 16-bits, sendo um 'bit' para sinal e o restante magnitude em complementação a 2.

A variável simples é especificada pelo seu identificador.

A variável indexada é referenciada pelo identificador do arranjo seguido da expressão de índice, entre colchetes ' IDENT [<exp>] , que especificará um único elemento do vetor. O índice da variável indexada é obtido pela avaliação da expressão, e em seguida testado dentro do domínio. Se ocorrer um índice inválido, durante a execução do programa, será dada mensagem de erro correspondente. O teste de índice é oneroso em espaço e tempo de execução, portanto o seu uso é controlado pelo programador que pode optar pela inserção ou remoção deste teste no código,gerado durante a fase de compilação, com a aplicação da diretiva (*?*).

Por exemplo:
```
PROGRAM TESTE-DE-INDICE :
    VAR VETOR : ARRAY [15..30] OF INTERGER; I:INTERGER;
    BEGIN
        VETOR [I]:= <exp> ; (* COM TESTE DE ÍNDICE*)
    (*?*)
        VETOR [I]:= <exp> ; (* SEM TESTE DE ÍNDICE*)
    (*?*)
        VETOR [I]:=  10   ; (* COM TESTE DE ÍNDICE*)
    END.
```

Com o objetivo de permitir o acesso direto à memória do micro-computador, criamos um identificador para especificar

o vetor memória denotado por MEM[<exp>] . Cada posição de memória é referenciada como uma variável indexada, sendo o resultado da expressão de índice o endereço absoluto de memória a ser acessada. Cada componente deste vetor representa, no caso do 8Ø8Ø/8Ø85 da INTEL, uma palavra de 8-bits (1'byte'), sendo possível ler ou escrever valores no domínio [Ø..255] ou[#ØØ..#FF]. Na avaliação de uma expressão, o valor de um elemento do vetor MEM é estendido para dezesseis 'bits', ficando compatível com as demais variáveis, sendo seu valor mantido na parte baixa da cadeia estendida.

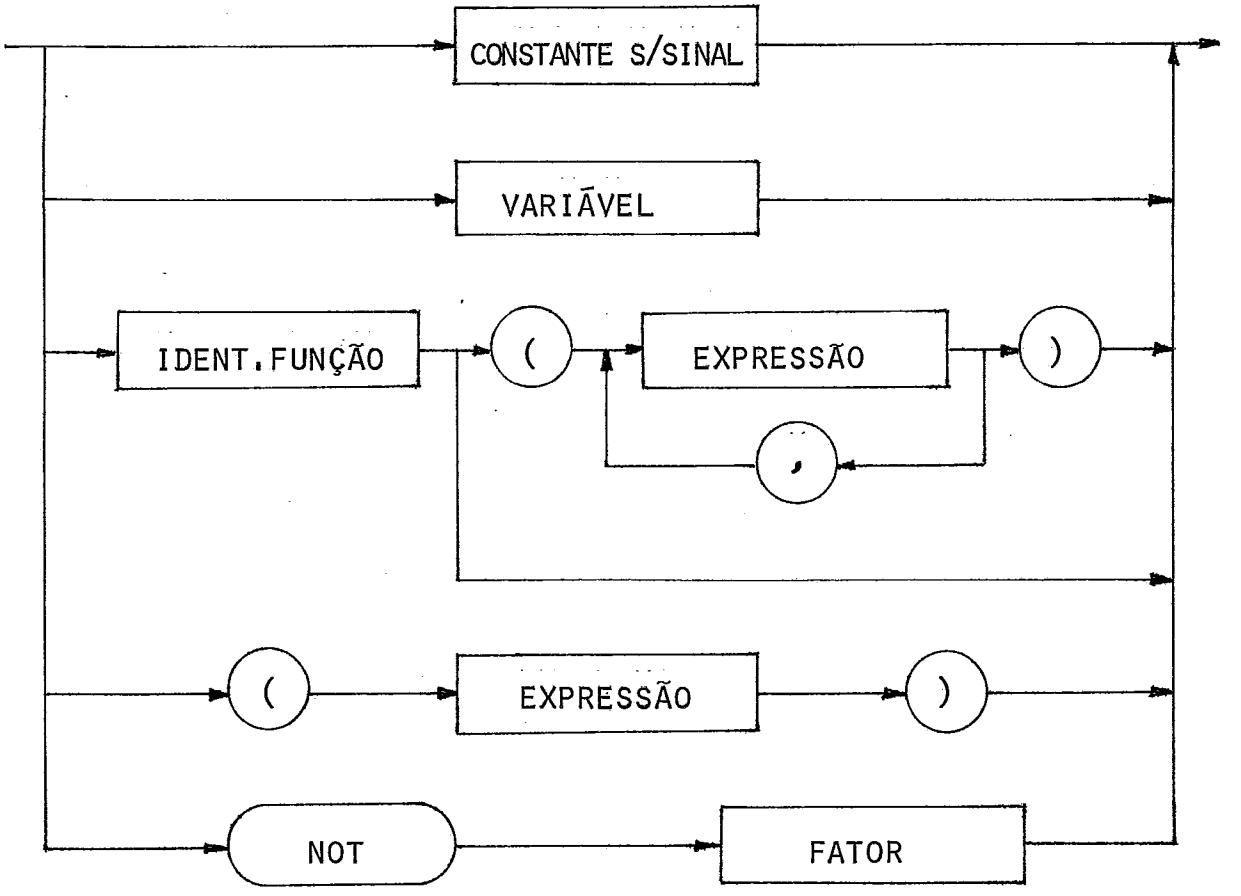
Por exemplo:
```
I := I * MEM [15] + MEM[18] ;
IF MEM [I] = 'A' THEN <comando> ;
```

```
PROGRAM ZERARMEMÓRIA ;
VAR I : INTERGER;
BEGIN
I := #EFFF ;
REPEAT  I := I+1 ;
        MEM [I] := Ø  ;
UNTIL   MEM [I] <> Ø  ;
END .
```

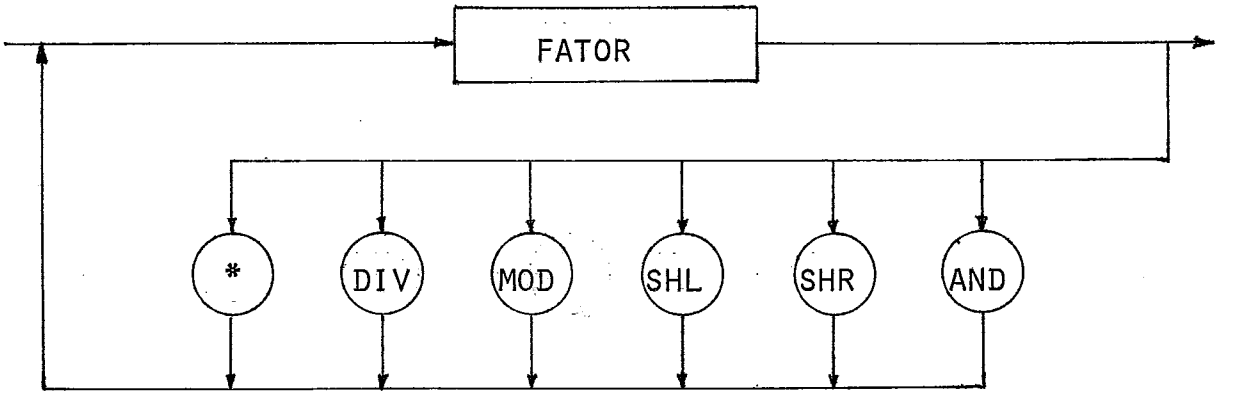
Certas concepções de 'hardware' fornecem a constante 255 às leituras feitas fora da memória real, o programa ZERAR MEMORIA, descrito acima, fará uma varredura da memória, preenchendo-a com zero, a partir do endereço (#EFFF) até encontrar o final da memória real, que corresponde a ler um valor diferente de zero escrito na ação anterior.

II.1.6 - EXPRESSÃO

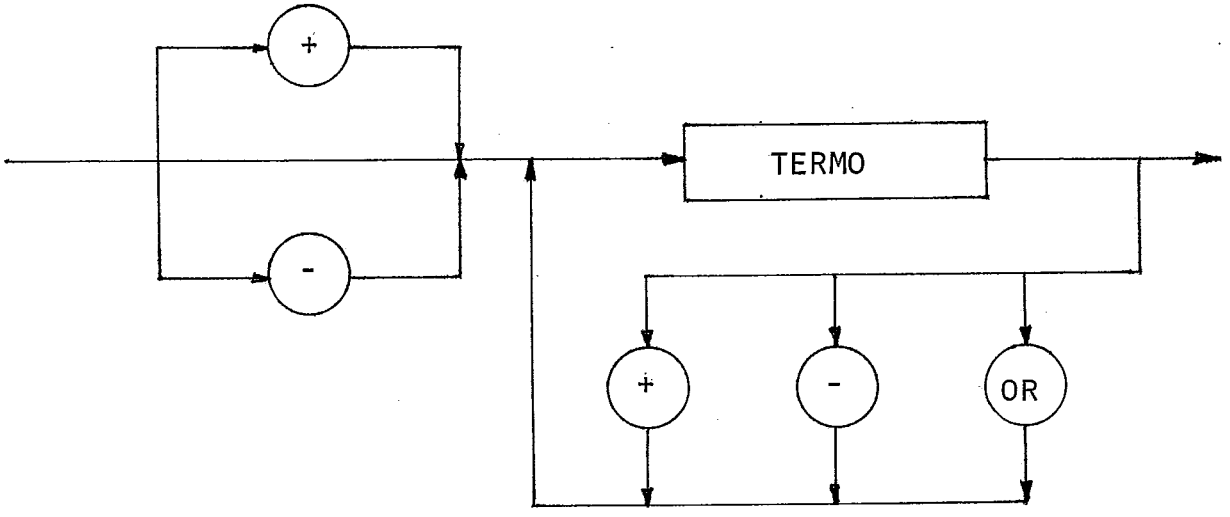
FATOR



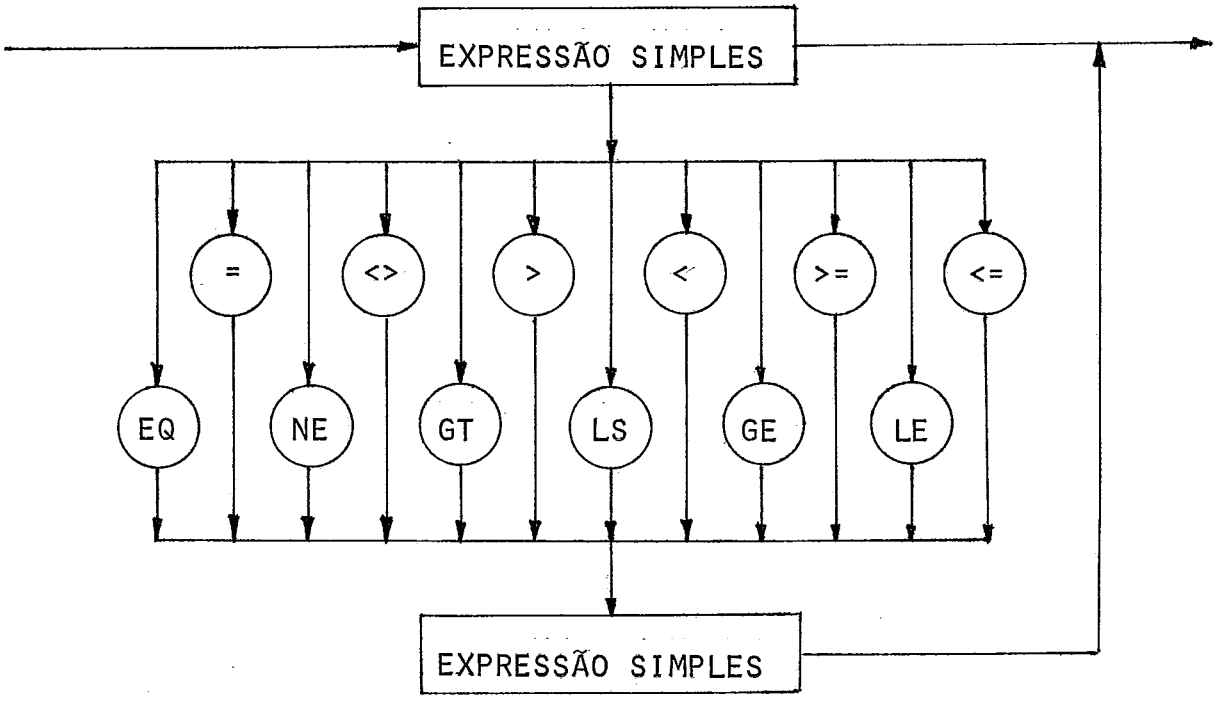
TERMO



EXPRESSÃO SIMPLES



EXPRESSÃO

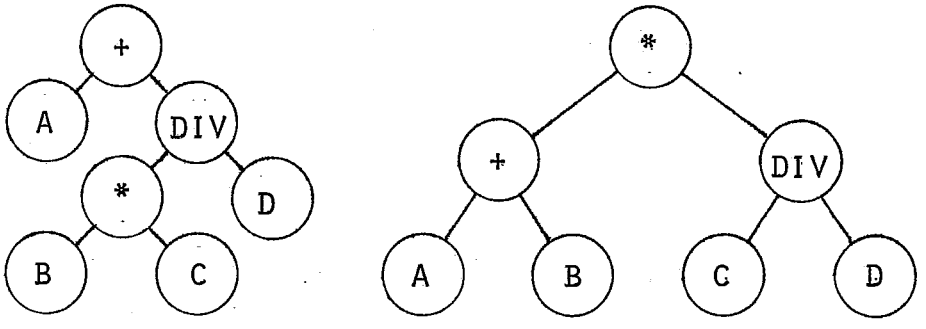


A expressão é uma construção que obedece determinadas regras de formação a fim de obter valores de variáveis, e gerar novos valores com a aplicação dos operadores. As expressões são constituidas de operadores e operandos, isto é, variáveis, constantes, e funções.

As regras de avaliação especificam as precedências dos operadores, classificados em quatro grupos: O operador "NOT" tem a maior precedência, seguido pelo grupo dos operadores de multiplicação (*, DIV, MOD, SHL, SHR, AND), depois pelos operadores de adição (+, -, OR) e com mais baixa precedência o grupo dos operadores relacionais (=,<>,>=,<=,> ,< ,). A precedência entre operadores do mesmo grupo é resolvida pelo sentido esquerda para direita, na avaliação da expressão.

As regras de precedência estão definidas na própria sintaxe da linguagem, como apresentamos nos diagramas referentes a expressão (FATOR, TERMO, EXPRESSÃO SIMPLES, EXPRESSÃO). Por exemplo:

A + B * C DIV D e (A+B) * (C DIV D)



2 + 5 * 8 DIV 4 = 12 (2+5) * (8 DIV 4) = 9

OPERADOR NOT

O operador NOT representa a negação lógica do seu operando. Na linguagem C-PASCAL, este operador quando aplicado a expressão acarretará uma complementação a 1 do resultado.

OPERADORES DE MULTIPLIÇACÃO

Os operadores de multiplicação são todos binários, e usam o (topo) e (topo-1) da pilha de avaliação como seus operandos, e após a operação armazenam o resultado no (topo).

A operação de multiplicação (*) é executada entre

dois operandos da pilha, e o produto calculado com precisão simples (16-bits).

A operação de divisão (DIV) é executada entre o dividendo (topo) e o divisor (topo-1), e seu quociente é truncado (não é feito arredondamento)[1] . Em tempo de execução são detectados dois erros nesta operação: DIVISÃO POR ZERO e DIVISÃO POR -32.768

Por exemplo:

```
 5 DIV 3 =  1        5 DIV -3 = -1
-5 DIV 3 = -1       -5 div -3 =  1
```

A operação resto da divisão (MOD) obedece a seguinte regra: a MOD b = a - (a DIV b) * b

Por exemplo:

```
 5 MOD 3 =  2        5 MOD -3 =  2
-5 MOD 3 = -2       -5 MOD -3 = -2
```

As operações de deslocamento (SHL,SHR) foram introduzidas no C-PASCAL para aproveitar com mais eficiência as instruções de 'SHIFT' disponíveis nos micro-computadores, bem como permitir o deslocamento lógico de uma cadeia de'bits' associada a uma variável. A característica do deslocamento lógico é o preenchimento com zeros da parte deslocada.

Por exemplo:

```
5 * 2 = 5 SHL 1 =  1Ø
#ØØFF SHL 4 = #ØFFØ
15 DIV 4 = 15 SHR 2 = 3
```

A operação AND gera, como resultado, uma cadeia de 16-bits proveniente de um AND lógico feito entre os dois operandos 'bit' a 'bit'.

Por exemplo:

```
#FAC5 AND #ØØFØ = #ØØCØ
WHILE (A>B) AND (B<>C) DO <comando> ;
WHILE (B AND #ØØFF) < 15 DO  <comando> ;
```

OPERADORES DE ADIÇÃO

A operação soma (+) é executada diretamente pela maioria dos microprocessadores, porque normalmente eles dispõem de instruções de soma em 16-bits.

O operador menos (-) pode ser usado como operador

```
-----------------------------------#
  CARGA DAS ROTINAS                #
     C-PASCAL                      #
-----------------------------------#

-IROTPAS.COM
-R
NEXT   PC
2600 0100
-----------------------------------#
  CARGA DO COMPILADOR              #
      C-PASCAL                     #
-----------------------------------#

-ICOMPAS.COM
-R500
NEXT   PC
2600 0100
-----------------------------------#
  CARGA DO PROGRAMA                #
    TORRE DE HANOI                 #
-----------------------------------#

-ITHANOI.CPA
-R2600
NEXT   PC
2C00 0100
-SF6FB
F6FB CD C3
2600   0001   PROGRAM TORREDEHANOI ;
2604   0002
2604   0003     CONST PERO = 0 ;
2604   0004           ORIGEM = 1 ;
2604   0005           DESTINO = 3 ;
2604   0006           AUXILIAR = 2 ;
2604   0007     VAR   NUMDISCOS : INTEGER ;
2604   0008
2604   0009     PROCEDURE TROCATORRE(ALTURA,TORI,TDES,TAUX:INTEGER) ;
2604   000A
2604   000B       PROCEDURE MOVEDISCO (RETIRAR,COLOCAR:INTEGER) ;
2604   000C       BEGIN
2604   000D           WRITELN(PERO,%RETIRAR,' ==> ',%COLOCAR)
2634   000E       END ;
2638   000F
2638   0010     BEGIN      (* PROCEDURE TROCATORRE *)
2638   0011         IF ALTURA > 0
2640   0012         THEN BEGIN
2648   0013              TROCATORRE (ALTURA-1,TORI,TAUX,TDES) ;
2664   0014              MOVEDISCO (TORI,TDES) ;
2670   0015              TROCATORRE (ALTURA-1,TAUX,TDES,TORI)
268C   0016              END
268C   0017     END ;
2690   0018
2690   0019   BEGIN  (* PROGRAMA PRINCIPAL *)
2694   001A
2694   001B       WRITE(PERO,'NUMERO DE DISCOS NA ORIGEM ?> ') ;
2714   001C       READLN(PERO,%NUMDISCOS) ;
2720   001D       TROCATORRE (NUMDISCOS,ORIGEM,DESTINO,AUXILIAR)
2734   001E
2734   001F   END .
```